ગુજરાત રાજ્યના શિક્ષણવિભાગના પત્ર-ક્રમાંક
મશબ/1214/15/છ, તા. 29-1-2014થી–મંજૂર

કમ્પ્યૂટર-અધ્યયન

ધોરણ 12

પ્રતિજ્ઞાપત્ર

ભારત મારો દેશ છે.
બધાં ભારતીયો મારાં ભાઈબહેન છે.
હું મારા દેશને ચાહું છું અને તેના સમૃદ્ધ અને
વૈવિધ્યપૂર્ણ વારસાનો મને ગર્વ છે.
હું સદાય તેને લાયક બનવા પ્રયત્ન કરીશ.
હું મારાં માતાપિતા, શિક્ષકો અને વડીલો પ્રત્યે આદર રાખીશ
અને દરેક જણ સાથે સભ્યતાથી વર્તીશ.
હું મારા દેશ અને દેશબાંધવોને મારી નિષ્ઠા અર્પું છું.
તેમનાં કલ્યાણ અને સમૃદ્ધિમાં જ મારું સુખ રહ્યું છે.

કિંમત : ₹ 105.00

ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ
'વિદ્યાયન', સેક્ટર 10-એ, ગાંધીનગર-382010

© ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ, ગાંધીનગર
આ પાઠ્યપુસ્તકના સર્વ હક ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળને હસ્તક છે.
આ પાઠ્યપુસ્તકનો કોઈ પણ ભાગ કોઈ પણ રૂપમાં ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળના નિયામકની લેખિત પરવાનગી વગર પ્રકાશિત કરી શકાશે નહિ.

વિષય-સલાહકાર
પ્રો. આર. પી. સોની

લેખન-સંપાદન
ડૉ. હર્ષલ આરોલકર (કન્વીનર)
શ્રી જીજ્ઞેશભાઈ સ્માર્ત
શ્રી જ્યોતિકાબહેન દોશી
શ્રી તૃપ્તિબહેન ડોડિયા
શ્રી હાર્દિકભાઈ જોશી

અનુવાદ
શ્રી રોહિતભાઈ દોશી
શ્રી ગિરીશ બ્રહ્મભટ્ટ
શ્રી સાકેત દવે
શ્રી રજનીકાન્ત પંડ્યા

સમીક્ષા
શ્રી બીમલભાઈ રાવલ
શ્રી પંકજકુમાર શુક્લ
શ્રી રાજેશ્રીબહેન પડિયા
શ્રી નિશીતાબહેન ગાંધી
શ્રી સેજલબહેન ત્રિવેદી
ડૉ. વિમલભાઈ પંડ્યા
શ્રી હિમાંશુભાઈ મણિયાર

ભાષાશુદ્ધિ
શ્રી ઓમપ્રકાશ દવે

સંયોજન
શ્રી આશિષ એચ. બોરીસાગર
(વિષય-સંયોજક : ગણિત)

નિર્માણ-આયોજન
શ્રી સી. ડી. પંડ્યા
(નાયબ નિયામક : શૈક્ષણિક)

મુદ્રણ-આયોજન
શ્રી હરેશ એસ. લીમ્બાચીયા
(નાયબ નિયામક : ઉત્પાદન)

પ્રસ્તાવના

માધ્યમિક અને ઉચ્ચતર માધ્યમિક શિક્ષણ બોર્ડે ઓપન સોર્સ ઑપરેટિંગ સિસ્ટમ અને તેને સુસંગત વિવિધ મુદ્દાઓ માટેના કમ્પ્યૂટર-અધ્યયનને લગતા ઓપન સોર્સ સૉફ્ટવેર આધારિત નવો અભ્યાસક્રમ તૈયાર કર્યો છે. આ અભ્યાસક્રમ ગુજરાત સરકાર દ્વારા મંજૂર કરવામાં આવ્યો છે.

ગુજરાત સરકાર દ્વારા મંજૂર થયેલા **ધોરણ 12**ના **કમ્પ્યૂટર-અધ્યયન** વિષયના નવા અભ્યાસક્રમ અનુસાર તૈયાર કરવામાં આવેલું આ પાઠ્યપુસ્તક વિદ્યાર્થીઓ સમક્ષ મૂકતાં મંડળ આનંદ અનુભવે છે.

આ વિષયનું અંગ્રેજી માધ્યમનું પાઠ્યપુસ્તક પ્રસિદ્ધ કરતાં પહેલાં એની હસ્તપ્રતની આ સ્તરે શિક્ષણકાર્ય કરતા શિક્ષકો અને તજ્જ્ઞો દ્વારા સર્વાંગી સમીક્ષા કરાવવામાં આવી છે અને તેમનાં સૂચનો અનુસાર હસ્તપ્રતમાં યોગ્ય સુધારા-વધારા કર્યા પછી આ પાઠ્યપુસ્તક પ્રસિદ્ધ કરવામાં આવ્યું છે. અંગ્રેજી માધ્યમના પાઠ્યપુસ્તકનો આ ગુજરાતી અનુવાદ છે.

પ્રસ્તુત પાઠ્યપુસ્તકને વિષયવસ્તુલક્ષી, રસપ્રદ અને ક્ષતિરહિત બનાવવા માટે મંડળે પૂરતી કાળજી લીધી છે, તેમ છતાં શિક્ષણમાં રસ ધરાવનાર વ્યક્તિઓ પાસેથી પુસ્તકની ગુણવત્તા વધારે તેવાં સૂચનો આવકાર્ય છે.

ડૉ. ભરત પંડિત
નિયામક
તા. 7-10-2014

ડૉ. નીતિન પેથાણી
કાર્યવાહક પ્રમુખ
ગાંધીનગર

પ્રથમ આવૃત્તિ : 2014, પુન:મુદ્રણ : 2014

પ્રકાશક : ગુજરાત રાજ્ય શાળા પાઠ્યપુસ્તક મંડળ, 'વિદ્યાયન', સેક્ટર 10-એ, ગાંધીનગર વતી ભરત પંડિત, નિયામક

મુદ્રક :

મૂળભૂત ફરજો

ભારતના દરેક નાગરિકની ફરજ નીચે મુજબ રહેશે :*

(ક) સંવિધાનને વફાદાર રહેવાની અને તેના આદર્શો અને સંસ્થાઓનો, રાષ્ટ્રધ્વજનો અને રાષ્ટ્રગીતનો આદર કરવાની;

(ખ) આઝાદી માટેની આપણી રાષ્ટ્રીય લડતને પ્રેરણા આપનારા ઉમદા આદર્શોને હૃદયમાં પ્રતિષ્ઠિત કરવાની અને અનુસરવાની;

(ગ) ભારતનાં સાર્વભૌમત્વ, એકતા અને અખંડિતતાનું સમર્થન કરવાની અને તેમનું રક્ષણ કરવાની;

(ઘ) દેશનું રક્ષણ કરવાની અને રાષ્ટ્રીય સેવા બજાવવાની હાકલ થતાં, તેમ કરવાની;

(ચ) ધાર્મિક, ભાષાકીય, પ્રાદેશિક અથવા સાંપ્રદાયિક ભેદોથી પર રહીને, ભારતના તમામ લોકોમાં સુમેળ અને સમાન બંધુત્વની ભાવનાની વૃદ્ધિ કરવાની, સ્ત્રીઓના ગૌરવને અપમાનિત કરે, તેવા વ્યવહારો ત્યજી દેવાની;

(છ) આપણી સમન્વિત સંસ્કૃતિના સમૃદ્ધ વારસાનું મૂલ્ય સમજી તે જાળવી રાખવાની;

(જ) જંગલો, તળાવો, નદીઓ અને વન્ય પશુપક્ષીઓ સહિત કુદરતી પર્યાવરણનું જતન કરવાની અને તેની સુધારણા કરવાની અને જીવો પ્રત્યે અનુકંપા રાખવાની;

(ઝ) વૈજ્ઞાનિક માનસ, માનવતાવાદ અને જિજ્ઞાસા તથા સુધારણાની ભાવના કેળવવાની;

(ટ) જાહેર મિલકતનું રક્ષણ કરવાની અને હિંસાનો ત્યાગ કરવાની;

(ઠ) રાષ્ટ્ર પુરુષાર્થ અને સિદ્ધિનાં વધુ ને વધુ ઉન્નત સોપાનો ભણી સતત પ્રગતિ કરતું રહે એ માટે, વૈયક્તિક અને સામૂહિક પ્રવૃત્તિનાં તમામ ક્ષેત્રે શ્રેષ્ઠતા હાંસલ કરવાનો પ્રયત્ન કરવાની.

(ડ) માતા-પિતાએ અથવા વાલીએ 6 વર્ષથી 14 વર્ષ સુધીની વયના પોતાના બાળક અથવા પાલ્યને શિક્ષણની તકો પૂરી પાડવાની.

*ભારતનું સંવિધાન : કલમ 51-क

અનુક્રમણિકા

1.	કમ્પોઝરનો ઉપયોગ કરી HTML ફૉર્મની રચના	1
2.	કેસ્કેડિંગ સ્ટાઇલશીટ અને જાવાસ્ક્રિપ્ટ	25
3.	કમ્પોઝરનો ઉપયોગ કરી સરળ વેબસાઇટની રચના	52
4.	ઇ-કૉમર્સનો પરિચય	72
5.	એમ-કૉમર્સનો પરિચય	90
6.	ઑબ્જેક્ટ આધારિત ખ્યાલો	116
7.	જાવાની મૂળભૂત બાબતો	127
8.	જાવામાં ક્લાસ અને ઑબ્જેક્ટ	158
9.	ઍરે અને સ્ટ્રિંગનો ઉપયોગ	186
10.	જાવામાં અપવાદરૂપ પરિસ્થિતિનું વ્યવસ્થાપન	203
11.	ફાઇલ-વ્યવસ્થાપન	220
12.	લેટેક્સની મદદથી દસ્તાવેજનું પ્રકાશન	241
13.	અન્ય ઉપયોગી નિઃશુલ્ક ટૂલ્સ અને સેવાઓ	260

આ પાઠ્યપુસ્તક વિશે...

પ્રિય શિક્ષકો,

કમ્પ્યૂટર-સાક્ષરતાને ખૂબ જ ઝડપથી વિસ્તારવાના ધ્યેય સાથે, ગુજરાત સરકારે ICT@School કાર્યક્રમ અંતર્ગત રાજ્યની 6000 કરતાં વધુ અનુદાનિત શાળાઓને અદ્યતન કમ્પ્યૂટર-સંસાધનો પૂરાં પાડ્યાં છે. નવી નીતિની પહેલરૂપે બધી શાળાઓને ઉબુન્ટુ (લિનક્સનું ભિન્ન સ્વરૂપ) ઑપરેટિંગ સિસ્ટમ અને અન્ય મુક્તપણે ઉપલબ્ધ નિ:શુલ્ક (ઓપનસૉર્સ) સૉફ્ટવેર પણ પૂરાં પાડવામાં આવ્યાં છે, જેથી લાઇસન્સની ચિંતા વગર મુક્ત રીતે સૉફ્ટવેર વાપરી શકે તેમજ તેની આપ-લે પણ કરી શકે. અગાઉનાં પાઠ્યપુસ્તકો મોટે ભાગે માલિકીહક ધરાવતા (પ્રોપ્રાઇટરી) સૉફ્ટવેર માટે લખાયાં હતાં, તેથી નવા અભ્યાસક્રમને આધારે નવેસરથી પાઠ્યપુસ્તકો તૈયાર કરવાની જરૂરિયાત હતી. 8મું ધોરણ પ્રાથમિક વિભાગમાં તબદીલ થવાને કારણે પણ આ ખૂબ જ જરૂરી હતું. આથી, ધોરણ 9થી 12 માટે કમ્પ્યૂટર-અધ્યયન માટેના વિવિધ વિષયો માટે મુક્તપણે ઉપલબ્ધ ઑપરેટિંગ સિસ્ટમ અને તેને અનુરૂપ સૉફ્ટવેર ટૂલ્સ પર આધારિત નવો અભ્યાસક્રમ તબક્કાવાર રીતે પૂરો પાડવામાં આવ્યો.

વિદ્યાર્થીઓ હવે ઑપરેટિંગ સિસ્ટમ, ઓપન ઑફિસના ઘટકો, HTML, સી પ્રોગ્રામિંગ, ઇન્ટરનેટ અને વેબસર્ફિંગ જેવા મૂળભૂત કમ્પ્યૂટર સૉફ્ટવેરથી માહિતગાર છે. 'કમ્પ્યૂટર-અધ્યયન' વિષયની શ્રેણીમાં 12મા ધોરણનું આ ચોથું પાઠ્યપુસ્તક છે. આ પુસ્તકનો હેતુ HTML ફૉર્મની રચના, જાવાસ્ક્રિપ્ટ, કમ્પોઝરનો ઉપયોગ કરી સરળ વેબસાઇટની રચના અને ઇ-કૉમર્સ તથા એમ-કૉમર્સનો પરિચય આપવાનો છે. આ ઉપરાંત, ઓબ્જેક્ટ ઓરિએન્ટેડ અભિગમના પરિચય બાદ મૂળભૂત જાવા ભાષા સમજાવવામાં આવી છે, જે વેબ-વિનિયોગો માટે વ્યાપકપણે ઉપયોગમાં લેવાતી ભાષા હોવાને કારણે ઘણી પ્રચલિત છે. અંતમાં, વિદ્યાર્થીઓની જાણકારી માટે લેટેક્સ જેવા પ્રચલિત ઓપનસૉર્સ પૅકેજ અને અન્ય પ્રકીર્ણ ટૂલ્સનો પરિચય આપવામાં આવ્યો છે.

અમે આશા રાખીએ છીએ કે, આવરી લીધેલો અભ્યાસક્રમ વિદ્યાર્થીઓને કમ્પ્યૂટર એપ્લિકેશન્સ અંગે સ્પષ્ટ સમજ મેળવવામાં ઉપયોગી બનશે અને આપ ઓપનસૉર્સ સૉફ્ટવેર ટૂલ્સનો ઉપયોગ કરી શિક્ષણ આપવામાં અને પ્રાયોગિક કાર્યમાં આનંદ અનુભવશો.

પ્રિય વિદ્યાર્થીઓ,

ધોરણ 9 થી 11માં તમે ઉબુન્ટુ ઓપરેટિંગ સિસ્ટમ, રાઇટર, કેલ્સી, ઇમ્પ્રેસ જેવા ઓપન ઑફિસ ઘટકો, બેઝ જેવા ડેટાબેઝ મેનેજમેન્ટ ટૂલ તથા HTMLમાં વેબપેજની રજૂઆત વિશે અભ્યાસ કર્યો. અલ્ગોરિધમની રચના અને સાદા સી પ્રોગ્રામિંગ વિશે પણ તમે જાણ્યું, જેને પ્રોસિજર આધારિત પ્રોગ્રામિંગ ભાષા કહેવામાં આવે છે. આ ઉપરાંત તમારી સમક્ષ એનિમેશન માટેના મલ્ટીમીડિયા ટૂલ્સને પણ રજૂ કરવામાં આવ્યું. ધોરણ 12ના આ પાઠ્યપુસ્તકમાં કમ્પોઝરનો ઉપયોગ કરી વેબપેજની રચના, ઇ-કૉમર્સ, ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમના પાયાના સિદ્ધાંતો, જાવા પ્રોગ્રામિંગ ભાષા તથા અન્ય ઉપયોગી ટૂલ્સ રજૂ કરવામાં આવ્યાં છે.

પ્રકરણ 1થી 3ની મદદથી તમે સરળ છતાં કાર્યક્ષમ વેબસાઇટની રચના કરવાનો પ્રયત્ન કરી શકશો. આ પ્રકરણોમાં HTML ફૉર્મ, સ્ટાઇલ શીટ, જાવાસ્ક્રિપ્ટ અને વેબસાઇટની રચના માટે કમ્પોઝર નામના ટૂલની ચર્ચા કરવામાં આવી છે. પ્રકરણ 4 અને 5માં ઇ-કૉમર્સ, એમ-કૉમર્સ અને એલ-કૉમર્સ વિશે જાણકારી આપવામાં આવી છે. આજકાલ સૉફ્ટવેરના વિકાસમાં મહદંશે ઑબ્જેક્ટ ઓરિએન્ટેડ પદ્ધતિનો ઉપયોગ કરવામાં આવે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમનો અભ્યાસ તમે પ્રકરણ 6માં કરશો. વેબ-વિકાસમાં અગ્રસર ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાનો પરિચય પ્રકરણ 7માં આપવામાં આવ્યો છે. ક્લાસ અને ઑબ્જેક્ટ, ઍરે અને સ્ટ્રિંગ, અપવાદરૂપ પરિસ્થિતિ અને ફાઇલનું વ્યવસ્થાપન જેવી સુવિધાઓ પ્રકરણ 8 થી 11માં સમાવવામાં આવી છે. એક અન્ય ઓપનસૉર્સ શબ્દપ્રક્રિયક LaTex કે જે પુસ્તકના લેખકો, પ્રકાશકો અને વૈજ્ઞાનિકોમાં સવિશેષ પ્રચલિત છે તેનો પરિચય પ્રકરણ 12માં આપવામાં આવ્યો છે. અંતિમ પ્રકરણમાં કમ્પ્યૂટર સાથે કાર્ય કરી શકાય તેવાં કેટલાંક અન્ય ઉપયોગી ઓપનસૉર્સ ટૂલ્સ જેવાં કે, આર્કાઇવ મૅનેજર, VLC મીડિયા પ્લેયર, ગૂગલ મૅપ્સ, આંકડાશાસ્ત્રીય ટૂલ R, સ્કાઇપ અને રેશનલ પ્લાન વગેરેને તેમનાં સામાના કાર્યોની ઉપયોગિતા સાથે સંક્ષેપમાં ચર્ચવામાં આવ્યાં છે.

તમે આ પુસ્તકનો કાળજીપૂર્વક અભ્યાસ કરી પ્રાયોગિક કાર્યમાં પ્રવૃત્ત થાઓ તથા વેબસાઇટની અને જાવાના પ્રોગ્રામની રચનામાં સંપૂર્ણ આત્મવિશ્વાસ કેળવી શકો એ અપેક્ષિત છે.

•

કમ્પોઝરનો ઉપયોગ કરી HTML ફૉર્મની રચના

1

ઇન્ટરનેટનો ઉપયોગ વધવાથી અનેક પ્રવૃત્તિઓ ઑનલાઇન બની છે. આપણે પોતાની કે કોઈ વસ્તુની માહિતી રજૂ કરવા માટે વેબપેજનો ઉપયોગ કરતા હોઈએ છીએ. વેબસાઇટનો મુલાકાતી પોતાની વિગતો દાખલ કરવા માટે HTML ફૉર્મની મદદ લે છે. ફૉર્મ મુલાકાતીની વિગતો દાખલ કરવા વિશે વધુ સંવાદન (interactivity) અને નિયંત્રણ પૂરાં પાડે છે. ઉદાહરણ તરીકે, જો તમે કોઈ વેબસાઇટ પર મેઇલ-એકાઉન્ટ ખોલવા કે નોંધણી કરાવવા ઇચ્છતા હો, તો તે માટે તમારે ફૉર્મમાં તમારી અંગત વિગતો ભરવાની જરૂર પડે છે. ઉપયોગકર્તાના મેઇલ-એકાઉન્ટને ગોઠવવા માટે આ વિગતોનો ઉપયોગ કરવામાં આવે છે. ત્યાર પછી વિનિયોગ (application) દ્વારા આ વિગતોનો સંગ્રહ કરવામાં આવે છે અને વેબસાઇટ પર નોંધાયેલા ઉપયોગકર્તાઓ વિશેની વિગતો મેળવવા માટે તેનો ઉપયોગ કરવામાં આવે છે.

ફૉર્મ એક સંગ્રાહક (container) છે, જેનો ઉપયોગ ઉપયોગકર્તા પાસેથી જુદા-જુદા પ્રકારની વિગતો મેળવવા માટે કરવામાં આવે છે. HTML ફૉર્મમાં લેબલ (label), ચેકબૉક્સ (checkbox), લખાણ ઉમેરવા માટેનું ક્ષેત્ર (text input field), રેડિયો-બટન (radio button), સબમિટ બટન (submit button), રિસેટ બટન (reset button) અને આવા અનેક ઘટકો ઉમેરવામાં આવે છે. આ ઘટકોનો ઉપયોગ ફૉર્મમાં વિગતો દાખલ કરવા તથા વિગતોની યથાર્થતા ચકાસવા માટે પણ કરવામાં આવે છે. HTML ટેગનો ઉપયોગ કરી આપણે એક સરળ ફૉર્મની રચના કરીશું, પરંતુ તે પહેલાં HTML ફૉર્મની રચના કરવા માટે ઉપયોગી એવા ઘટકોની ચર્ચા કરીએ. આ ઘટકો નીચેના વિભાગમાં વર્ણવ્યા છે :

- ફૉર્મ (Form)
- ઇનપુટ (Input)
- ટેક્સ્ટ-એરિયા (Text Area)
- સિલેક્ટ અને ઑપ્શન (Select and Option)

ફૉર્મના ઘટકો (Form Elements)

HTML ફૉર્મની રચના કરવા માટે form ઘટકનો ઉપયોગ કરવામાં આવે છે. ફૉર્મમાં સમાવિષ્ટ તમામ ઘટકોના સંગ્રાહક તરીકે તે કાર્ય કરે છે. આ ઘટકને અમલમાં મૂકવા માટે <form>... </form> ટેગનો ઉપયોગ કરવામાં આવે છે. નીચે form ઘટકનું ઉદાહરણ દર્શાવ્યું છે.

```
<form action="register.html" method="post">
.
.
નિવેશ-ઘટકો (input elements)
.
.
</form>
```

form ઘટક action અને method નામની બે લાક્ષણિકતાઓ (attributes)નો ઉપયોગ કરે છે. જ્યારે ફૉર્મ સબમિટ કરવામાં આવે, ત્યારે તેની વિગતો ક્યા સ્થાને મોકલવી છે, તે સ્પષ્ટ કરવા માટે action લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે. કિંમત તરીકે તે ફાઇલનું નામ સ્વીકારે છે. ફૉર્મમાં વિગતો ઉમેર્યા પછી જ્યારે ઉપયોગકર્તા સબમિટ બટન પર ક્લિક કરે છે, ત્યારે આ ફાઇલ ખોલવામાં આવે છે.

વિગતો મોકલતી વખતે method લાક્ષણિકતાના ઉપયોગ દ્વારા HTTP પદ્ધતિ સ્પષ્ટ કરવામાં આવે છે. તે બે કિંમત સ્વીકારી શકે છે : GET અને POST. GET પદ્ધતિ ફૉર્મમાંથી વિગતો મેળવી, તેને URLના અંતમાં ઉમેરી સર્વરને મોકલી આપે છે. આ પદ્ધતિ એક સમયે માત્ર મર્યાદિત માહિતી મોકલવાની અનુમતિ આપે છે. POST પદ્ધતિમાં વિગતોને HTTP ટ્રાન્ઝેક્શન દ્વારા બ્લૉક સ્વરૂપે મોકલવામાં આવે છે. વિગતોને વિનંતીમાં સમાવીને મોકલવામાં આવે છે. આ પદ્ધતિમાં વિગતોની લંબાઈ બાબતે કોઈ મર્યાદા નથી. method લાક્ષણિકતાની પૂર્વનિર્ધારિત કિંમત GET છે.

નિવેશ-ઘટક (Input Element)

ફૉર્મમાં રેડિયો-બટન, ટૅક્સ્ટબૉક્સ અને ચેકબૉક્સ જેવાં વિવિધ ક્ષેત્ર (field) ઉમેરવા માટે નિવેશ-ઘટકોનો ઉપયોગ કરવામાં આવે છે. <input> ....</input> અથવા માત્ર <input> ટેગનો ઉપયોગ કરી આ ઘટક અમલમાં મૂકી શકાય છે. <input> ટેગનો ઉપયોગ type, name અને value જેવી લાક્ષણિકતાઓ સાથે કરવામાં આવે છે.

ફૉર્મમાં ઉમેરવામાં આવનાર ફિલ્ડની સ્પષ્ટતા input ઘટક સાથે type લાક્ષણિકતા દ્વારા કરવામાં આવે છે. ફૉર્મમાં આવેલા ફિલ્ડને નામ આપવા માટે name લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે. ફૉર્મમાં ફિલ્ડને પૂર્વનિર્ધારિત કિંમત સાથે દર્શાવવા માટે value લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે. કોષ્ટક 1.1માં આ લાક્ષણિકતાઓને તેના ઉપયોગ સાથે દર્શાવી છે.

Type	સમજૂતી	Example
Radio	ફૉર્મમાં રેડિયો-બટનની રચના કરે છે. રેડિયો-બટનના જૂથમાંથી એક સમયે માત્ર કોઈ પણ એક રેડિયો-બટન પસંદ કરી શકાય છે. સામાન્ય રીતે વિકલ્પોનાં જૂથમાંથી એક વિકલ્પ પસંદ કરવા માટે રેડિયો-બટનનો ઉપયોગ કરવામાં આવે છે.	<INPUT TYPE = "radio" NAME = "var" VALUE = "txt">
Checkbox	ફૉર્મમાં ચેકબૉક્સની રચના કરે છે. એકસાથે એકથી વધુ ચેકબૉક્સની પસંદગી શક્ય છે. આપેલ વિકલ્પોનાં જૂથમાં એકથી વધુ વિકલ્પ પસંદ કરવા માટે ચેકબૉક્સનો ઉપયોગ કરવામાં આવે છે.	<INPUT TYPE="checkbox" NAME = "var" VALUE ="txt" >
Text	ફૉર્મમાં લખાણ ઉમેરવા માટે ટૅક્સ્ટ-ફિલ્ડની રચના કરે છે. ટૅક્સ્ટ ફિલ્ડમાં ઉપયોગકર્તા ઇચ્છિત કોઈ પણ વિગતો ઉમેરી શકે છે.	<INPUT TYPE="text" NAME = "var" VALUE = "txt" >
Password	ફૉર્મમાં પાસવર્ડ ફિલ્ડની રચના કરે છે. આ ફિલ્ડ ટૅક્સ્ટ-ફિલ્ડ સમાન છે, પરંતુ તેમાં ઉમેરવામાં આવેલ અક્ષરો ઉપયોગકર્તા સમક્ષ દર્શાવવામાં આવતા નથી. તેને બદલે અક્ષરોનું અવાચ્ય સ્વરૂપમાં રૂપાંતરણ કરવામાં આવે છે.	<INPUT TYPE="password" NAME = "var" >
Submit	ફૉર્મમાં સબમિટ બટનની રચના કરે છે. સબમિટ બટન પર ક્લિક કરવામાં આવે, ત્યારે ફૉર્મમાં દાખલ કરવામાં આવેલ વિગતો ફૉર્મ ઘટક સાથે ઉપયોગમાં લેવામાં આવેલ action લાક્ષણિકતાને આપવામાં આવેલ ફાઈલ તરફ મોકલવામાં આવે છે.	<INPUT TYPE="submit" VALUE = "label" >
Reset	ફૉર્મમાં રિસેટ બટનની રચના કરે છે. રિસેટ બટન પર ક્લિક કરવાથી ફૉર્મમાં દાખલ કરવામાં આવેલ વિગતો દૂર કરી ફૉર્મને પૂર્વનિર્ધારિત કિંમતો સાથે દર્શાવવામાં આવે છે.	<INPUT TYPE = "reset" VALUE = "label" >

કોષ્ટક 1.1: input ટેગ સાથે ઉપયોગમાં લેવામાં આવતા type લાક્ષણિકતાની કિંમતો

ટેક્સ્ટ-એરિયા ઘટક (Textarea Element)

ટેક્સ્ટ-એરિયા ઘટક એકથી વધુ લીટીનું લખાણ નિવેશ કરવાની સુવિધા પૂરી પાડે છે. આ ઘટકનો અમલ કરવા માટે <textarea>…</textarea> ટેગનો ઉપયોગ કરવામાં આવે છે. તેમાં અમર્યાદિત અક્ષરો ઉમેરી શકાય છે. નોંધ (Comment), અહેવાલ (report) કે લાંબી સમજૂતી (long description) ઉમેરવા માટે તેનો ઉપયોગ કરી શકાય છે. ટેક્સ્ટ-એરિયા ઘટકનું કદ rows અને cols લાક્ષણિકતાઓ દ્વારા બદલી શકાય છે. ટેક્સ્ટ-એરિયામાં ઉપર કે નીચે ગયા (scroll) વિના જોઈ શકાય તેવી હરોળોની સંખ્યા નક્કી કરવા માટે rows લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે. ટેક્સ્ટ-એરિયામાં ડાબી કે જમણી બાજુ ગયા (scroll) વિના જોઈ શકાય તેવા સ્તંભની સંખ્યા નક્કી કરવા માટે cols લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે. ફોર્મમાં ટેક્સ્ટ-એરિયા ઉમેરવાની રીત નીચેનાં ઉદાહરણમાં દર્શાવી છે.

```
<form method="post" action="comment.html">
Input your comments: <br /> <textarea name="comments" rows="4" cols="20">
...Your comments here...
</textarea>
</form>
```

સિલેક્ટ અને ઑપ્શન ઘટક (Select and Option Element)

ફોર્મમાં ડ્રૉપડાઉન યાદી (dropdown list) કે મેનુની રચના કરવા માટે select ઘટકનો ઉપયોગ કરવામાં આવે છે. મેનુમાં દર્શાવવામાં આવનાર કિંમતો ઉમેરવા માટે option ઘટકનો ઉપયોગ કરવામાં આવે છે. ડ્રૉપડાઉન મેનુની રચના કરવા માટે <select>....</select> ટેગનો ઉપયોગ કરવામાં આવે છે. મેનુના ઘટકોની રચના કરવા માટે <option>... </option> ટેગનો ઉપયોગ કરવામાં આવે છે. નીચેનું ઉદાહરણ select અને option ઘટકનો ઉપયોગ દર્શાવે છે.

```
<select>
        <option value="Ahmedabad" >Ahmedabad</option>
        <option value="Rajkot" >Rajkot</option>
        <option value="Surat" >Surat </option>
</select>
```

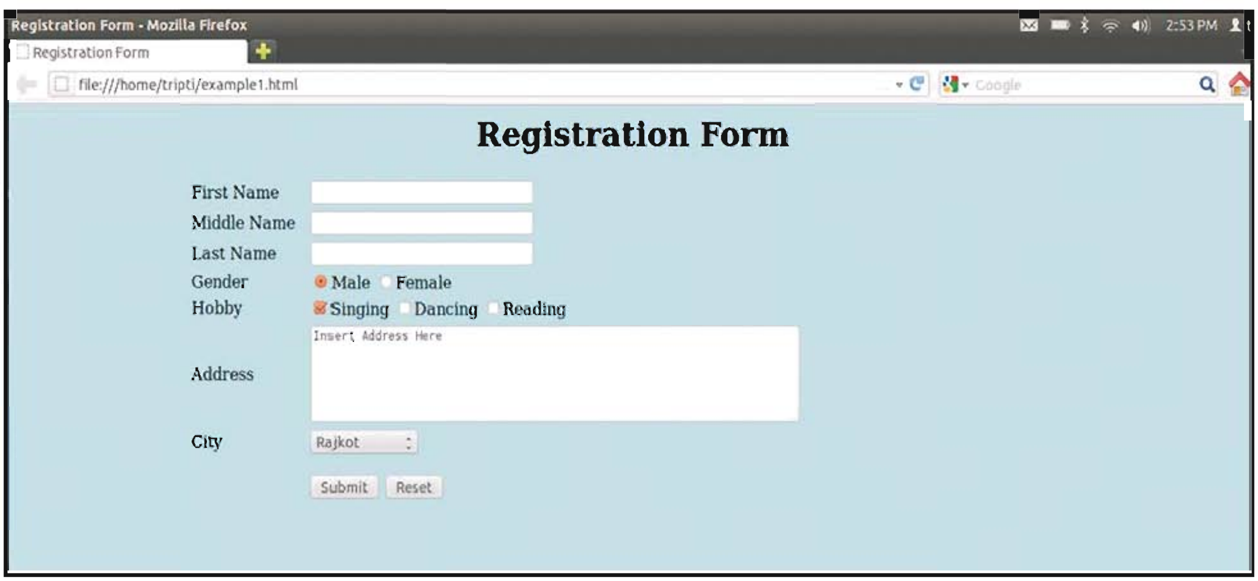
હવે આપણે, અત્યાર સુધીમાં શીખેલા ઘટકોનો ઉપયોગ કરી નોંધણી માટેના એક નમૂનારૂપ ફોર્મની રચના કરીએ. કોડ-લિસ્ટિંગ 1.1માં ફોર્મ બનાવવા માટેનો HTML કોડ દર્શાવ્યો છે. કોડનું પરિણામ આકૃતિ 1.1માં દર્શાવવામાં આવ્યું છે.

```
<HTML>
<HEAD>
        <TITLE>Registration Form</TITLE>
</HEAD>
<BODY bgcolor="lightblue">
<h1> <center>Registration Form</center></h1>
<FORM name="frmRegistration" action="form.html">
        <center>
        <TABLE BORDER="0">
<TR>
        <TD width="12%">First Name</TD>
        <TD width="1%"> </TD>
```

```
		<TD> <INPUT type="textbox" name="txtFirstName"></TD>
	</TR>
	<TR>
		<TD>Middle Name</TD>
		<TD> </TD>
		<TD><INPUT type="text box" name="txtMiddleName"></TD>
	</TR>
	<TR>
		<TD>Last Name</TD>
		<TD> </TD>
		<TD> <INPUT type="text box" name="txtLastName"></TD>
	</TR>
	<TR>
		<TD>Gender</TD>
		<TD> </TD>
		<TD>
		<INPUT type="radio" name="Gender" value="male" CHECKED>Male
		<INPUT type="radio" name="Gender" value="female" >Female
		</TD>
	</TR>
	<TR>
		<TD>Hobby</TD>
		<TD> </TD>
		<TD>
		<INPUT type="checkbox" name="chkSinging" value="Sing" CHECKED>Singing
		<INPUT type="checkbox" name="chkDancing" value="Dance">Dancing
		<INPUT type="checkbox" name="chkReading" value="Read">Reading
		</TD>
	</TR>
	<TR>
		<TD>Address</TD>
		<TD> </TD>
		<TD>
		<Textarea name="txtAddress" rows="5" cols="70">Insert Address Here</Textarea>
		</TD>
	</TR>
	<TR>
		<TD>City</TD>
		<TD> </TD>
		<TD>
```

```
                <Select Name="cmbCity">
                <Option >Ahmedabad</Option>
                <Option >Baroda</Option>
                <Option selected>Rajkot</Option>
                <Option >Surat</Option>
                </Select>
                </TD>
        </TR>
        <TR>
                <TD> </TD>
                <TD> </TD>
                <TD> <INPUT type="submit" name="cmdSubmit" value="Submit">
                <INPUT  type="reset"  name="cmdReset"  value="Reset">
                </TD>
        </TR>
        </TABLE>
        </center>
</FORM>
</BODY>
</HTML>
```

કોડલિસ્ટિંગ 1.1 : નોંધણી માટેનું નમૂનારૂપ ફૉર્મ બનાવવા માટેનો HTML કોડ



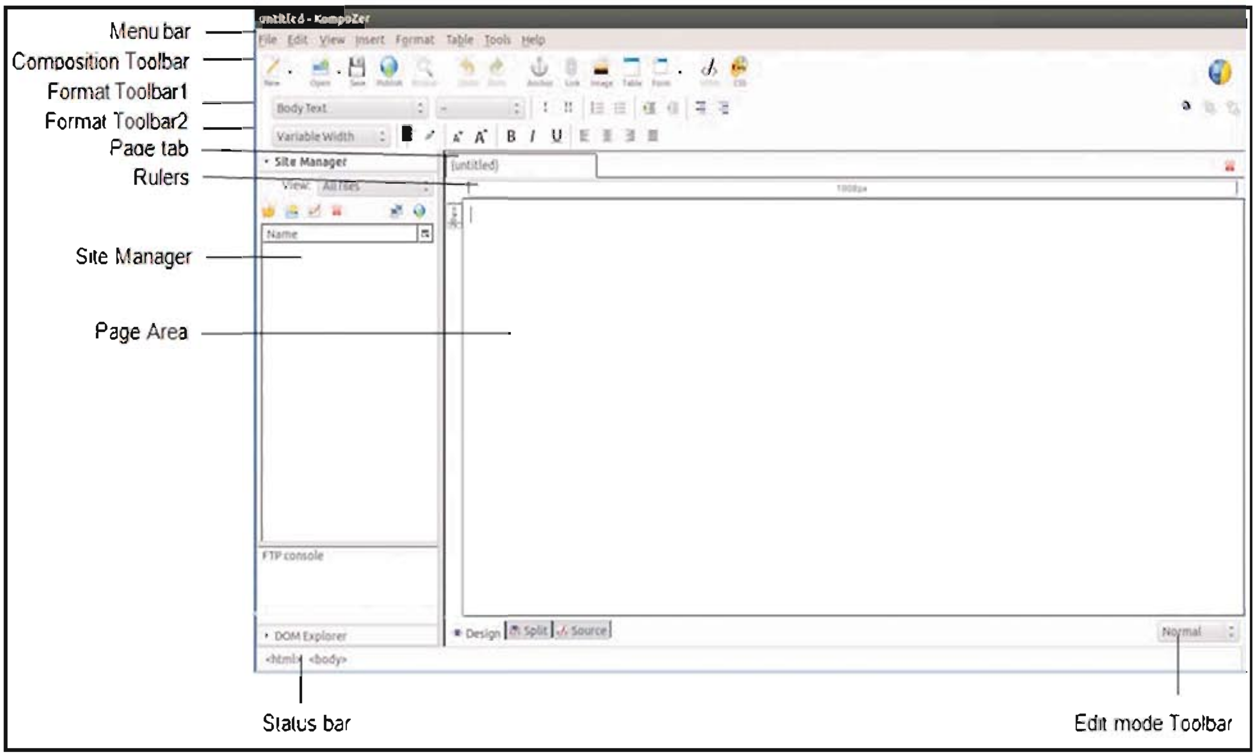
આકૃતિ 1.1 : વેબબ્રાઉઝરમાં દર્શાવેલું નોંધણી માટેનું ફૉર્મ

તમે જોઈ શકો છો કે HTML ટેગનો ઉપયોગ કરી ફૉર્મની રચના કરવી તે એક કંટાળાજનક પ્રક્રિયા છે. IDE (Integrated Development Environment)નો ઉપયોગ કરવો તે આ માટેની એક સરળ રીત છે. IDE એક એવો સૉફ્ટવેર વિનિયોગ છે, જે પ્રોગ્રામરને સૉફ્ટવેરનો વિકાસ કરવા માટેની સંપૂર્ણ સુવિધા પૂરી પાડે છે. તે GUI (Graphical User Interface), ટૅક્સ્ટ કે કોડ-એડિટર (text/code editor), કંપાઇલર કે ઇન્ટરપ્રિટર (compiler/interpreter) અને ડીબગર (debugger) પૂરા પાડે છે. કમ્પોઝર (KompoZer); એક્લિપ્સ (Eclipse), જેબિલ્ડર (JBuilder) અને નેટબિન્સ (Netbeans) એ કેટલાંક ઓપનસોર્સ IDEનાં ઉદાહરણ છે. હવે, વેબપેજની રચના કરવા કમ્પોઝરના ઉપયોગની ચર્ચા કરીએ.

કમ્પોઝરનો પરિચય (Introduction to KompoZer)

કમ્પોઝર એ વેબવિકાસ માટેનું એક નિઃશુલ્ક અને ઓપનસોર્સ IDE છે. તે http://www.KompoZer.net પરથી ડાઉનલોડ કરી શકાય છે. તે WYSIWYG "What You See Is What You Get" તરીકે ઓળખાતાં માધ્યમ સાથે વેબપેજ એડિટર પૂરું પાડે છે. તે એક સંપૂર્ણ વેબ-ઑથરિંગ સિસ્ટમ (Web Authoring System) છે, જે વેબપેજના વિકાસ અને વેબફાઈલના વહીવટનું સંચાલન કરે છે. કમ્પોઝર દ્વારા વેબપેજની રચના કરવી એ ઝડપી અને સરળ કાર્ય છે. તદુપરાંત, ઉપયોગકર્તા સૉર્સકોડનો ઉપયોગ કરીને તથા તેમાં સુધારા કરીને વેબપેજમાં ફેરફારો પણ કરી શકે છે. કમ્પોઝરમાં સાઈટ-મૅનેજર (Site-Manager)નો સમાવેશ કરવામાં આવ્યો છે, જે સ્થાનિક (local) તથા દૂરસ્થિત (remote) સર્વર એમ બંને સ્થાને રહેલ ફાઈલોનો ઝડપી ઉપયોગ પૂરો પાડે છે. કમ્પોઝરનો ઉપયોગ કરી વેબપેજ તથા તેને સંલગ્ન ફાઈલોને દૂરસ્થિત સર્વર પર અપલોડ કરી શકાય છે. તે કેસ્કેડિંગ સ્ટાઈલશીટ (Cascading Style Sheet) દ્વારા 'સ્ટાઈલ' (Style)ને પણ સમર્થન પૂરું પાડે છે. CSS વિશે આપણે હવે પછીના પ્રકરણમાં અભ્યાસ કરીશું.

કમ્પોઝરની મદદથી ફૉર્મની રચના વિશેનો અભ્યાસ કરતાં પહેલાં, સૌપ્રથમ કમ્પોઝરના ઈન્ટરફેસને સમજીએ. કમ્પોઝરને તેના આઈકનની મદદથી શરૂ કરો. જુદાં જુદાં ટૂલબાર અને સ્ટેટસબાર (જો દેખાતાં ન હોય તો) દર્શાવવા માટે **View → Show/Hide** વિકલ્પ પર ક્લિક કરો. યાદીમાં આપેલ Composition Toolbar, Format Toolbar1, Format Toolbar2, Edit Mode Toolbar અને Status bar એ તમામ વિકલ્પોને પસંદ કરો. Site-Manager અને Rulers વિકલ્પને પણ પસંદ કરો. ટૂલબાર પસંદ કર્યા પછીની વિન્ડો આકૃતિ 1.2માં દર્શાવી છે.



આકૃતિ 1.2 : કમ્પોઝર ઈન્ટરફેસ

આકૃતિ 1.2માં File, Edit, View, Insert, Format, Table, Tools અને Help જેવા વિકલ્પો ધરાવતો મેનુબાર વિન્ડોની ઉપરના ભાગમાં જોઈ શકાય છે. મેનુબારની નીચે ત્રણ ટૂલબાર આવેલા છે : Composition, Format Toolbar1 અને Format Toolbar2.

નવી ફાઈલની રચના કરવા, ફાઈલ ખોલવા, ફાઈલનો સંગ્રહ કરવા તથા વેબસાઈટને પ્રકાશિત (publish) કરવા માટે કમ્પોઝિશન ટૂલબારનો ઉપયોગ કરવામાં આવે છે. લખાણની ગોઠવણી કરવા, નિશાની (bullet) અને અનુક્રમ (Numbering) ઉમેરવા તથા ગોઠવણીને લગતાં અન્ય કાર્ય કરવા માટે ફૉર્મેટ ટૂલબાર-1 અને ફૉર્મેટ ટૂલબાર-2નો ઉપયોગ કરવામાં આવે છે.

વિન્ડોની મધ્યમાં બે વિભાગ જોઈ શકાય છે : સાઇટ-મૅનેજર અને ખાલી વેબપેજ. સાઇટ-મૅનેજર એ સાઇટમાં અથવા એકથી વધુ સાઇટ વચ્ચેના નેવિગેશન માટે ઉપયોગમાં લઈ શકાય તેવું એક સક્ષમ સાધન છે. Close બટન પર ક્લિક કરીને અથવા તો F9 કી દબાવીને સાઇટ-મૅનેજરના વિભાગને બંધ કરી શકાય છે. Pagepane એક ખાલી અને નામ વગરનું (Untitled) વેબપેજ દર્શાવે છે. વિન્ડોની નીચે આવેલા જમણી બાજુના ભાગમાં ત્રણ મોડ સાથેનો Edit mode ટૂલબાર આપવામાં આવે છે : Normal, HTML Tags અને Preview. આ ત્રણે મોડ સુધારા માટેની સુવિધા પૂરી પાડે છે.

પ્રિવ્યૂ મોડ બ્રાઉઝરમાં જોઈ શકાય તે પ્રકારનો પાનાનો દેખાવ દર્શાવે છે. તફાવત એ છે કે પ્રિવ્યૂ મોડમાં સ્ક્રિપ્ટનો અમલ કરવામાં આવતો નથી અને તેથી તેની અસર દર્શાવવામાં આવતી નથી. પ્રિવ્યૂ મોડમાં લિંકનો અમલ પણ કરી શકાતો નથી.

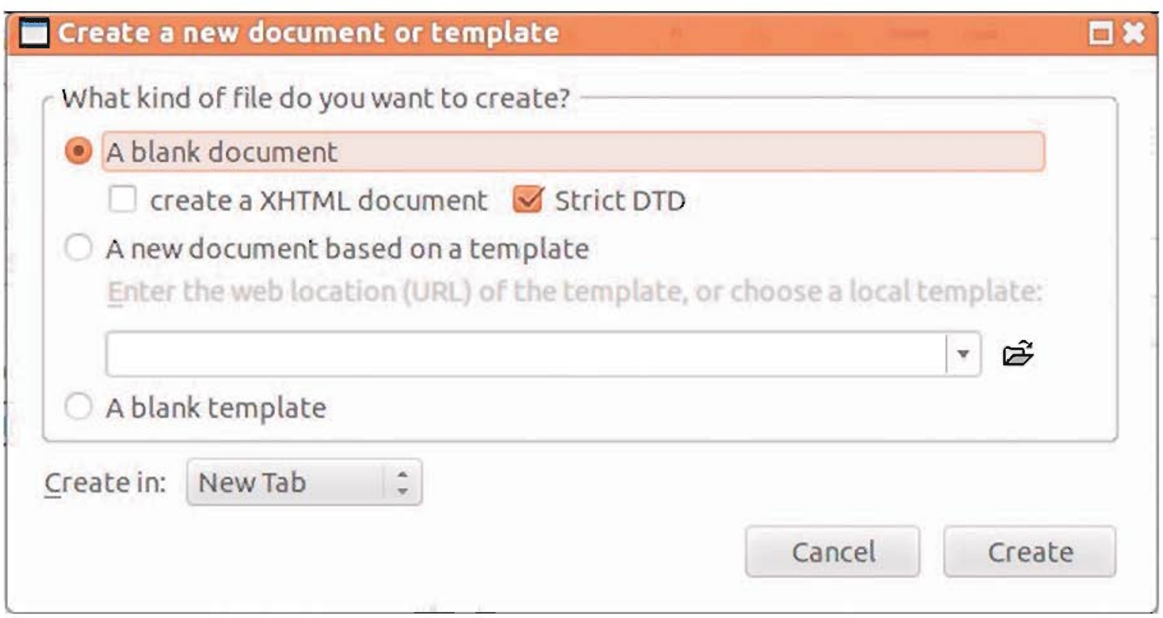
નૉર્મલ વ્યૂ એ પ્રિવ્યૂ મોડ જેવો જ છે. આ મોડમાં કોષ્ટકની સીમારેખા દર્શાવવામાં આવે છે. જેઓ HTMLથી પરિચિત છે તેવા ઉપયોગકર્તાને HTML ટેગ વ્યૂ મદદરૂપ બને છે. તમામ ટેગની શરૂઆત દર્શાવવા માટે પીળા રંગના નિશાનનો ઉપયોગ કરવામાં આવે છે. આ નિશાન પર ક્લિક કરવાથી ઘટકના સમગ્ર વિભાગને પસંદ કરી હાઇલાઇટ (Highlight) કરી શકાય છે.

પેજપેનની ડાબી બાજુ Design, Split અને Source વિભાગો (tabs) દર્શાવવામાં આવ્યા છે. વેબપેજની રૂપરેખા તૈયાર કરવા માટે Design વિભાગનો ઉપયોગ કરવામાં આવે છે. વર્તમાન ઘટકના HTML સૉર્સને Split વિભાગમાં દર્શાવવામાં આવે છે. HTML કોડની તમામ વિગતો Source વિભાગમાં દર્શાવવામાં આવે છે જે સૉર્સકોડ સુધારવામાં મદદરૂપ બને છે.

વિન્ડોના નીચેના ભાગમાં સ્ટેટસબાર જોઈ શકાય છે. જ્યારે પાનાંમાં કોઈ પણ વસ્તુ પર ક્લિક કરવામાં આવે છે, ત્યારે તેનું માળખું સ્ટેટસબારમાં દર્શાવવામાં આવે છે. જો આપણે કોઈ પણ ટૂલબારની ગોઠવણી બદલવા માંગતા હોઈએ, તો તે ટૂલબાર પર રાઇટ ક્લિક કરી Customize Toolbar વિકલ્પ પર ક્લિક કરી શકાય. ત્યાર પછી તે ટૂલબારને આપણી પસંદગી પ્રમાણે ગોઠવી શકાશે.

નવી ફાઇલ બનાવવી (Create a New File)

નવી ફાઇલ બનાવવા માટે કમ્પોઝર શરૂ કરો. મેનુબારમાં **File → New** વિકલ્પ પર ક્લિક કરો. આમ કરવાથી આકૃતિ 1.3માં દર્શાવ્યા મુજબ "Create a new document or template" શીર્ષક સાથે એક ડાયલોગબૉક્સ ખૂલશે. ડાયલોગબૉક્સમાં આપેલા વિકલ્પો પૈકી "A blank document" વિકલ્પ પસંદ કરો. ડાયલોગબૉક્સના નીચેના ભાગમાં "Create in" લેબલ જોઈ શકાશે. તેની બાજુમાં આવેલા ડ્રૉપડાઉન મેનુમાંથી New Tab વિકલ્પ પસંદ કરો. તે નવા વિભાગ (Tab)માં વેબપેજની રચના કરશે. Create બટન પર ક્લિક કરો.



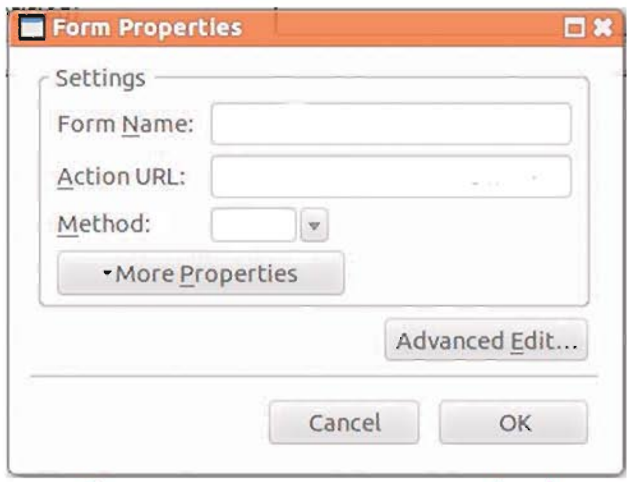
આકૃતિ 1.3 : નવી ફાઇલની રચના

હયાત ફાઈલ ખોલવી (Open an Existing File)

હયાત ફાઈલ ખોલવા માટે કમ્પોઝિશન ટૂલબાર પર આવેલા આઈકન પર ક્લિક કરો. વૈકલ્પિક રીતે, **File → Open** વિકલ્પ પણ પસંદ કરી શકાય. જો હાલમાં જ ખોલવામાં આવી હોય, તો તેવી ફાઈલને **File → Recent Pages** વિકલ્પની મદદથી પણ ખોલી શકાય છે.

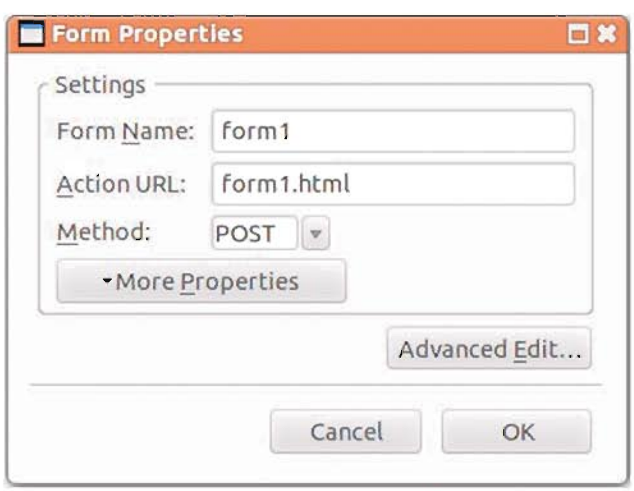
હવે, કમ્પોઝરની મદદથી ફોર્મની રચના કરતા શીખીએ. આપણે એક સરળ ફોર્મ બનાવીએ; જેમાં બે ઇનપુટ ફિલ્ડ હોય : નામ અને ઇ-મેઇલ સરનામું તથા સબમિટ બટન. ફોર્મની રચના કરવા માટે નીચેનાં પગલાંને અનુસરો :

- કમ્પોઝર શરૂ કરો. નવી ફાઈલ બનાવો.
- મેનુબારમાંથી **Insert → Form → Define Form** વિકલ્પ પસંદ કરો. વૈકલ્પિક રીતે, કમ્પોઝિશન ટૂલબારમાં આવેલ બટન પર ક્લિક કરી શકાય. આમ કરવાથી આકૃતિ 1.4માં આવેલ Form Properties ડાયલોગ-બોક્સ રજૂ કરવામાં આવશે. More Properties પર ક્લિક કરી અતિરિક્ત વિકલ્પો દર્શાવી શકાશે.



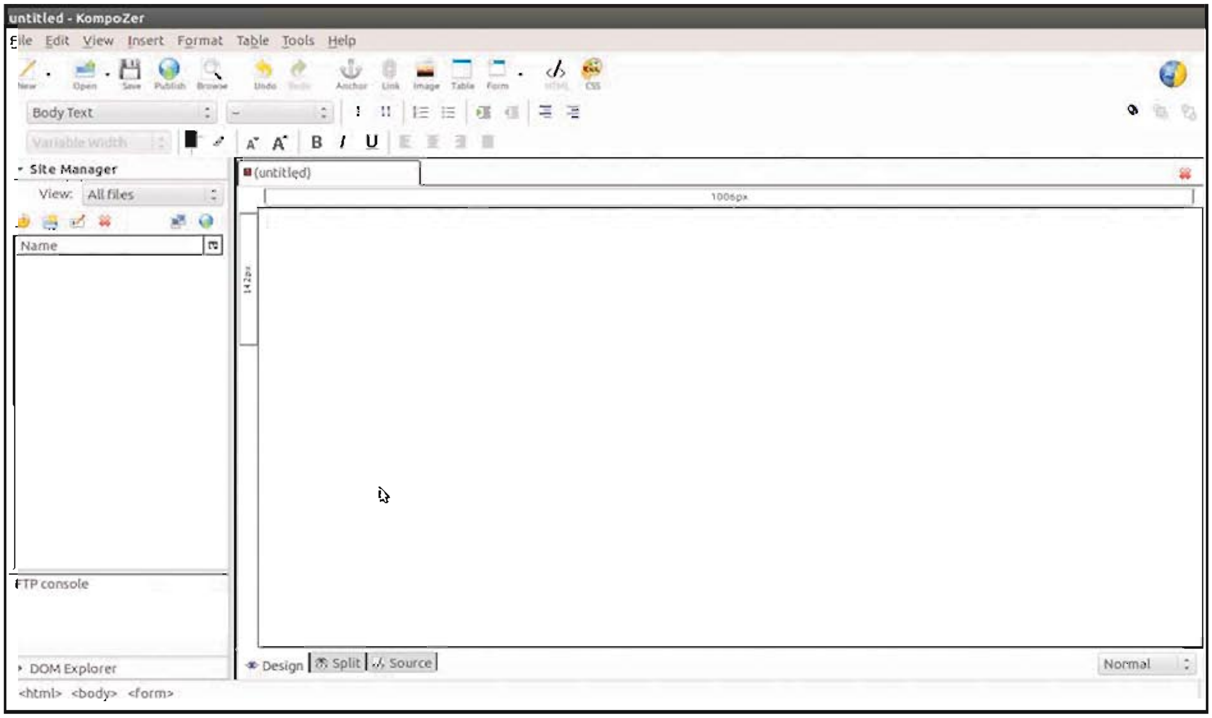
આકૃતિ 1.4 : Form Properties ડાયલોગબૉક્સ

- ફોર્મ માટે યોગ્ય નામ ઉમેરો. ફોર્મની વિગતો મોકલવાની હોય તે ફાઈલનું નામ Action URL વિકલ્પની કિંમત સ્વરૂપે ઉમેરો. Method ડ્રોપડાઉન મેનુમાંથી POST પદ્ધતિ પસંદ કરો. અને OK બટન પર ક્લિક કરો. Form Properties ડાયલોગબૉક્સમાં ઉમેરવામાં આવેલી વિગતો આકૃતિ 1.5માં દર્શાવી છે.



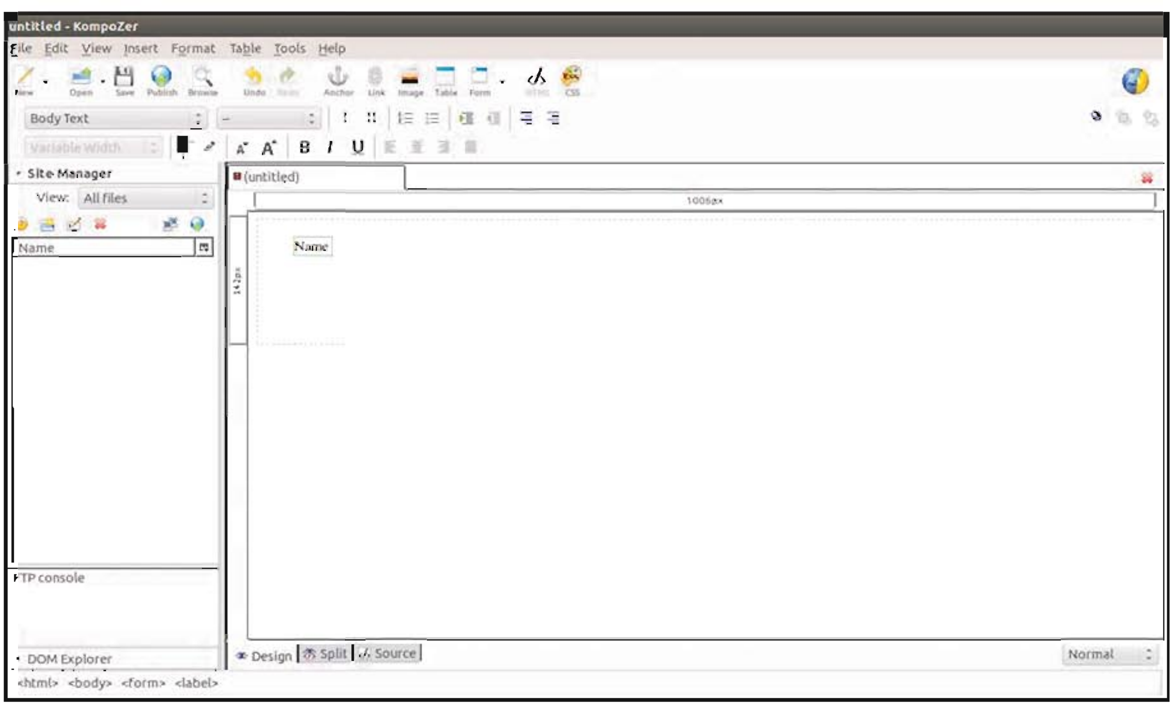
આકૃતિ 1.5 : Form Properties ડાયલોગબૉક્સમાં ઉમેરેલી વિગતો

- આકૃતિ 1.6માં દર્શાવ્યા મુજબ ફૉર્મ શીર્ષકરહિત (untitled) પાનામાં આછા ભૂરા રંગની સીમારેખા સાથે ઉમેરવામાં આવશે. સામાન્ય દેખાવમાં ફૉર્મની આસપાસ ટપકાંવાળું ભૂરું ચોકઠું દર્શાવવામાં આવશે. ટેક્સ્ટબૉક્સ, રેડિયો-બટન, ચેકબૉક્સ અને ડ્રૉપડાઉન બૉક્સ જેવા ફૉર્મના તમામ ઘટકો આ બૉક્સમાં મૂકવામાં આવશે. એકથી વધુ વાર એન્ટર કી દબાવી ફૉર્મમાં કાર્ય કરવા માટેની થોડી જગ્યા બનાવો.



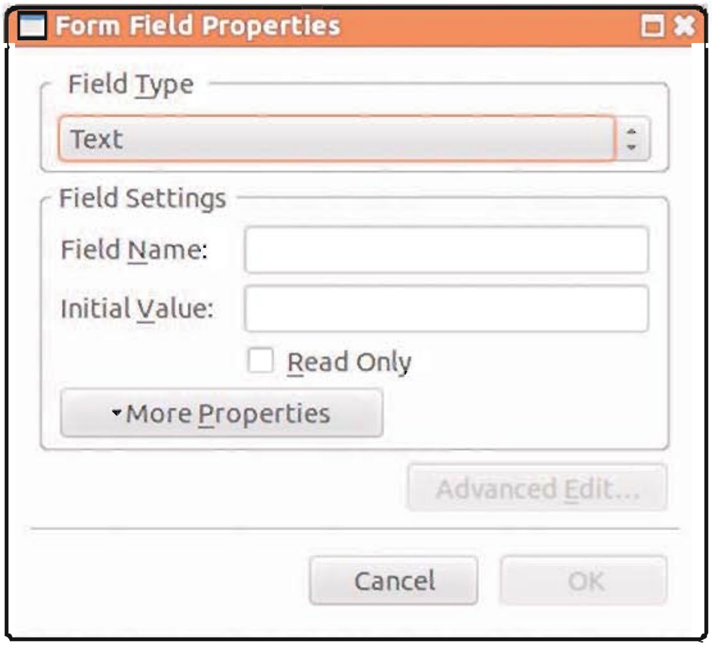
આકૃતિ 1.6 : ફૉર્મની આછા ભૂરા રંગની સીમારેખા

- સૌપ્રથમ આપણે name ક્ષેત્ર માટે લેબલ ઉમેરીશું. **Insert → Form → Define label** પસંદ કરો. ફૉર્મમાં જે સ્થાને લેબલ ઉમેરવું હોય, ત્યાં કર્સર ગોઠવો. આકૃતિ 1.7માં દર્શાવ્યા મુજબ લેબલમાં "Name" લખાણ ટાઇપ કરો. લેબલફિલ્ડમાંથી બહાર આવવા ફિલ્ડ સિવાયના અન્ય કોઈ પણ સ્થાને ક્લિક કરો.



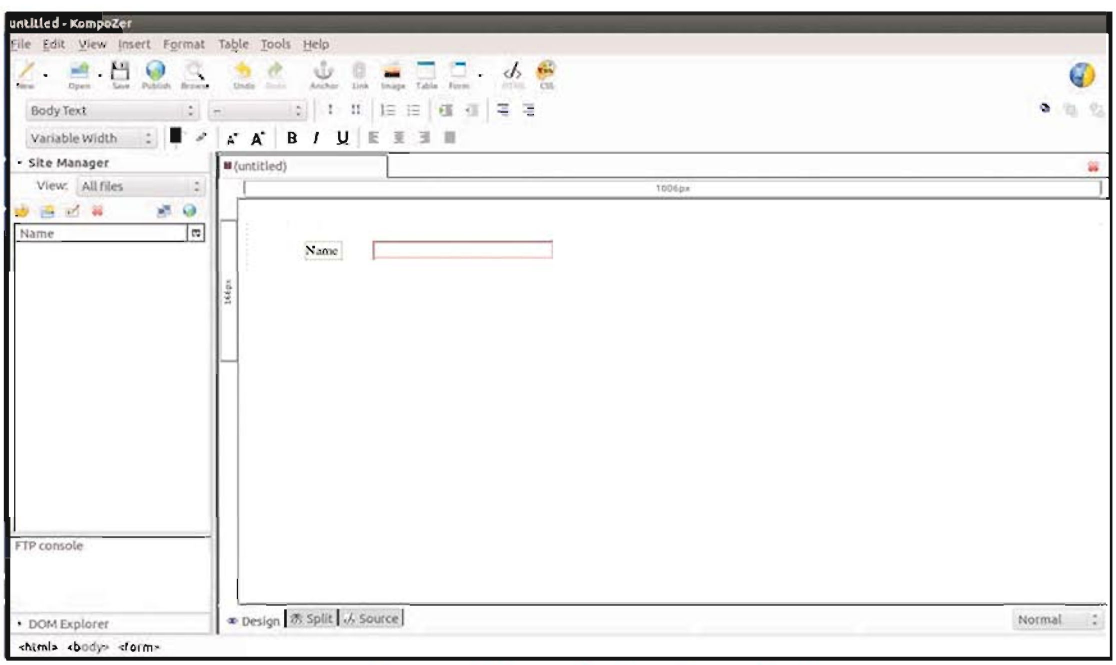
આકૃતિ 1.7 : ફૉર્મમાં ઉમેરવામાં આવેલ લેબલફિલ્ડ

- ફૉર્મમાં ઇનપુટ ટેક્સ્ટફિલ્ડ ઉમેરવા માટે **Insert → Form → Form Field** વિકલ્પ પસંદ કરો. આકૃતિ 1.8 Form Field Properties ડાયલોગબૉક્સ દર્શાવે છે. ડ્રૉપડાઉન મેનુ વિવિધ ઇનપુટ ફિલ્ડના પ્રકાર દર્શાવે છે, જેની ચર્ચા આપણે અગાઉ કરી ચૂક્યા છીએ. More Properties બટન પર ક્લિક કરો, તે ફિલ્ડના કદ અને મહત્તમ લંબાઈ જેવા કેટલાક અન્ય ગુણધર્મો દર્શાવે છે.



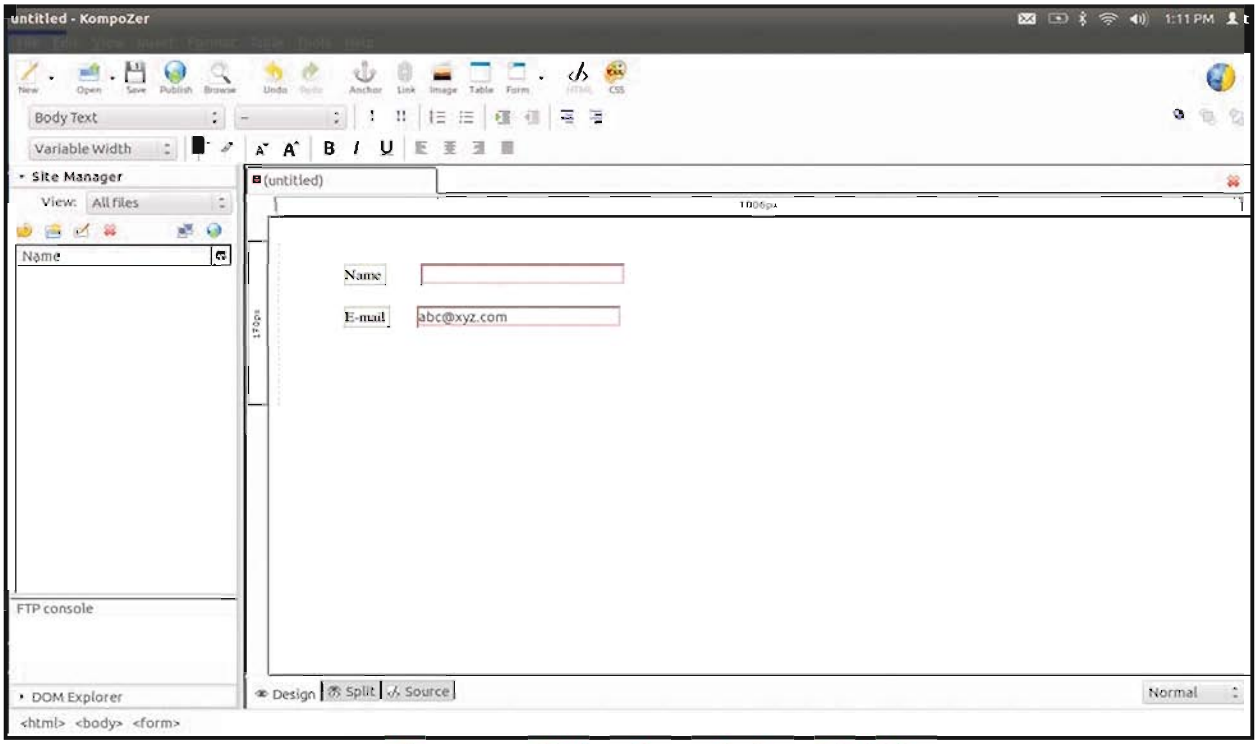
આકૃતિ 1.8 : ફૉર્મમાં ફિલ્ડ ઉમેરવું

- ડ્રૉપડાઉન મેનુમાંથી Text વિકલ્પ પસંદ કરો. Field Settings વિભાગમાં આવેલ Field Name ટેક્સ્ટબૉક્સમાં ફિલ્ડનું નામ ઉમેરો. અહીં આપણે ફિલ્ડનાં નામ તરીકે name ઉમેર્યું છે. જરૂરી વિગતો ઉમેરતાં પહેલાં ફિલ્ડમાં અન્ય કોઈ વિગત દર્શાવવા માટે Initial Value ટેક્સ્ટબૉક્સમાં થોડું લખાણ ઉમેરો. અહીં આપણે આ બૉક્સ ખાલી રાખ્યું છે. OK બટન પર ક્લિક કરો. ટેક્સ્ટ ઇનપુટ ફિલ્ડ ઉમેર્યા પછીનું ફૉર્મ આકૃતિ 1.9માં દર્શાવ્યું છે.



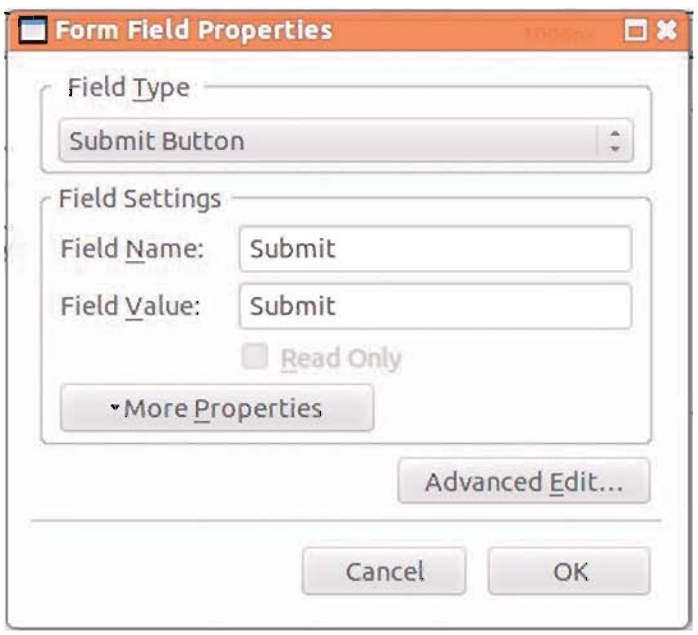
આકૃતિ 1.9 : લેબલ અને ટેક્સ્ટફિલ્ડ ઉમેરવાં

- હવે, આવી જ રીતે Name લેબલ ફિલ્ડની નીચે અન્ય લેબલ ફિલ્ડ E-mail ઉમેરો. Name ફિલ્ડ ઉમેર્યું તે જ રીતે E-mail માટે પણ ઇનપુટ ટેક્સ્ટફિલ્ડ ઉમેરો. અહીં Initial Value ટેક્સ્ટબૉક્સમાં abc@xyz.com લખાણ ઉમેરો. તે ઉપયોગકર્તાને ઇ-મેઇલ સરનામાંનું સ્વરૂપ સમજવામાં મદદરૂપ બનશે. બંને ટેક્સ્ટફિલ્ડ ઉમેર્યા પછીનું ફૉર્મ આકૃતિ 1.10માં દર્શાવ્યું છે.



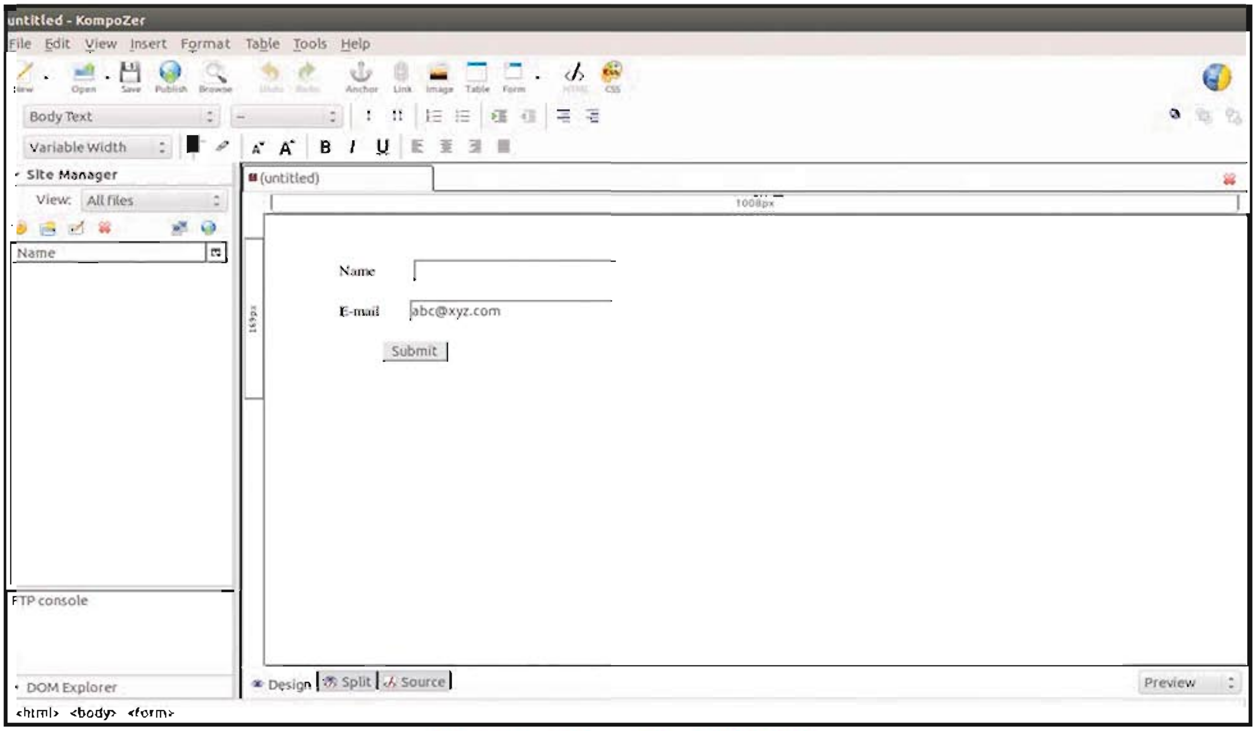
આકૃતિ 1.10 : બંને ટેક્સ્ટફિલ્ડ ઉમેર્યા પછીનું ફૉર્મ

- અંતમાં આપણે ફૉર્મમાં સબમિટ બટન ઉમેરીશું. **Insert → Form → Form Field** પર ક્લિક કરો. ડ્રૉપડાઉન મેનુમાંથી Submit Button વિકલ્પ પસંદ કરો. Field Name અને Field Value બંને ટેક્સ્ટબૉક્સમાં Submit લખાણ ટાઇપ કરો અને OK બટન પર ક્લિક કરો. સબમિટ બટન માટેના Form Field Properties ડાયલોગબૉક્સનો દેખાવ આકૃતિ 1.11માં દર્શાવ્યો છે.



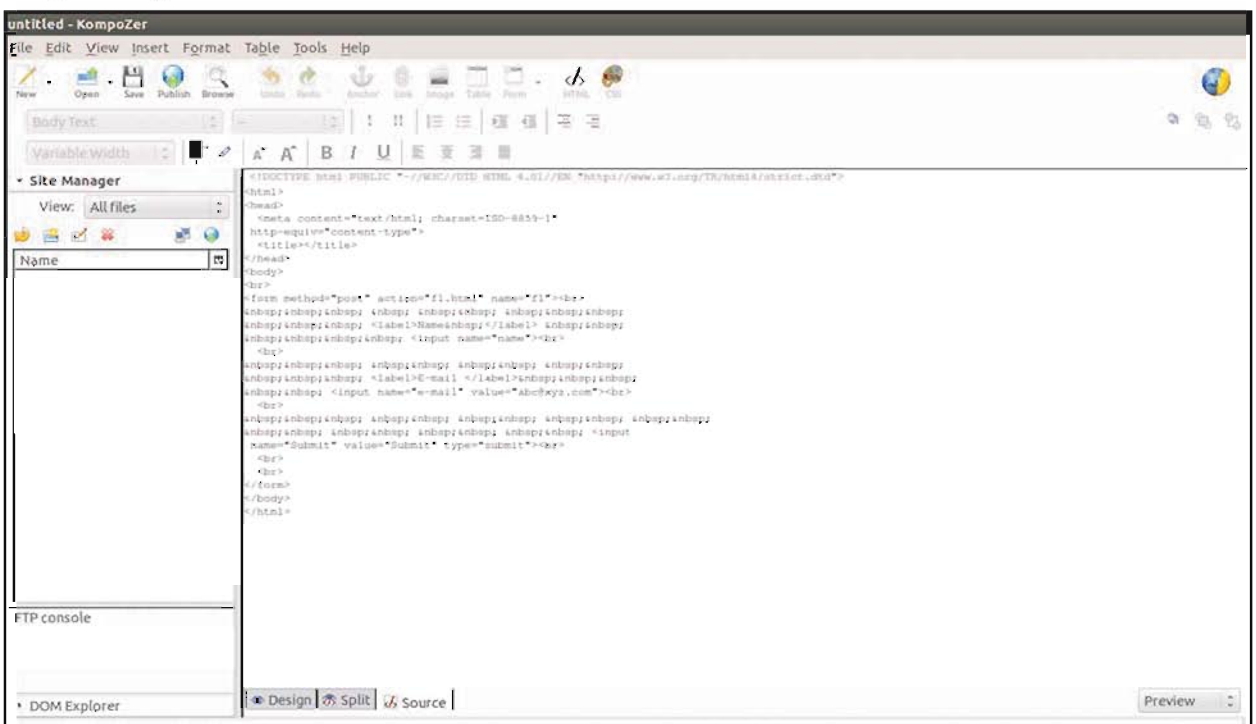
આકૃતિ 1.11 : ઇનપુટ ફિલ્ડ સબમિટ બટન

- હાલમાં ફૉર્મ સામાન્ય દેખાવ (Normal View) સ્વરૂપે છે. ફૉર્મના પૂર્વદર્શન (preview) માટે એડિટમોડ ટૂલબાર પર આવેલાં **Preview** ટૂલબટન પર ક્લિક કરો. આકૃતિ 1.12માં ફૉર્મ પ્રિવ્યૂ મોડમાં દર્શાવ્યું છે.



આકૃતિ 1.12 : પ્રિવ્યૂ મોડમાં ફૉર્મનો દેખાવ

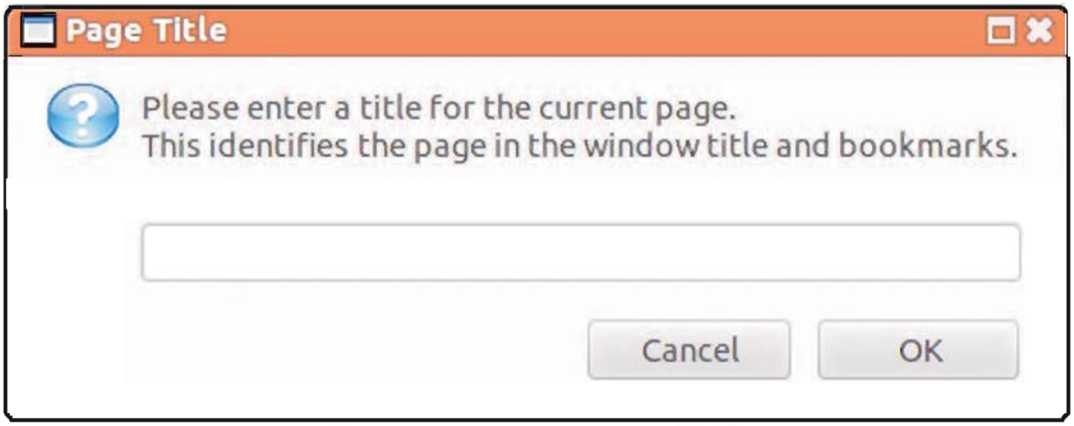
આમ, કમ્પોઝરનો ઉપયોગ કરી આપણે પ્રથમ ફૉર્મની રચના કરી. તમે જોઈ શકો છો કે ટૂંકા સમયમાં ફૉર્મની રચના કરવા માટે કમ્પોઝર કેવી રીતે મદદરૂપ બને છે તથા તે લાંબા સમય સુધી સૉર્સકોડ લખવાના કંટાળાજનક કાર્યમાંથી પણ આપણને મુક્ત કરે છે. Source વિભાગ પર ક્લિક આપી હાલમાં બનાવેલ ફૉર્મના સૉર્સકોડને પણ જોઈ શકાય છે. જુઓ આકૃતિ 1.13.



આકૃતિ 1.13 : ફૉર્મના સૉર્સકોડનો દેખાવ

- તૈયાર થયેલ ફાઇલનો હવે સંગ્રહ કરીએ. **File → Save** વિકલ્પ પર હવે ક્લિક કરો. વૈકલ્પિક રીતે કમ્પોઝિશન ટૂલબાર પર આવેલા

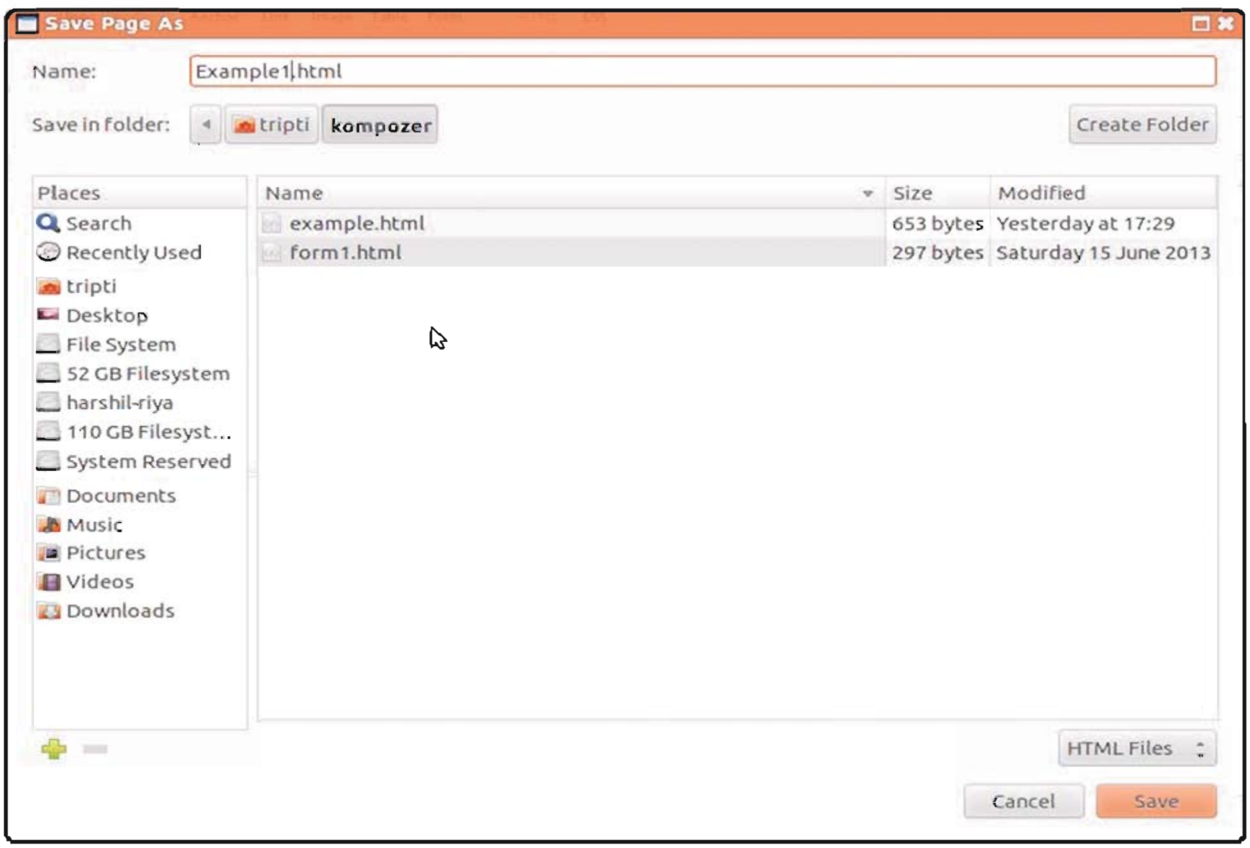
 બટન પર પણ ક્લિક કરી શકાય. આમ કરવાથી આકૃતિ 1.14માં દર્શાવેલ Page Title ડાયલોગબૉક્સ ખૂલશે. અહીં વેબપેજને યોગ્ય શીર્ષક આપી શકાશે. આપણે તેને example1 નામ આપ્યું છે. હવે, OK બટન પર ક્લિક કરો.



આકૃતિ 1.14 : Page Title ડાયલોગબૉક્સ

વેબપેજને બ્રાઉઝરમાં દર્શાવવામાં આવે ત્યારે બ્રાઉઝર વિન્ડોના ટાઇટલબારમાં પાનાંનું શીર્ષક જોઈ શકાય છે. જો આપણે એકથી વધુ વેબપેજની રચના કરી હોય, તો તેવા કિસ્સામાં પાનાંનાં શીર્ષક તરીકે વેબસાઇટનું નામ આપવું જોઈએ. આપણે ફૉર્મ સાથેના એક જ વેબપેજની રચના કરી હોવાથી આ ઉદાહરણમાં શીર્ષક પાનાંને example1 નામ આપ્યું છે.

- OK બટન પર ક્લિક કર્યા બાદ આકૃતિ 1.15માં દર્શાવેલ Save Page As ડાયલોગબૉક્સ ખૂલશે, જે ફાઇલનું નામ અને ફાઇલનો સંગ્રહ કરવા માટેના સ્થાન વિશેની માહિતી પૂછશે.



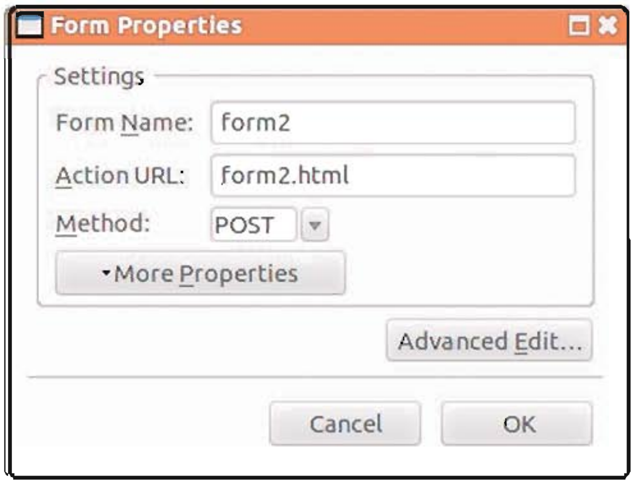
આકૃતિ 1.15 : Save Page As ડાયલોગબૉક્સ

ફાઇલને ***html*** કે ***htm*** અનુલંબન (extension) આપીને સંગ્રહ કરવાનું યાદ રાખો. Save બટન પર ક્લિક કરો. આમ કરવાથી ફરી મુખ્ય વિન્ડો પર જઈ શકાશે.

નોંધ : વેબસાઇટની રચના કરતી વખતે જો કોઈ પાનું હોમપેજ હોય કે જે વેબસાઇટનું URL ટાઇપ કરવાથી ખૂલે તેમ રાખવાનું હોય, તો તેનો index.html નામ સાથે સંગ્રહ કરો.

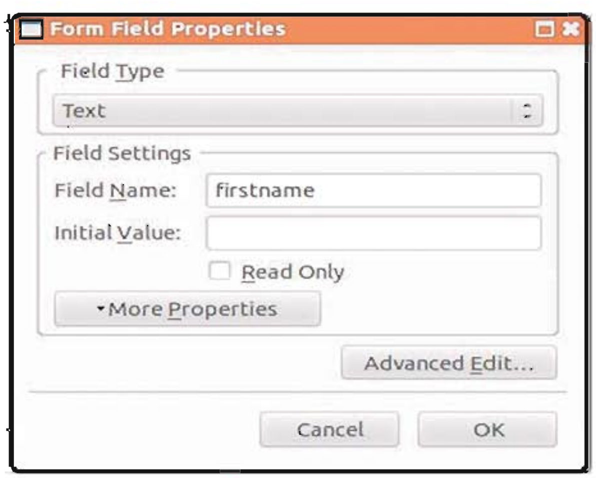
કમ્પોઝરમાં સરળ ફૉર્મની રચના કરતાં, તેને ખોલતાં તથા સંગ્રહ કરતાં શીખ્યા પછી હવે આપણે HTML ટૅગનો ઉપયોગ કરી પહેલાં બનાવ્યું હતું તેવું જ એક નોંધણીફૉર્મ ફરી બનાવીએ. નોંધણી માટેના ફૉર્મની રચના કરવા માટે નીચેનાં પગલાંને અનુસરો :

- નવી ફાઇલની રચના કરો.
- મેનુબારમાંથી **Insert → Form → Define Form** વિકલ્પ પસંદ કરો. Form Properties ડાયલોગબૉક્સમાં આકૃતિ 1.16માં દર્શાવ્યા મુજબની વિગતો ઉમેરો.



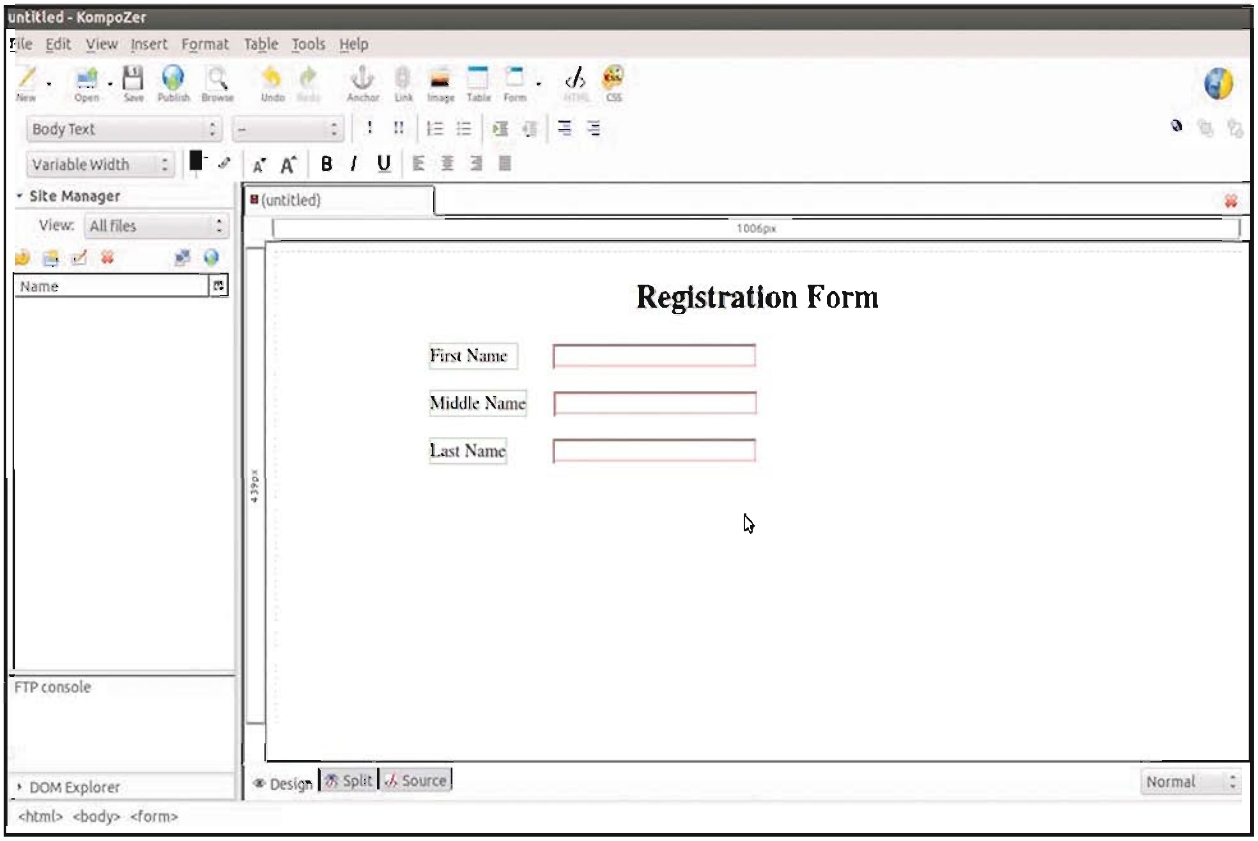
આકૃતિ 1.16 : Form Properties ડાયલોગબૉક્સ

- OK બટન પર ક્લિક કરો. આછા ભૂરા રંગની સીમારેખા ધરાવતું ફૉર્મ દર્શાવવામાં આવશે. ફૉર્મમાં જગ્યા ઉમેરવા માટે એન્ટર કી દબાવો.
- ફૉર્મને શીર્ષક આપવા માટે ફૉર્મેટ ટૂલબાર-1માંથી Heading-1 પસંદ કરો. ફૉર્મેટ ટૂલબાર-2માંથી Centre Align આઇકોન પસંદ કરો. "Registration Form" લખાણ ઉમેરો.
- લેબલ ઉમેરવા માટે **Form → Define Label** વિકલ્પ પસંદ કરો. Field Name ટૅક્સ્ટબૉક્સમાં "First Name" ટાઇપ કરો. ત્યાર બાદ "First name" લેબલ માટેનું ઇનપુટ ફિલ્ડ ઉમેરવા માટે **Form → Form Field** વિકલ્પ પર ક્લિક કરો. Field Type મેનુમાંથી ટૅક્સ્ટ વિકલ્પ પસંદ કરો. આકૃતિ 1.17 Form Field Properties ડાયલોગબૉક્સ દર્શાવે છે. આપણે ફિલ્ડનાં નામ તરીકે "firstname" લખાણનો ઉપયોગ કર્યો છે, તે નોંધો. OK બટન પર ક્લિક કરો.



આકૃતિ 1.17 : First name માટે Form Field Properties ડાયલોગબૉક્સ

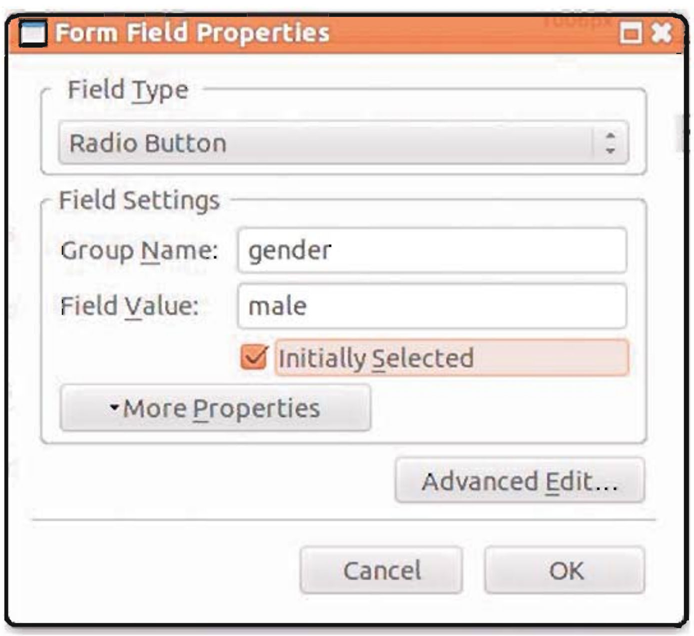
● આવી જ રીતે, ફૉર્મમાં "Middle Name" અને "Last Name" લેબલ ઉમેરો. ફિલ્ડ ઉમેર્યા પછી ફૉર્મનો દેખાવ આકૃતિ 1.18માં દર્શાવ્યા મુજબ દેખાશે.



આકૃતિ 1.18 : ફિલ્ડ ઉમેર્યા પછી ફૉર્મનો દેખાવ

● હવે, Gender ફિલ્ડ માટે રેડિયો-બટન ઉમેરવાની જરૂર છે. પ્રથમ, "Gender" નામના લેબલની રચના કરો.

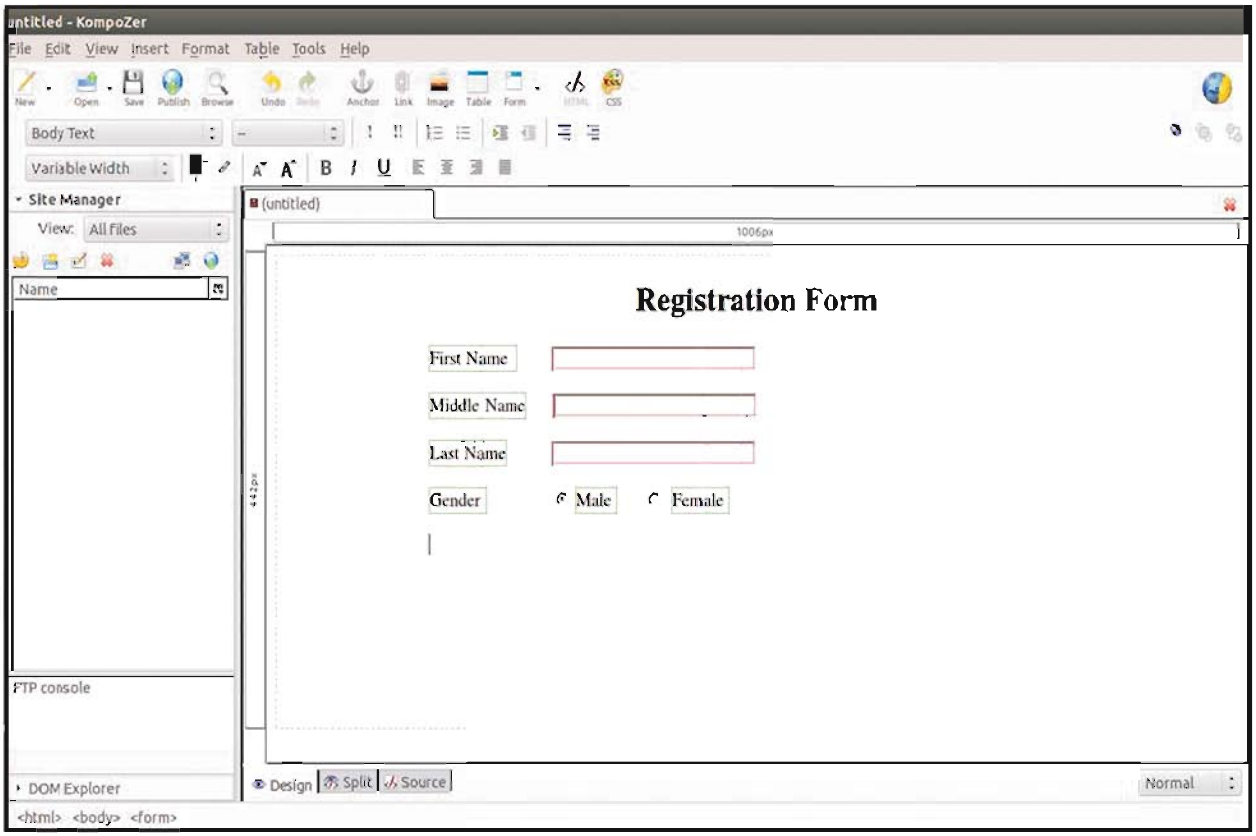
● રેડિયો-બટન ઉમેરવા માટે **Form → Form Field** વિકલ્પ પસંદ કરો અને આકૃતિ 1.19માં દર્શાવ્યા મુજબ ડ્રૉપડાઉન મેનુમાંથી Field Typeના વિકલ્પ તરીકે Radio Button પસંદ કરો.



આકૃતિ 1.19 : રેડિયો-બટન માટેની Form Field Properties

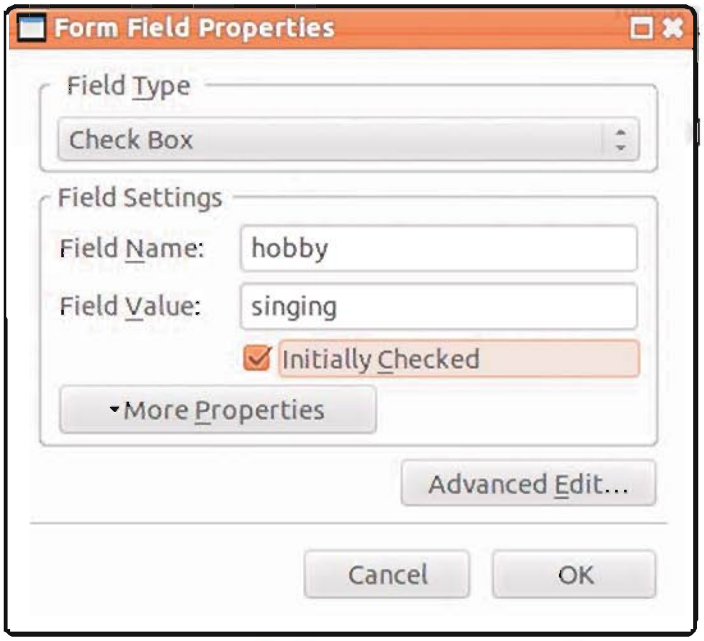
Group-Name બૉક્સમાં નામ ટાઈપ કરો. (નામમાં જગ્યા (space) સમાવી શકાતી નથી તેની નોંધ કરો.) અહીં આપણે Group Name તરીકે "gender" લખાણ ઉમેર્યું છે. આ જ રીતે Field Value ટેક્સ્ટબૉક્સમાં "male" ટાઈપ કરો. જ્યારે ફૉર્મ ખૂલે, ત્યારે જો આપણે male વિકલ્પને પૂર્વનિર્ધારિત રીતે પસંદ કરેલો રાખવા ઇચ્છતા હોઈએ, તો "Initially Selected" લખાણ આગળ આવેલું ચેકબૉક્સ પસંદ કરી OK બટન પર ક્લિક કરો.

- આ રેડિયો-બટનની પાસે "Male" શીર્ષક સાથે એક લેબલ ઉમેરો.
- આ જ રીતે, "Female" નામના અન્ય રેડિયો-બટનની રચના કરો. યાદ રાખો કે, જ્યારે આપણે રેડિયો-બટનની રચના જૂથમાં કરતા હોઈએ, ત્યારે તમામ શક્ય જવાબો માટે જૂથનું નામ (Group Name) એકસમાન હોવું જોઈએ. માટે Group Name તરીકે "gender" ઉમેરો. Field Value ટેક્સ્ટબૉક્સમાં "female" ટાઈપ કરો. OK બટન પર ક્લિક કરો.
- આ રેડિયો-બટનની પાસે "Female" લેબલ ઉમેરો. રેડિયો-બટન અને તેના લેબલ ઉમેર્યા પછી ફૉર્મ આકૃતિ 1.20માં દર્શાવ્યા મુજબ દેખાશે.



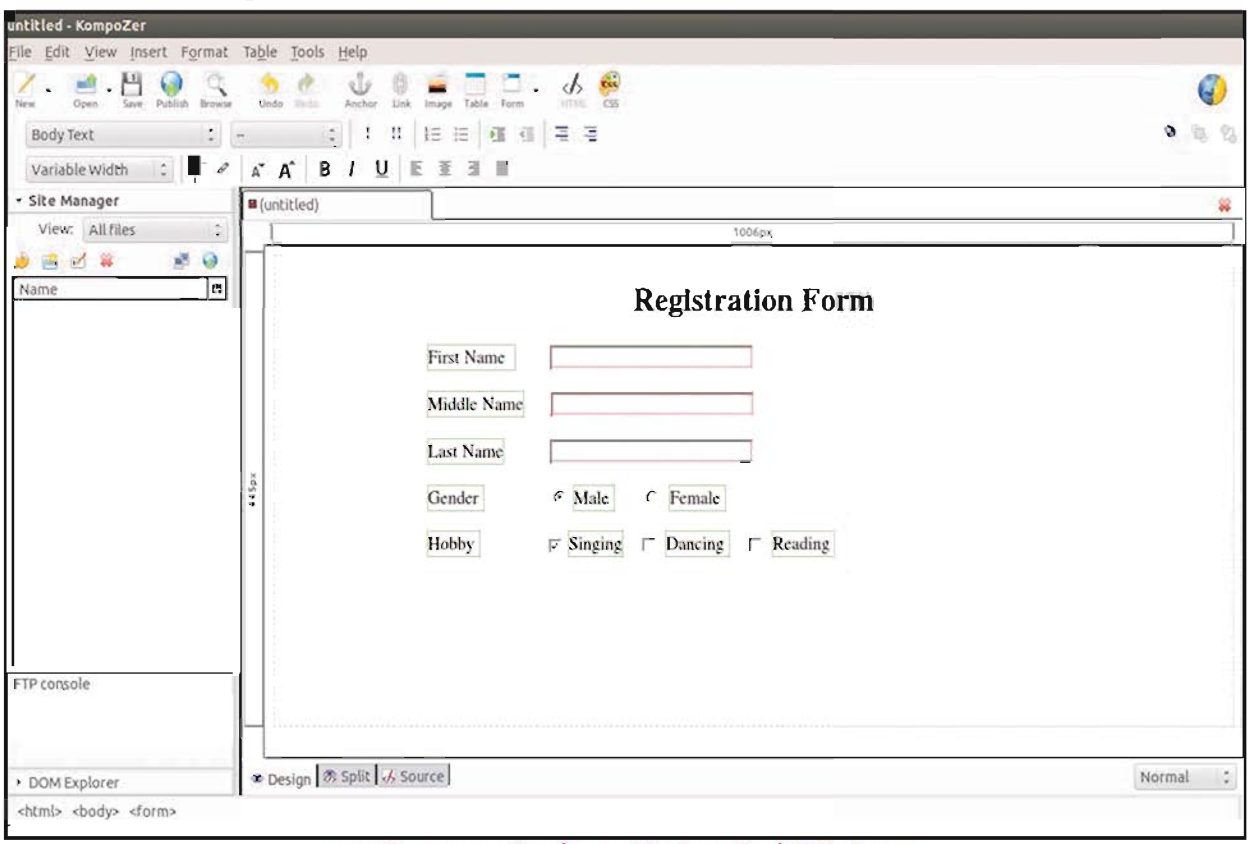
આકૃતિ 1.20 : રેડિયો-બટન ઉમેર્યા પછી ફૉર્મનો દેખાવ

- હવે આપણને "Hobby" ફિલ્ડની જરૂર છે. વ્યક્તિને એકથી વધુ શોખ હોઈ શકે છે, માટે Hobby ફિલ્ડ માટે એકથી વધુ વિકલ્પોની પસંદગી શક્ય હોવી જોઈએ. આમ, Hobby ફિલ્ડ માટે આપણને ચેકબૉક્સ બનાવવાની જરૂર છે. Hobby માટે લેબલની રચના કરો.
- હવે, **Form → Form Field** વિકલ્પ પસંદ કરી ડ્રોપડાઉન મેનુમાંથી ફિલ્ડના પ્રકાર તરીકે CheckBox પસંદ કરો. આકૃતિ 1.21માં દર્શાવ્યા પ્રમાણે, Field Name બૉક્સમાં નામ અને Field Value બૉક્સમાં કિંમત ઉમેરો. ફૉર્મ ખૂલે ત્યારે આ વિકલ્પ પૂર્વનિર્ધારિત રીતે પસંદ થયેલો રાખવા માટે "Initially Selected" લખાણ આગળ આવેલા ચેકબૉક્સને પસંદ કરો. OK બટન પર ક્લિક કરો.



આકૃતિ 1.21 : ચેકબૉક્સ પ્રકારના ફિલ્ડની ગોઠવણ

- આ ચેકબૉક્સની બાજુમાં "Singing" લેબલ ઉમેરો.
- આ જ રીતે, Field Value તરીકે "dancing" અને "reading" ઉમેરી અન્ય બે ચેકબૉક્સની રચના કરો. ચેકબૉક્સના તમામ વિકલ્પોનાં Field Name સમાન હોય તે યાદ રાખો.
- ચેકબૉક્સની બાજુમાં "Dancing" અને "Reading" લેબલ ઉમેરો. ચેકબૉક્સ અને તેનાં લેબલ ઉમેર્યા પછી ફૉર્મ આકૃતિ 1.22માં દર્શાવ્યા મુજબ દેખાશે.



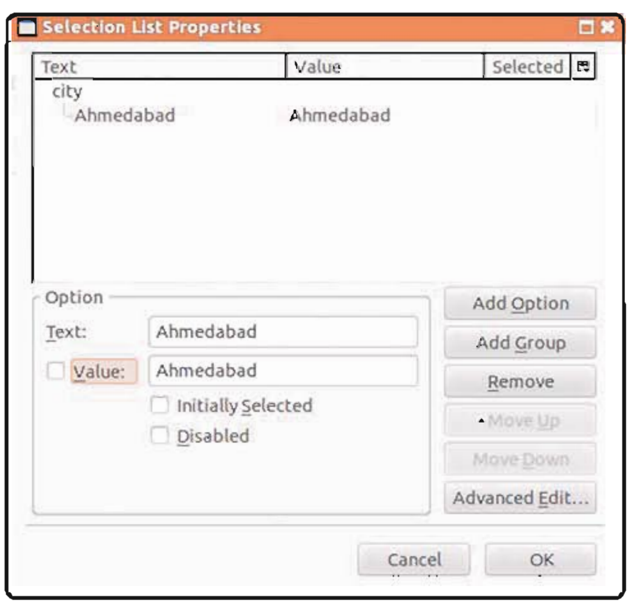
આકૃતિ 1.22 : ચેકબૉક્સ ઉમેર્યા પછી ફૉર્મનો દેખાવ

- ત્યાર બાદ આપણને "Address" ફિલ્ડની જરૂર છે. ઉપયોગકર્તા સરનામાંનાં ફિલ્ડમાં વધુ લીટીનું લખાણ ઉમેરી શકે છે. માટે તેના ફિલ્ડનો પ્રકાર ટેક્સ્ટ-એરિયા રાખીશું. પ્રથમ "Address" નામના લેબલની રચના કરો. હવે, **Form → Text Area** વિકલ્પ પસંદ કરો. આકૃતિ 1.23માં દર્શાવેલ Text Area Properties ડાયલોગબૉક્સ ખૂલશે.

આકૃતિ 1.23 : Text Area Properties ડાયલોગબૉક્સ

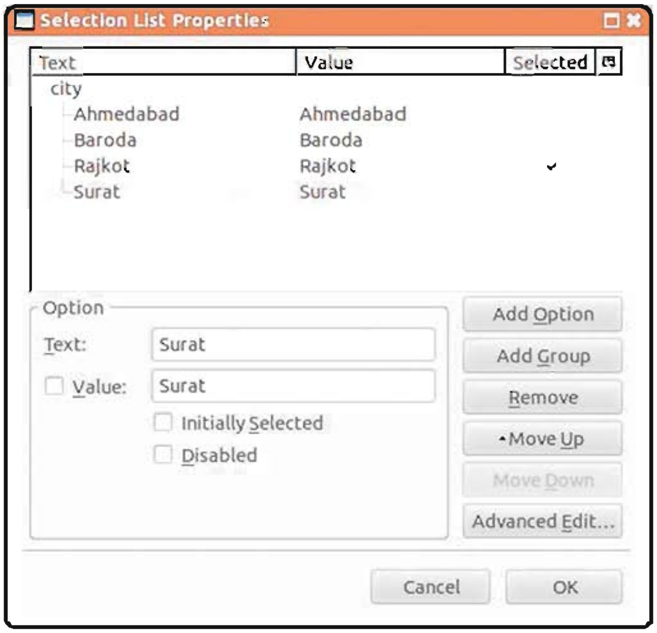
Field Name બૉક્સમાં નામ ઉમેરો. ટેક્સ્ટ-એરિયા માટે જરૂરી હરોળ અને સ્તંભની સંખ્યા ઉમેરો. અહીં આપણે હરોળની સંખ્યા 5 અને સ્તંભની સંખ્યા 70 રાખી છે. ફોર્મ ખૂલે ત્યારે ફિલ્ડમાં પૂર્વનિર્ધારિત રીતે દર્શાવવામાં આવનાર યોગ્ય લખાણને Initial Text ફિલ્ડમાં ઉમેરો. OK બટન પર ક્લિક કરો.

- ત્યાર બાદ આપણે "City" ફિલ્ડ ઉમેરીશું. શહેર માટેનાં ફિલ્ડનું લેબલ ઉમેરો. ઉપયોગકર્તાને ડ્રૉપડાઉન મેનુમાંથી શહેરની પસંદગી અંગે પૂછવામાં આવશે. તેથી શહેરના ફિલ્ડ માટે પસંદગીયાદી (Selection List)નો ઉપયોગ કરવાની જરૂર છે. **Form → Selection List** વિકલ્પ પસંદ કરો. આકૃતિ 1.24માં દર્શાવ્યા મુજબ Selection List Properties ડાયલોગબૉક્સ ખૂલશે.



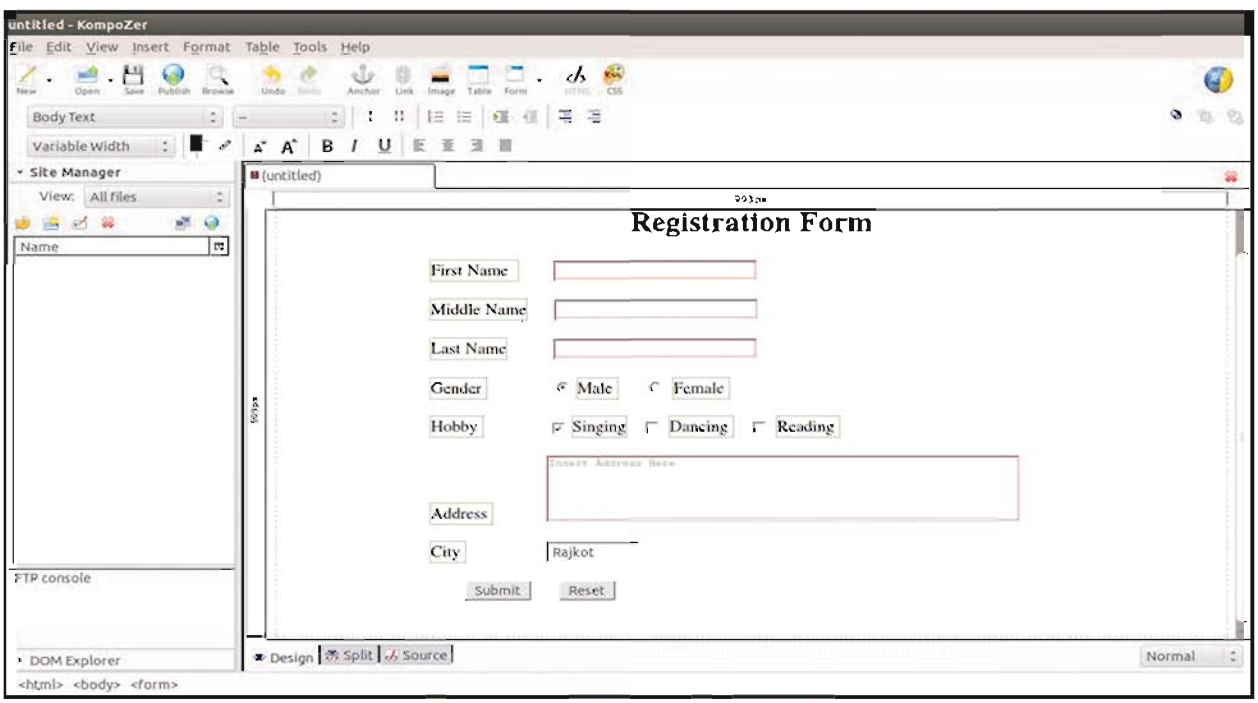
આકૃતિ 1.24 : Selection List Properties ડાયલોગબૉક્સ

List Name બૉક્સમાં "city" ટાઇપ કરી, Add Option બટન પર ક્લિક કરો. ત્યાર પછી, Textfieldમાં "Ahmedabad" ટાઇપ કરો. Add Option વિકલ્પ પર ફરી ક્લિક કરી શહેરનું નામ "Baroda" ઉમેરો. આ જ પ્રમાણે, "Rajkot" અને "Surat" શહેર પણ ઉમેરો. રાજકોટ શહેર ઉમેરતી વખતે "Initially Selected" વિકલ્પ પસંદ કરવાનું યાદ રાખો. OK બટન પર ક્લિક કરો. શહેરનું નામ ઉમેર્યા બાદ Selection List Properties ડાયલોગબૉક્સનો દેખાવ આકૃતિ 1.25માં દર્શાવ્યો છે.



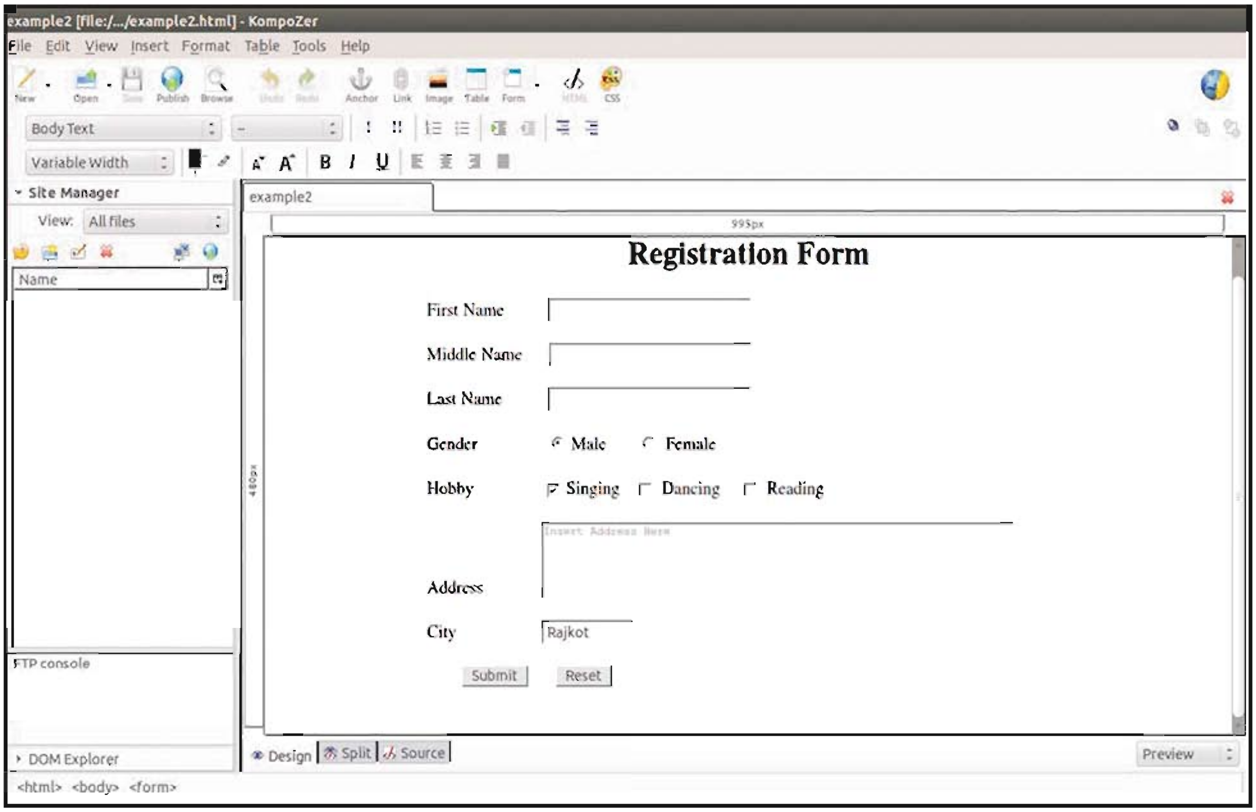
આકૃતિ 1.25 : વિવિધ શહેરો માટેનું Selection List Properties ડાયલોગબૉક્સ

- ત્યાર બાદ, આપણે "Submit" બટન ઉમેરીશું. **Form → Form Field** વિકલ્પ પસંદ કરો. ડ્રોપડાઉન મેનુમાંથી Submit Button વિકલ્પ પસંદ કરો. Field Name અને Field Value ટેક્સ્ટબૉક્સમાં Submit લખાણ ઉમેરો. OK બટન પર ક્લિક કરો.
- આ જ રીતે, આપણે "Reset" બટન ઉમેરીશું. ડ્રોપડાઉન મેનુમાંથી Reset Button વિકલ્પ પસંદ કરો. Field Name અને Field Value ટેક્સ્ટબૉક્સમાં Reset લખાણ ઉમેરો. OK બટન પર ક્લિક કરો. આકૃતિ 1.26માં ફૉર્મનો અંતિમ દેખાવ નોર્મલ વ્યૂ દ્વારા દર્શાવ્યો છે.



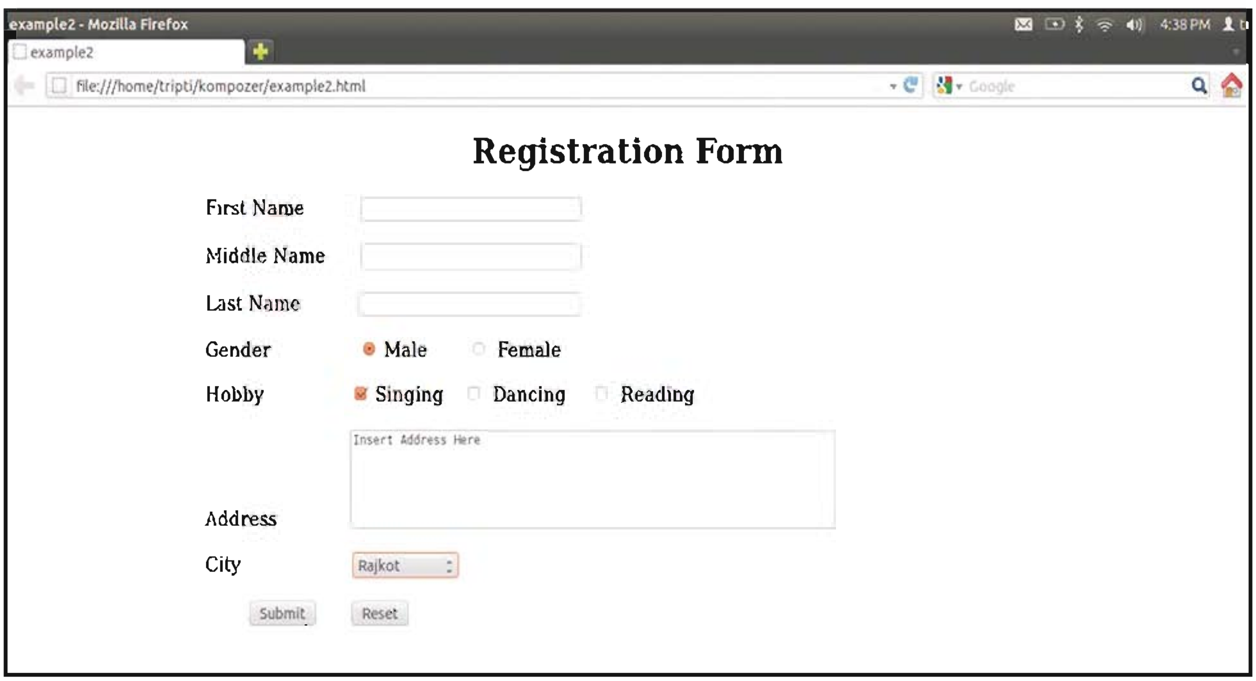
આકૃતિ 1.26 : નોર્મલ વ્યૂમાં ફૉર્મનો દેખાવ

ફાઈલનો "example2" નામથી સંગ્રહ કરો. ફૉર્મનો પ્રિવ્યૂ મોડ આકૃતિ 1.27માં દર્શાવ્યો છે.



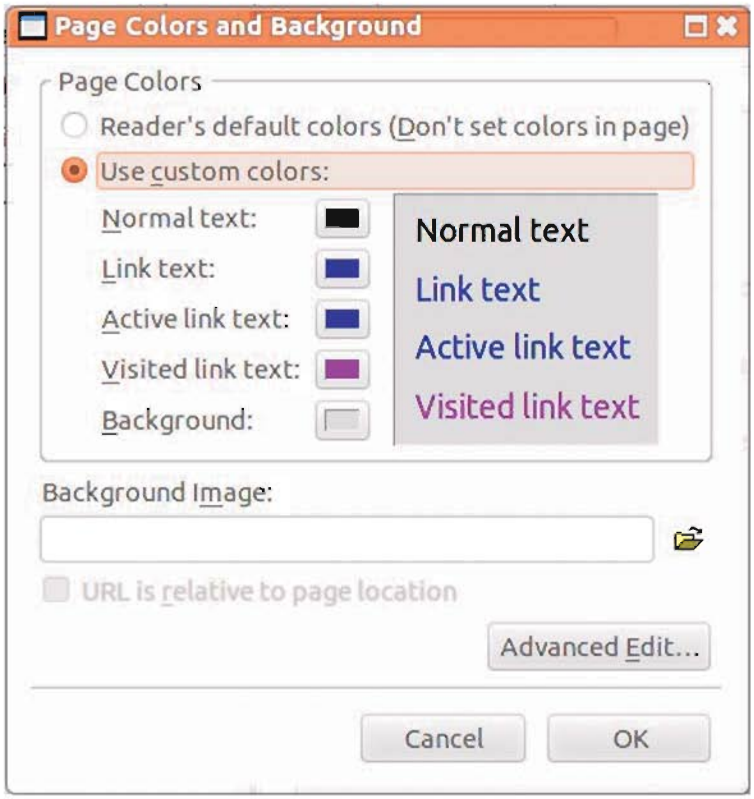
આકૃતિ 1.27 : પ્રિવ્યૂ મોડમાં ફૉર્મની રજૂઆત

આકૃતિ 1.28 નોંધણી માટેના ફૉર્મનો દેખાવ બ્રાઉઝરમાં દર્શાવે છે.



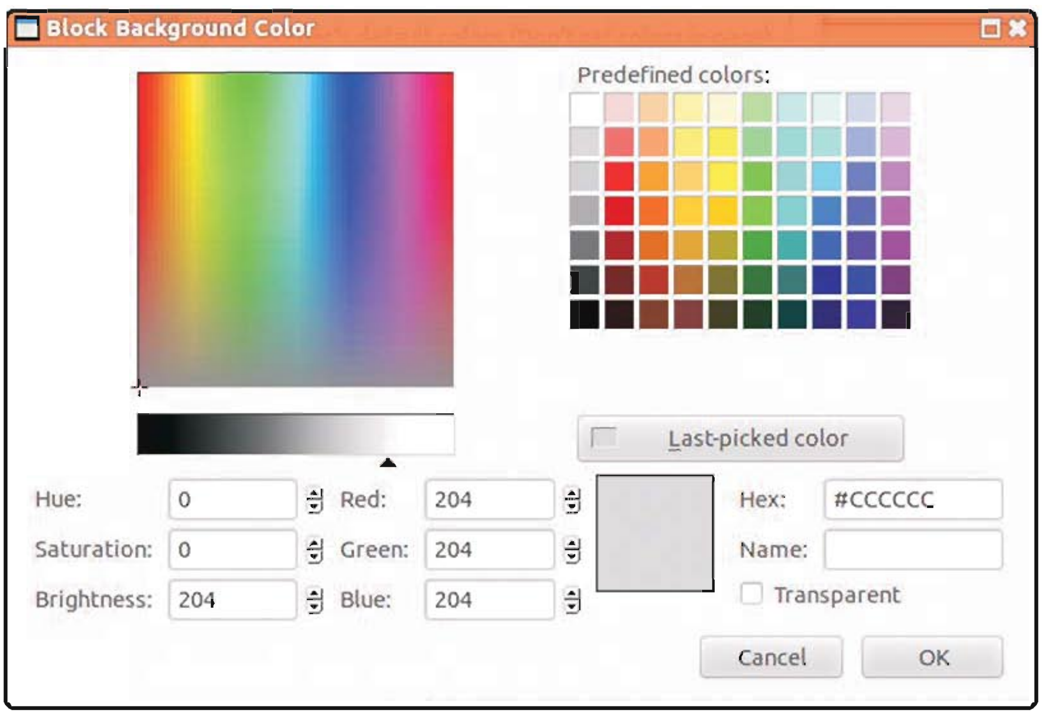
આકૃતિ 1.28 : બ્રાઉઝરમાં ફૉર્મનો દેખાવ

ફૉર્મના બૅકગ્રાઉન્ડનો રંગ સફેદ છે, તેની નોંધ લો. જો ફૉર્મને અન્ય રંગ બૅકગ્રાઉન્ડમાં આપવો હોય, તો **Format → Page Colors and Background** વિકલ્પ પસંદ કરો. આમ કરવાથી આકૃતિ 1.29માં દર્શાવ્યા મુજબનું ડાયલોગબૉક્સ ખૂલશે.



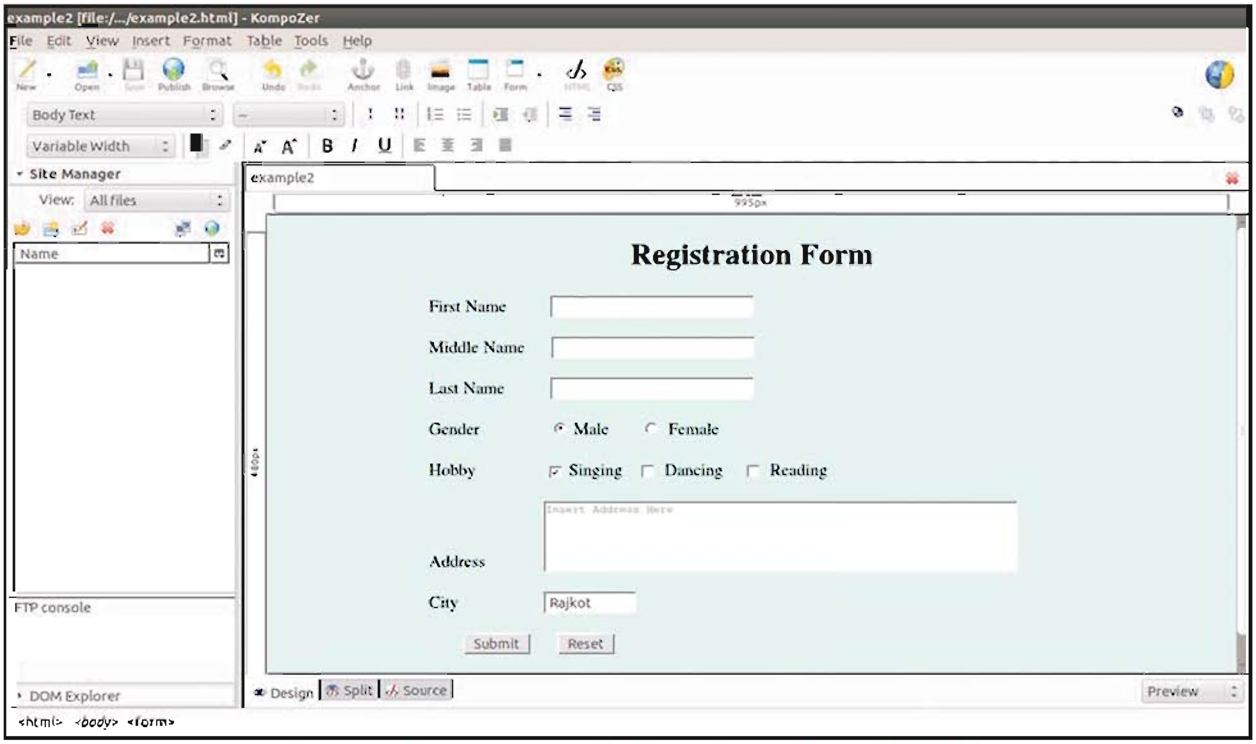
આકૃતિ 1.29 : Page Colors and Background ડાયલોગબૉક્સ

"Use Custom Colors" વિકલ્પ પસંદ કરો. Background વિકલ્પ પર ક્લિક કરી આકૃતિ 1.30માં દર્શાવ્યા મુજબ Block Background Color ડાયલોગબૉક્સમાંથી તમારી પસંદગીનો રંગ પસંદ કરો. OK બટન પર ક્લિક કરો, આથી ફરી આકૃતિ 1.29માં આપવામાં આવેલું ડાયલોગબૉક્સ રજૂ કરવામાં આવશે. ફરી OK બટન પર ક્લિક કરો.



આકૃતિ 1.30 : બૅકગ્રાઉન્ડ રંગની પસંદગી

રંગની પસંદગી કર્યા પછી ફૉર્મનો દેખાવ આકૃતિ 1.31માં દર્શાવ્યા મુજબ દેખાશે.



આકૃતિ 1.31 : બૅકગ્રાઉન્ડ રંગ ઉમેર્યા બાદ પ્રિવ્યૂ મોડમાં ફૉર્મનો દેખાવ

બ્રાઉઝરનો ઉપયોગ કરી ફૉર્મ જુઓ અને બૅકગ્રાઉન્ડ રંગમાં થયેલું પરિવર્તન ધ્યાનમાં લો.

સારાંશ

વેબ પર વિગતો સ્વીકારવા માટે ફૉર્મનો ઉપયોગ કરવામાં આવે છે. HTMLમાં ફૉર્મ એક સંગ્રાહક છે. ઉપયોગકર્તા પાસેથી જુદા-જુદા પ્રકારના નિવેશ એકત્રિત કરવા માટે ફૉર્મનો ઉપયોગ કરવામાં આવે છે. ઉપયોગકર્તા પોતાની અંગત માહિતી, કોઈ ઉત્પાદન અંગેનો પ્રતિસાદ, મોજણી કે પરિવહનની વિગતો, ક્રેડિટકાર્ડની વિગતો વગેરે ફૉર્મમાં ઉમેરે છે. કમ્પોઝર વેબવિકાસ માટેનું એક નિ:શુલ્ક અને ઑપનસૉર્સ IDE છે, જેના ઉપયોગથી વેબસાઇટની રચના કરી શકાય છે. તે WYSIWYG - "What You See Is What You Get" નામે ઓળખાતા એક સરળ ગ્રાફિકલ ઇન્ટરફેસ ધરાવતું વેબપેજ એડિટર પૂરું પાડે છે. કમ્પોઝરની મદદથી ફૉર્મની રચના સરળતાથી અને ઝડપથી કરી શકાય છે.

સ્વાધ્યાય

1. ફૉર્મ શું છે ? HTMLમાં ફૉર્મની રચના કરવા માટેના ઘટકોની યાદી બનાવો.
2. HTML ફૉર્મમાં નિવેશ માટેના ઘટકોનો ઉપયોગ જણાવો. ઇનપુટ ટૅગની વિવિધ લાક્ષણિકતા વિશે લખો.
3. HTML ફૉર્મમાં ટેક્સ્ટ-એરિયા ઘટકનો હેતુ જણાવો.
4. સિલેક્ટ અને ઑપ્શન વિકલ્પ વિશે જણાવો.
5. કમ્પોઝર વિન્ડોમાં દર્શાવાતા જુદા-જુદા ટૂલબારની યાદી બનાવો.
6. ફૉર્મની બે લાક્ષણિકતાઓ કઈ છે ? સમજાવો.

7. આપેલ વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) ઉપયોગકર્તા પાસેથી જુદા-જુદા પ્રકારના નિવેશ એકત્રિત કરવા માટે નીચેનામાંથી કયા સંગ્રાહક એકમનો ઉપયોગ કરવામાં આવે છે ?

(a) Form (b) Webpage

(c) Text (d) Input

(2) HTML ફૉર્મની રચના કરવા માટે નીચેનામાંથી કયા ઘટકનો ઉપયોગ કરવામાં આવે છે ?

(a) Textarea (b) Form

(c) Select અને Option (d) Input

(3) ફૉર્મ ઘટકનો અમલ કરવા માટે નીચેનામાંથી કયા ટેગનો ઉપયોગ કરવામાં આવે છે ?

(a) <form>... </form> (b) <form>... <form>

(c) </form>... </form> (d) <frm>... </frm>

(4) ફૉર્મ સબમિટ કરતી વખતે ફૉર્મની વિગતો કયા સ્થાને મોકલવાની છે, તેની સ્પષ્ટતા માટે નીચેનામાંથી કઈ લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે ?

(a) method (b) action

(c) submit (d) input

(5) ફૉર્મની વિગતો મોકલતી વખતે HTTP પદ્ધતિ સ્પષ્ટ કરવા માટે નીચેનામાંથી ફૉર્મની કઈ લાક્ષણિકતાનો ઉપયોગ કરવામાં આવે છે ?

(a) submit (b) action

(c) method (d) input

(6) method લાક્ષણિકતા સાથે કઈ કિંમતોનો ઉપયોગ કરી શકાય છે ?

(a) GET અને POST (b) GET અને SET

(c) GET અને PUT (d) SET અને POST

(7) નીચેનામાંથી કઈ પદ્ધતિ એક સમયે માત્ર મર્યાદિત માહિતી મોકલવાની અનુમતિ પૂરી પાડે છે ?

(a) GET (b) POST

(c) SET (d) PUT

(8) નીચેનામાંથી કઈ પદ્ધતિ HTTP વ્યવહાર દ્વારા વિગતોને બ્લૉક સ્વરૂપે મોકલવાની સુવિધા પૂરી પાડે છે ?

(a) GET (b) SET

(c) PUT (d) POST

(9) નીચેનામાંથી કઈ લાક્ષણિકતા દ્વારા ફૉર્મમાં નિર્માણ કરવાના ફિલ્ડના પ્રકારનો સ્પષ્ટ નિર્દેશ કરવામાં આવે છે ?

(a) Input (b) Type

(c) Name (d) Value

(10) નીચેનામાંથી ક્યો ઘટક એકથી વધુ લીટીના નિવેશ માટે અનુમતિ આપે છે ?

(a) Textarea (b) Input

(c) Select અને Option (d) Form

(11) નીચેનામાંથી ક્યા ઘટકનો ઉપયોગ ફૉર્મમાં ડ્રૉપડાઉન યાદી અથવા મેનુની રચના કરવા માટે જરૂરી છે ?

(a) Input (b) Textarea

(c) Select (d) Form

(12) નીચેનાંમાંથી ક્યું નિઃશુલ્ક, ઑપનસૉર્સ વેબવિકાસ માટેનું IDE છે ?

(a) HTML (b) KompoZer

(c) Scite (d) Base

(13) WYSIWYG એટલે શું ?

(a) When You See Is When You Get (b) What You See Is When You Get

(c) What You See Is What You Get (d) When You See Is What You Get

પ્રાયોગિક સ્વાધ્યાય

1. વિદ્યાર્થીઓની અંગત વિગતો મેળવવા માટેના ફૉર્મની રચના કરો.
2. તમારી શાળાના મુલાકાતીને પ્રતિસાદ (feedback) આપવા માટેના ફૉર્મની રચના કરો.
3. તમે તમારાં માતા-પિતા સાથે રજાઓમાં ફરવા ગયા હતા; પ્રવાસ-આયોજકે તમારી પાસે પ્રવાસને લગતી સમીક્ષા (review)માંગી છે. આ માટેના ફૉર્મની રચના કરો.

કેસ્કેડિંગ સ્ટાઇલશીટ અને જાવાસ્ક્રિપ્ટ 2

કેસ્કેડિંગ સ્ટાઇલશીટ (Cascading Style- Sheet) અથવા CSS વેબસાઇટમાં આવેલા દર્શનીય ઘટકોની શૈલી (Style) સ્પષ્ટ કરવા માટેની સુવિધા પૂરી પાડે છે. દસ્તાવેજની માહિતીને તેને દર્શાવવાની વિગતોથી અલગ રાખવા તે મદદરૂપ બને છે. દસ્તાવેજને કેવી રીતે દર્શાવવો તેની વિગતોને સ્ટાઇલ (Style) તરીકે ઓળખવામાં આવે છે. સ્ટાઇલ જે-તે ઘટકનો દેખાવ સ્ક્રીન પર નક્કી કરે છે. સ્ટાઇલને અન્ય માહિતીથી અલગ રાખવાથી નીચે જણાવેલ લાભ મેળવી શકાય :

- કોડિંગમાં નકલ (duplication) ટાળી શકાય.
- એક જ વિગત માટે જુદા-જુદા ઉદ્દેશોને અનુલક્ષીને જુદી-જુદી સ્ટાઇલનો ઉપયોગ કરી શકાય.
- કોડની સરળ જાળવણી કરી શકાય.

ઉદાહરણ તરીકે, HTMLમાં આપણે ઇચ્છતા હોઈએ કે <p> ટેગથી ઉમેરેલા કેટલાક ફકરાનું લખાણ ઘાટા (bold) અક્ષરોમાં આવવું જોઈએ, તો તે માટે સૉર્સકોડમાં આવેલ તમામ ફકરાના <p> ટેગ માટે અક્ષરો ઘાટા કરવા માટેનો <b> ટેગ ઉમેરવો પડે. જો વેબસાઇટ વિશાળ હોય તો ટેગનું આ પુનરાવર્તન કંટાળાજનક અને સમયનો વ્યય કરનારું બની રહે. પરંતુ CSSનો ઉપયોગ કરી આપણે તમામ ઘટકોનો પ્રકાર આપણી પસંદગી પ્રમાણેની સ્ટાઇલ સાથે દર્શાવી શકીએ. માટે, વેબસાઇટમાં વધુ સંખ્યામાં <p> ટેગ આવેલા હોય તોપણ, CSSનો ઉપયોગ કરી, તમામ <p> ટેગ માટે એકસાથે સ્ટાઇલ ગોઠવી શકીએ. આમ, એ કહી શકાય કે, HTMLનો ઉપયોગ માહિતીના વર્ણન માટે કરવામાં આવે છે, શૈલી માટે નહીં, જ્યારે CSS દસ્તાવેજમાં ઘટકોની શૈલીનું વર્ણન કરે છે, તેની વિગતોનું નહીં. વેબસાઇટમાં CSSનો ઉપયોગ કરી ફોન્ટના પ્રકાર, ફોન્ટ અને ઘટકોના રંગ, આજુબાજુની જગ્યા (Pad Space), હાંસિયા અને ઘટકોની સ્થિતિનું નિયંત્રણ કરી શકાય છે.

CSSની વાક્યરચના (Syntax of CSS)

CSSની વાક્યરચનામાં rules નામે ઓળખાતાં વિશિષ્ટ ચિહ્નોનો ઉપયોગ કરવામાં આવે છે. CSS rulesના બે મુખ્ય વિભાગ છે : પસંદગીકાર (selector) અને એક કે વધુ ઘોષણાઓ (declarations). સિલેક્ટર એ HTML ઘટક છે, જેના પર આપણે સ્ટાઇલને લાગુ કરવા માંગીએ છીએ. સિલેક્ટર તરીકે ઉપયોગમાં લેવામાં આવેલ HTML ઘટક સાથે સંકળાયેલ ગુણધર્મ અને તેને અનુરૂપ કિંમતોનો સમાવેશ ડિક્લેરેશન વિભાગમાં કરવામાં આવે છે. CSSની સામાન્ય વાક્યરચના નીચે મુજબ વ્યાખ્યાયિત કરી શકાય :

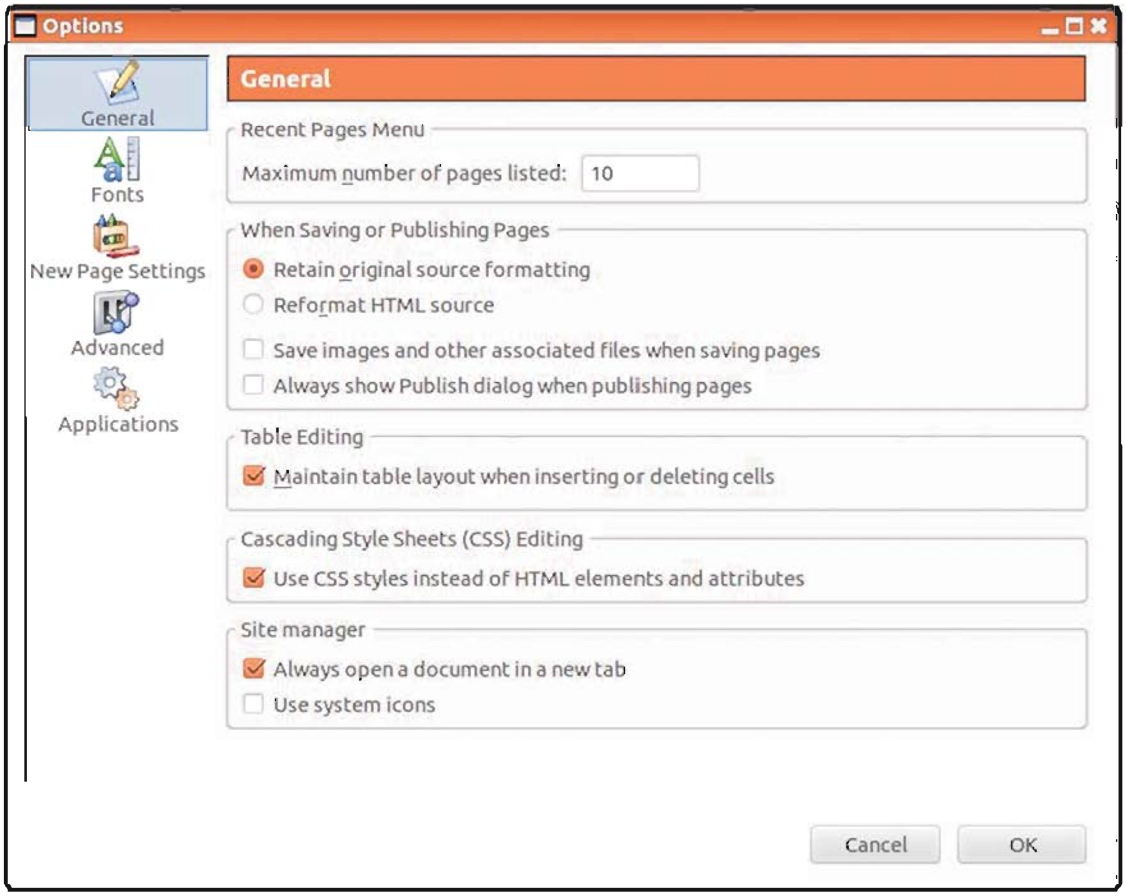
selector {property : value}

આ વાક્યરચનાનું એક ઉદાહરણ આકૃતિ 2.1માં આપ્યું છે.

```
H1{ color : green }
 |      |       |
Selector Property Value
```

આકૃતિ 2.1 CSSની વાક્યરચના

કમ્પોઝરનો ઉપયોગ કરી CSS સરળતાથી ઉમેરી શકાય છે. હવે, કમ્પોઝરમાં વેબપેજ સાથે CSS કેવી રીતે ઉમેરી શકાય, તેનો અભ્યાસ કરીએ. કમ્પોઝર પૂર્વનિર્ધારિત રીતે CSSનો ઉપયોગ કરે છે, તેની ખાતરી કરવા **Tools → Preferences** વિકલ્પ પસંદ કરો. આમ કરવાથી આકૃતિ 2.2માં દર્શાવ્યા મુજબ Option ડાયલોગબૉક્સ ખૂલશે. વિન્ડોની ડાબી બાજુ આવેલ General વર્ગ પસંદ કરો. તેમાં "Use CSS Style Instead of HTML Elements and Attributes" પસંદ કરેલ ન હોય, તો તેને પસંદ કરો. હવે, કમ્પોઝરમાં HTMLને બદલે લખાણની ગોઠવણી માટે CSS સ્ટાઇલનો ઉપયોગ કરવામાં આવશે.



આકૃતિ 2.2 Options ડાયલોગબૉક્સ

ધારો કે, આપણે એક એવી વેબસાઇટની રચના કરવી છે, જેનાં તમામ વેબપેજનાં શીર્ષકો એક નિશ્ચિત શૈલીને અનુસરે. ઉદાહરણ તરીકે, વેબપેજમાં ઉમેરવામાં આવેલાં તમામ Heading1 (h1) નીચે આપેલ શૈલીને અનુસરવાં જોઈએ :

➢ Font : Times New Roman

➢ Case : Uppercase

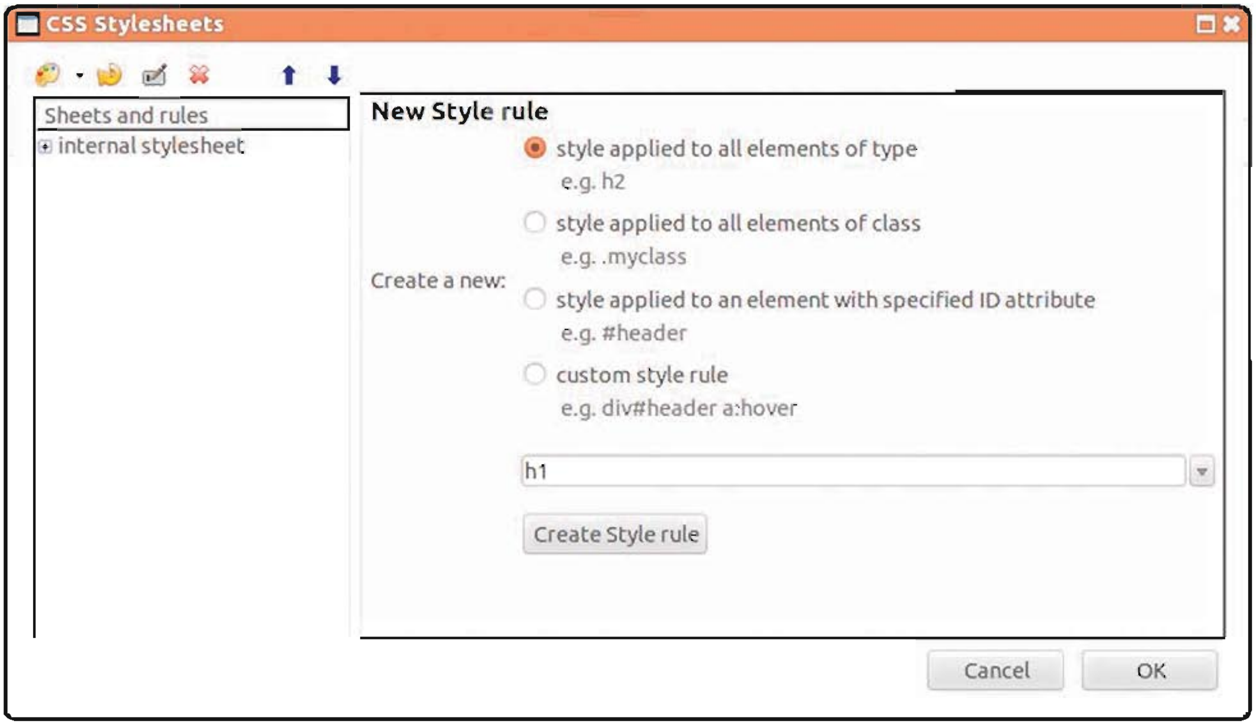
➢ Alignment : Centre aligned

➢ Background color : Light Blue

➢ Border : Dotted Border

ઉપરનાં શીર્ષક માટે CSSની રચના કરવા માટે નીચે આપેલ પગલાંને અનુસરો.

- નવી ફાઇલ ખોલો. ફાઇલનો શીર્ષક આપી સંગ્રહ કરો.
- કમ્પોઝિશન ટૂલબાર પર Cascade બટન પર ક્લિક કરો. (નોંધ : જો ફાઇલનો સંગ્રહ કરવામાં ન આવ્યો હોય તો, Page title ડાયલોગબૉક્સ ખૂલશે. પાનાંને શીર્ષક આપ્યા પછી ફાઇલનો સંગ્રહ કરો.) આમ કરવાથી આકૃતિ 2.3માં દર્શાવ્યા મુજબનું ડાયલોગબૉક્સ ખૂલશે. આ ડાયલોગબૉક્સનો ઉપયોગ કરી દરેક ઘટક માટે સ્ટાઇલ વ્યાખ્યાયિત કરી શકાય છે. પ્રથમ રેડિયો-બટન "style applied to all the elements of type" પર ક્લિક કરો.

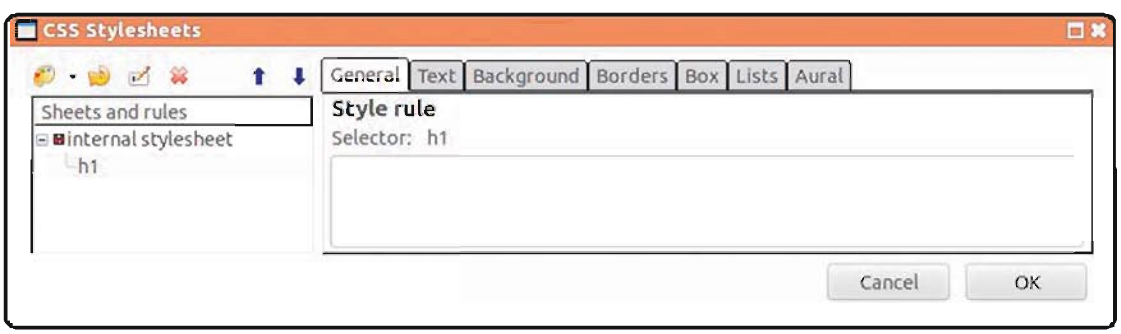


આકૃતિ 2.3 : CSS Stylesheets ડાયલોગબૉક્સ

- જેના માટે શૈલીના નિયમો (Style rule) વ્યાખ્યાયિત કરવાના હોય તે ઘટકને ડ્રૉપડાઉન મેનુમાંથી પસંદ કરો. આપણે Heading1 માટે શૈલીની રચના કરવાની હોવાથી આકૃતિ 2.3માં ડ્રૉપડાઉન મેનુમાંથી h1 (Heading1) પસંદ કરવામાં આવ્યું છે.

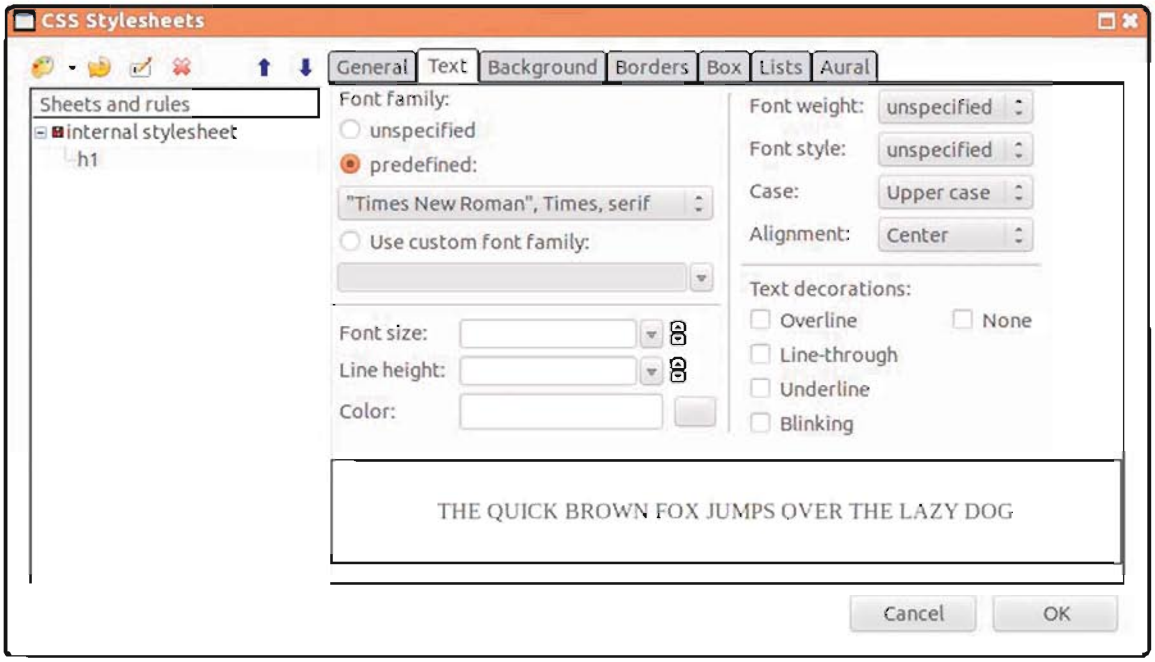
- Create Style rule બટન પર ક્લિક કરો. જોઈ શકાશે કે CSS Stylesheets ડાયલોગબૉક્સ ખૂલ્લું રહે છે, પરંતુ તેમાંનાં વિકલ્પો બદલાઈ ગયા છે.

- આકૃતિ 2.4માં ડાબી બાજુના વિભાગમાં internal stylesheet શીર્ષકની તરત નીચે h1 ઘટક દર્શાવ્યો છે. જે ઘટક માટે શૈલીના નિયમ વ્યાખ્યાયિત કરવામાં આવે તે દરેકને internal stylesheet શીર્ષક હેઠળ આ યાદીમાં દર્શાવવામાં આવે છે. ડાયલોગબૉક્સમાં જમણી બાજુ General, Text, Background, Borders, Box, Lists અને Aural જેવા વિવિધ વિભાગો (tabs) જોઈ શકાય છે. ઘટકને નિશ્ચિત શૈલી આપવા માટે આ દરેક વિભાગનો ઉપયોગ કરી શકાય છે. પરંતુ જે-તે ઘટક માટે શૈલીના નિયમની રચના કરવામાં આવે, ત્યારે આ બધા જ વિભાગો તેને લાગુ ન પડે એમ પણ બને. હવે, h1 સિલેક્ટર માટે શૈલીના નિયમોની રચના કરીએ.



આકૃતિ 2.4 : h1ના વિકલ્પો સાથે CSS stylesheets ડાયલોગબૉક્સ

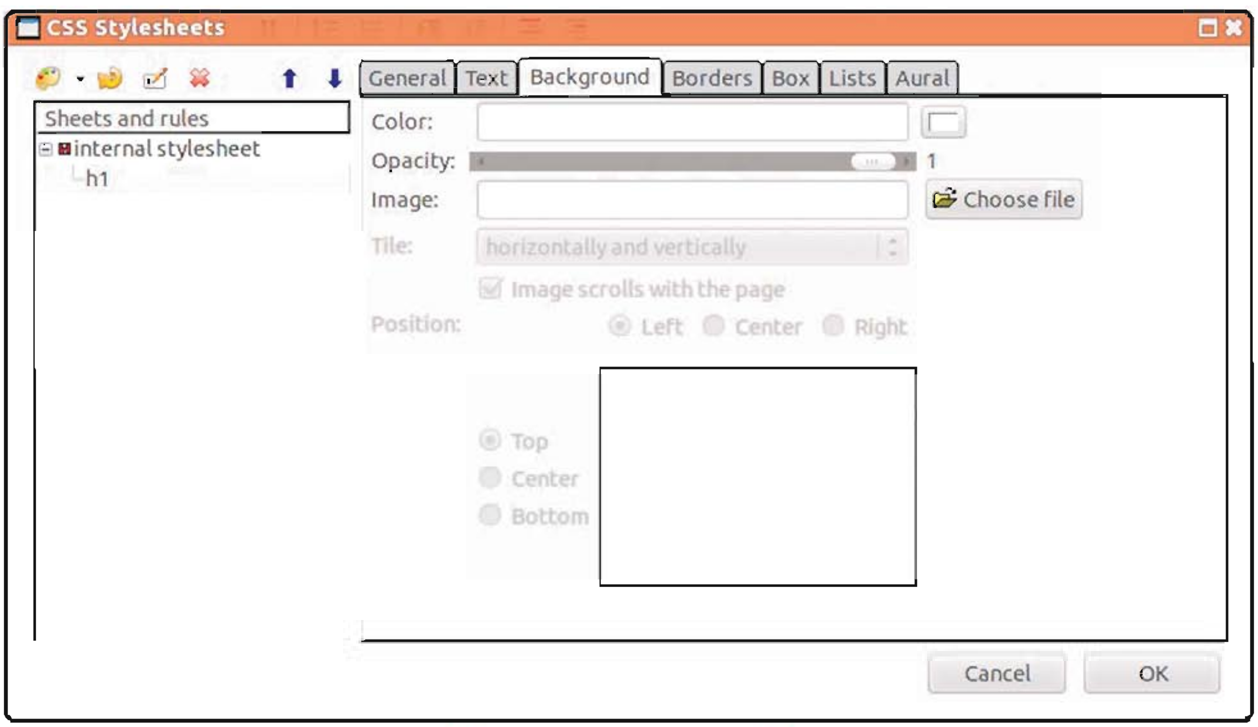
- ડાયલોગબૉક્સની ડાબી બાજુના વિભાગમાં આપેલ h1ને પસંદ કરો. આકૃતિ 2.5માં દર્શાવ્યા મુજબ Text વિભાગ પસંદ કરો. આમ કરવાથી Font family, Font size, Line height, Color, Case, Alignment જેવા વિવિધ વિકલ્પો દર્શાવવામાં આવશે.



આકૃતિ 2.5 : Text વિભાગ

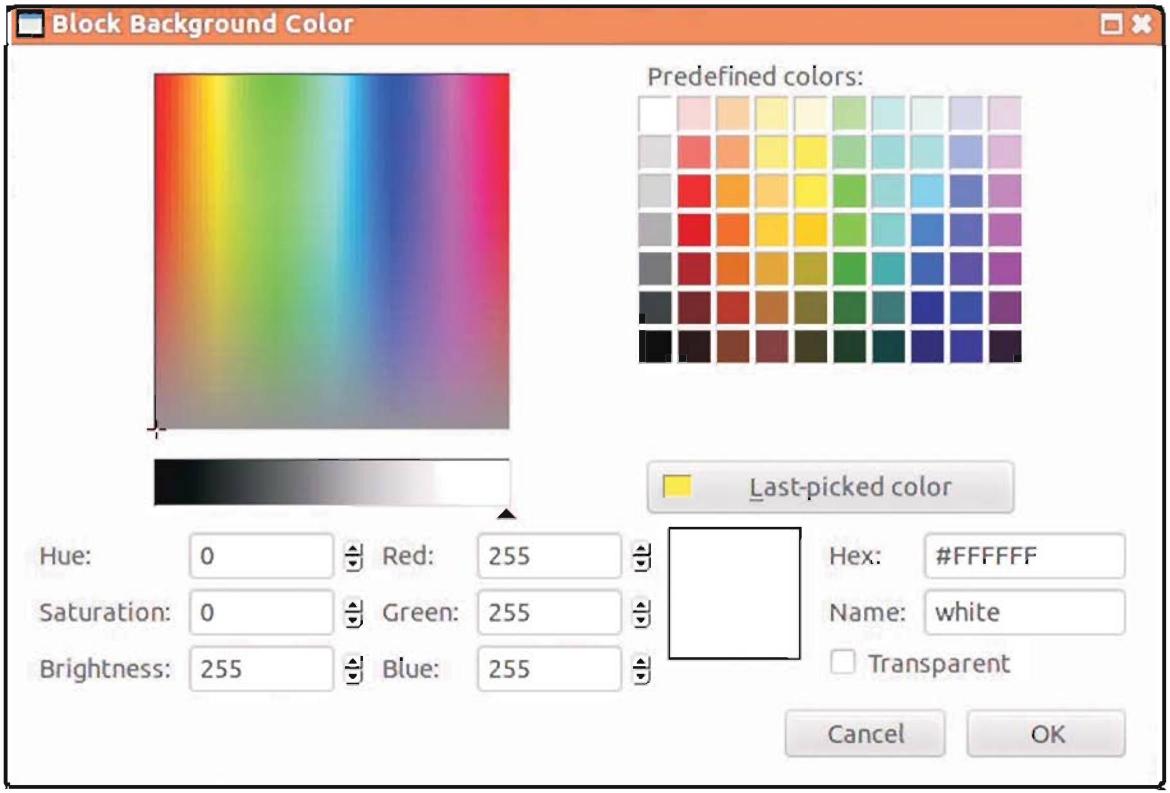
- h1 માટે જરૂરી શૈલી માટેના વિકલ્પોની પસંદગી કરીએ. Font family વિભાગમાં predefined રેડિયો-બટન પસંદ કરી ડ્રૉપડાઉન મેનુમાંથી "Times New Roman" વિકલ્પ પસંદ કરો. Case મેનુમાંથી "Upper Case" અને Alignment મેનુમાંથી "Center" વિકલ્પ પસંદ કરો. પસંદ કરેલા વિકલ્પો આકૃતિ 2.5માં જોઈ શકાય છે. વિકલ્પો બદલવાથી વિન્ડોમાં આવેલું પૂર્વનિર્ધારિત લખાણ પણ શૈલીને અનુરૂપ આપોઆપ બદલાઈ જાય છે તે જોઈ શકાશે.

- બૅકગ્રાઉન્ડ રંગ પસંદ કરવા માટે Background વિભાગ (tab) પસંદ કરો. આમ કરવાથી આકૃતિ 2.6માં દર્શાવ્યા મુજબ બૅકગ્રાઉન્ડ માટેના વિકલ્પો ખૂલશે.



આકૃતિ 2.6 : Background વિભાગ

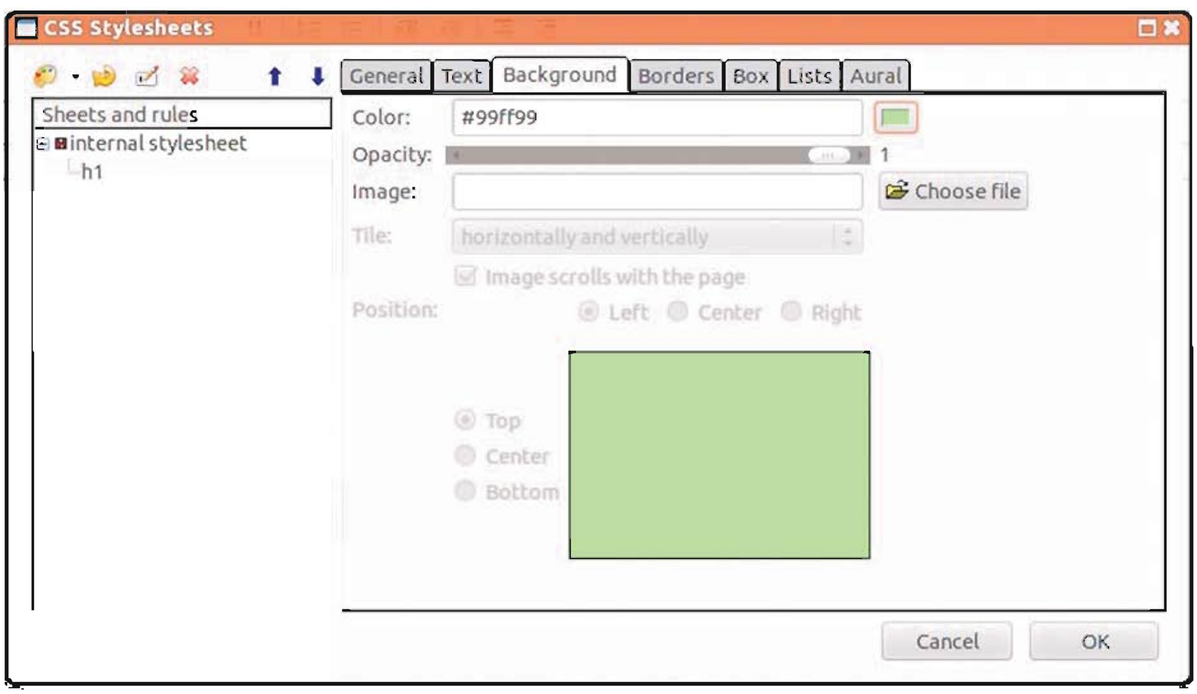
- Color વિકલ્પમાં આવેલ Color Palette બટન પર ક્લિક કરો. આકૃતિ 2.7માં દર્શાવ્યા મુજબ Block Background Color ડાયલોગબૉક્સ રજૂ કરવામાં આવશે. તમારો અપેક્ષિત રંગ પસંદ કરી OK બટન પર ક્લિક કરો.



આકૃતિ 2.7 : Block Background Color ડાયલોગબૉક્સ

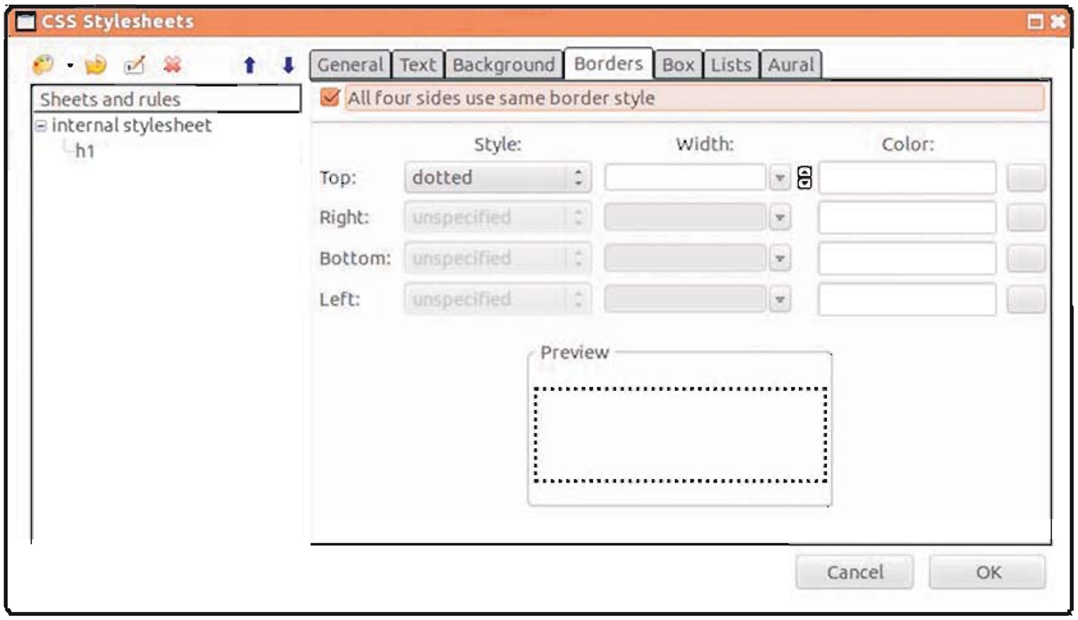
- રંગ પસંદ કર્યા પછી બૅકગ્રાઉન્ડ વિભાગનો દેખાવ આકૃતિ 2.8માં દર્શાવ્યો છે.

  નોંધ : જો બૅકગ્રાઉન્ડ તરીકે ચિત્રનો ઉપયોગ કરવા માંગતા હોઈએ, તો Image વિકલ્પમાં "Choose file" પર ક્લિક કરી બૅકગ્રાઉન્ડ માટેની ફાઇલ પસંદ કરવી.



આકૃતિ 2.8 : h1 માટે Background વિભાગમાં પસંદ કરવામાં આવેલા વિકલ્પો

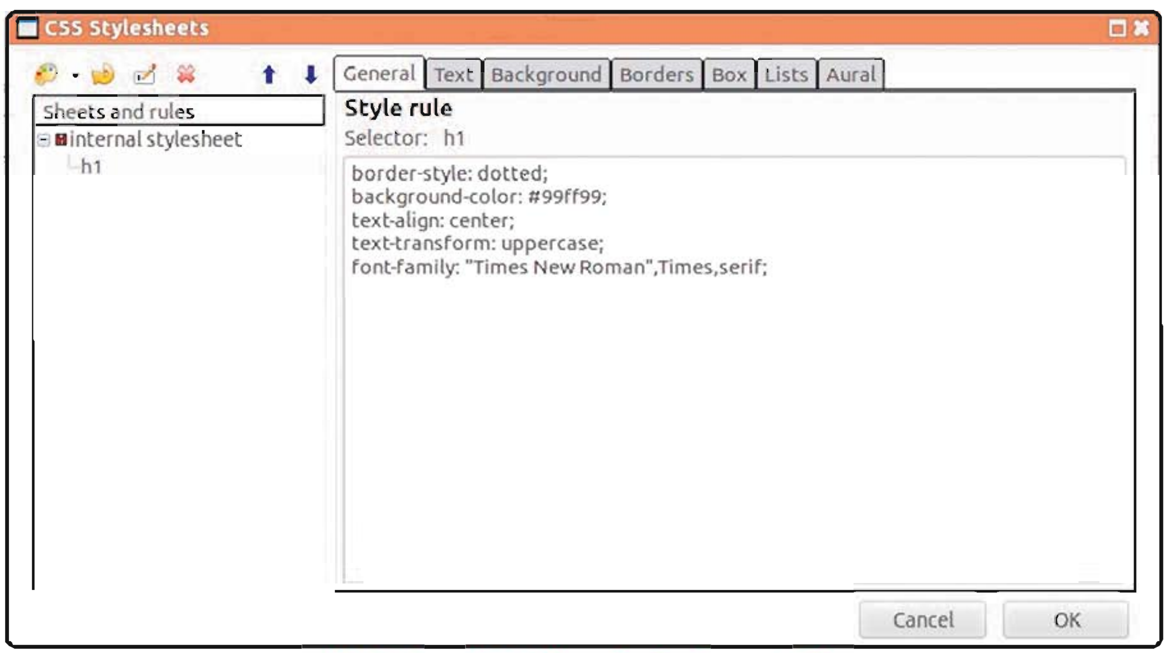
- સીમારેખા ઉમેરવા માટે Border વિભાગ પસંદ કરો. આકૃતિ 2.9માં દર્શાવ્યા મુજબ સીમારેખા માટેના વિકલ્પો દર્શાવવામાં આવશે. જો ચારેબાજુ પર એકસમાન સીમારેખા દર્શાવવી હોય, તો "All four sides use same border style" લખાણની આગળ આવેલું ચેકબોક્સ પસંદ કરો. હવે, માત્ર "Top" વિકલ્પ ઉપલબ્ધ રહેશે, અન્ય તમામ વિકલ્પો અનુપલબ્ધ બનશે. આમ થવાનું કારણ એ છે કે, એક જ બાજુને આપવામાં આવેલી અસર તમામ ચાર બાજુઓ પર લાગુ પાડવામાં આવશે. Style ડ્રોપડાઉન મેનુમાંથી Dotted વિકલ્પ પસંદ કરો. અહીં સીમારેખા માટે અપેક્ષિત જાડાઈ (width) અને રંગ (color) પણ પસંદ કરી શકાય છે. સીમારેખા માટે પસંદ કરેલ શૈલીનું પૂર્વદર્શન (preview) પણ કરી શકાય છે. OK બટન પર ક્લિક કરો.



આકૃતિ 2.9 : h1 માટે Border વિભાગમાં પસંદ કરવામાં આવેલા વિકલ્પ

CSS Stylesheet ડાયલોગબૉક્સમાં અન્ય વિભાગ પસંદ કરવામાં આવે, તો બીજા વિકલ્પ પણ ઉપલબ્ધ છે. જોકે, આપણા ઉદાહરણમાં h1ની શૈલી લાગુ પાડવા માટે માત્ર પસંદગીના વિકલ્પોની જરૂર છે.

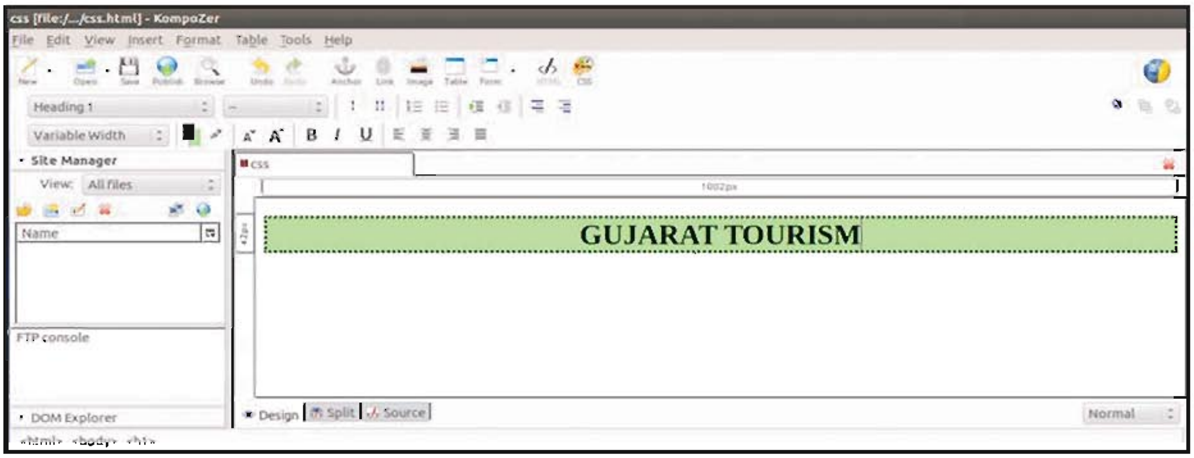
- General વિભાગ પર ક્લિક કરો. આકૃતિ 2.10માં દર્શાવ્યા મુજબ નિર્માણ કરવામાં આવેલ CSS કોડ દર્શાવવામાં આવશે. જો HTMLની જાણકારી હોય, તો આ કોડ સુધારી પણ શકાય છે.



આકૃતિ 2.10 : CSS કોડ દર્શાવતો General વિભાગ

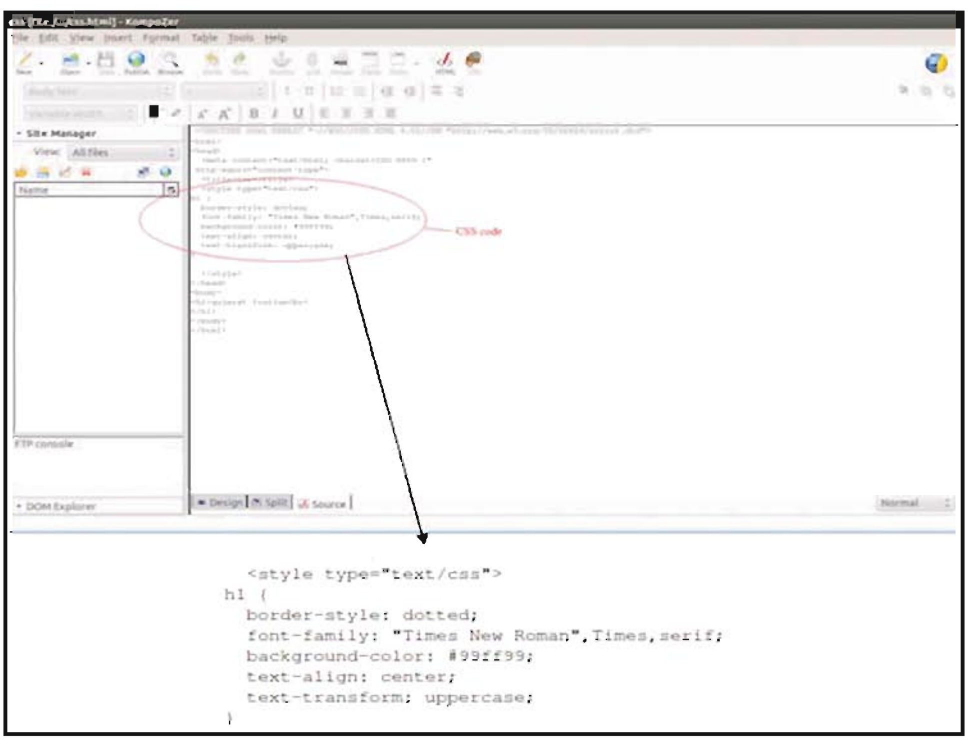
આમ, આપણે h1ને લાગુ પાડી શકાય, તેવી શૈલીની રચના કરી. વેબપેજના અન્ય કોઈ પણ ઘટકની શૈલીની રચના કરવા માટે આ જ પદ્ધતિનો ઉપયોગ કરી શકાય. અન્ય ઘટક ઉમેરવા માટે CSS Stylesheet ડાયલોગબૉક્સની ઉપર ડાબી બાજુના ખૂણા પર આવેલા CSS બટન ની તીરની નિશાની પર ક્લિક કરો. ડ્રૉપડાઉન મેનુમાંથી "Style-rule" વિકલ્પ પસંદ કરો. શૈલી માટેના નિયમ ઉમેર્યા બાદ OK બટન પર ક્લિક કરો.

હવે, h1 ટૅગ માટે રચવામાં આવેલ CSSનો ઉપયોગ વેબપેજમાં કરીએ. પાનાંમાં શીર્ષક ઉમેરવા માટે ફૉર્મેટ ટૂલબાર-1માંથી Heading1 વિકલ્પ પસંદ કરો. Heading1 વિકલ્પ પસંદ કરવામાં આવે, ત્યારે કર્સર આપોઆપ વેબપેજની મધ્યમાં ગોઠવાઈ જાય છે. (કારણકે આપણે CSSમાં Center ગોઠવણ લાગુ પાડી છે.) થોડું લખાણ ઉમેરો. લખાણ અપરકેસમાં અને Times New Roman ફૉન્ટ સાથે ઉમેરાશે. આકૃતિ 2.11 CSS લાગુ પાડવામાં આવી હોય તેવું Heading1 લખાણ દર્શાવે છે. માટે, હવે પછી કોઈ પણ પાનામાં Heading1 ઉમેરવામાં આવશે, તો તેના માટેની શૈલી દરેક વખતે સમાન રહેશે.



આકૃતિ 2.11 : વેબપેજમાં લાગુ પાડવામાં આવેલ CSS

CSS stylesheet કોડ જોવા માટે વિન્ડોની નીચેના ભાગમાં આવેલ Source વિભાગ પસંદ કરો. આકૃતિ 2.12 પાનાંના સૉર્સકોડમાં આવેલ શીર્ષક વિભાગનો CSS કોડ દર્શાવે છે.



આકૃતિ 2.12 : Source વ્યૂમાં CSS કોડ

CSSના ફાયદા (Advantages of CSS)

ઉપર્યુક્ત ચર્ચા બાદ કહી શકાય કે, CSS તેની નીચે આવેલા સ્તરમાં ફેરફાર કર્યા વગર વેબસાઇટની સંરચના કરવાની સુવિધા પૂરી પાડે છે. વેબસાઇટની ગોઠવણ અને સંરચના માટેના ગુણધર્મોને તેની નીચે આવેલા તેના તાર્કિક બંધારણથી અલગ રાખવામાં આવે, તો વેબસાઇટમાં ફેરફાર કરતી વખતે તેમાં આવેલી વિગતોમાં અકસ્માતે ફેરફાર થવાનો ભય રહેતો નથી.

ઇચ્છિત ઘટકની શૈલીમાં સુધારો કરવા માટે પ્રોગ્રામરે માત્ર CSS ફાઇલમાં ફેરફાર કરવાનો રહે છે, જે કાર્યને સરળ બનાવે છે. દરેક ઘટક માટેની શૈલી માત્ર એક વાર જ ઉમેરવાની હોવાથી, HTMLની સરખામણીએ CSSમાં ઓછો કોડ ઉમેરવો પડે છે. અને તેનાથી વેબપેજ ઝડપથી દર્શાવી શકાય છે. CSSનો ઉપયોગ વેબસાઇટની રચનાને ઝડપી અને કાર્યક્ષમ બનાવે છે.

CSSના ગેરફાયદા (Disadvantages of CSS)

જુદાં-જુદાં બ્રાઉઝર માટે CSSની સુસંગતતા બદલાતી રહે છે. આનો અર્થ એ થયો કે, સ્ટાઇલશીટની કેટલીક સુવિધાઓને દરેક બ્રાઉઝર સમર્થન આપતું નથી અને ઉપયોગકર્તાની અપેક્ષા અનુસાર તે શૈલીને દર્શાવી શકતું નથી. પરંતુ હવે, બ્રાઉઝરની અદ્યતન આવૃત્તિઓ વધુ પ્રમાણભૂત બની છે અને સુસંગતતાનો મુદ્દો ક્ષીણ થયેલો છે.

જાવાસ્ક્રિપ્ટ (JavaScript)

મૂળભૂત રીતે વેબપેજમાં દેખાવ (appearance)નું નિયંત્રણ કરવા માટે HTMLનો ઉપયોગ કરવામાં આવે છે. HTMLમાં બનાવવામાં આવેલા વેબપેજ સ્થિત (Static) છે અને બ્રાઉઝર રેન્ડર કરે પછી તેને બદલી શકાતાં નથી. બીજી તરફ, ઇન્ટરનેટના વિકાસ સાથે લોકો વધુ સારું સંવાદન અને દર્શનીય આલેખનની અપેક્ષા રાખતા થયા. પરંતુ HTML માત્ર અપરિવર્તનશીલ વેબપેજ આપી શકે છે. આમ, વેબ માટેની નવી પ્રોગ્રામિંગ ભાષામાં વધુ સંવાદનની જરૂર ઊભી થઈ. તેથી નેટસ્કેપે જાવાસ્ક્રિપ્ટનો વિકાસ કર્યો. જાવાસ્ક્રિપ્ટ એ વેબપેજમાં પ્રોગ્રામિંગનું પાસું ઉમેરવાની સુવિધા આપતી સ્ક્રિપ્ટિંગ ભાષા છે.

સ્ક્રિપ્ટિંગ ભાષા સરળ અને હળવી પ્રોગ્રામિંગ ભાષા છે, જેમાં 'સી' કે 'જાવા' ભાષા જેવાં અદ્યતન કાર્યોનો સમાવેશ કરવામાં આવ્યો હોતો નથી. વેબસાઇટના આલેખનમાં સુધારો કરવા તથા ફૉર્મની યથાર્થતા ચકાસવા માટે વેબપેજમાં જાવાસ્ક્રિપ્ટનો ઉપયોગ કરવામાં આવે છે. તે HTML પાનામાં સંવાદન ઉમેરે છે. અને તેને HTML કોડમાં પ્રત્યક્ષ રૂપે ઉમેરવામાં આવે છે. હાલમાં મોઝિલા ફાયરફોક્સ, ક્રોમ, સફારી અને ઇન્ટરનેટ એક્સપ્લોરર જેવાં તમામ વેબબ્રાઉઝર જાવાસ્ક્રિપ્ટને સમર્થન આપે છે. જાવાસ્ક્રિપ્ટના ઉપયોગ પછી વેબપેજ અપરિવર્તનશીલ રહેતું નથી, પરંતુ ઉપયોગકર્તા સાથેના સંવાદન, બ્રાઉઝરનું નિયંત્રણ અને HTML વિગતોની ગતિશીલ (dynamic) રજૂઆત માટેના કોડનો સમાવેશ કરી શકે છે.

કમ્પોઝરની મદદથી ફૉર્મ કેવી રીતે બનાવી શકાય તે આપણે જોયું. જ્યારે ઉપયોગકર્તા ફૉર્મમાં વિગતો ઉમેરે ત્યારે, તે કેટલાંક મહત્ત્વનાં ફિલ્ડને ખાલી છોડી દે અથવા ફિલ્ડમાં વિગતો ખોટા સ્વરૂપમાં ઉમેરે તેવું પણ બને. આવા કિસ્સામાં કેટલીક યથાર્થતા પૂરી પાડવી જરૂરી બને છે. યથાર્થતા ઉપયોગકર્તાને ભૂલ કરતા અટકાવે છે. જ્યારે ઉપયોગકર્તા ખોટી વિગત ઉમેરે અથવા ફિલ્ડને ખાલી છોડી દે ત્યારે ભૂલનો સંદેશ દર્શાવવો જોઈએ અને ફૉર્મને સબમિટ થતાં રોકવું જોઈએ.

હાલમાં, જાવાસ્ક્રિપ્ટના સૌથી સામાન્ય વિનિયોગ ક્લાયન્ટ બાજુની સ્ક્રિપ્ટ છે, જે વેબબ્રાઉઝરમાં ચલાવવામાં આવે છે. HTML ફૉર્મમાં ઉમેરેલી વિગતોને સર્વર તરફ મોકલવામાં આવે તે પહેલાં ક્લાયન્ટ બાજુ તેની યથાર્થતા ચકાસવા માટે જાવાસ્ક્રિપ્ટનો ઉપયોગ કરવામાં આવે છે. જાવાસ્ક્રિપ્ટનો ઉપયોગ કરી સામાન્ય રીતે ફૉર્મમાં નીચેની બાબતો તપાસવામાં આવે છે :

- ઉપયોગકર્તાએ કોઈ જરૂરી ફિલ્ડ ખાલી છોડી દીધું છે ?
- ઉપયોગકર્તાએ યોગ્ય ઇ-મેઇલ સરનામું દાખલ કર્યું છે ?

- બે ફિલ્ડની વિગતો સમાન છે કે નહીં ?
- ઉપયોગકર્તાએ યોગ્ય તારીખ દાખલ કરી છે ?
- ઉપયોગકર્તાએ આંકડાકીય ફિલ્ડમાં લખાણ ઉમેર્યું છે ? ઉદાહરણ તરીકે, જથ્થા (quantity) માટેના ફિલ્ડમાં આંકડાકીય વિગતને બદલે ઉપયોગકર્તા શાબ્દિક વિગત દાખલ કરે.

ફૉર્મની યથાર્થતા કેવી રીતે ચકાસવી, તેનો અભ્યાસ કરતાં પહેલાં જાવાસ્ક્રિપ્ટ કોડ કેવી રીતે લખવો તે શીખીએ.

જાવાસ્ક્રિપ્ટ કોડનો અમલ HTML વેબપેજમાં જ કરવામાં આવે છે. આમ, જાવાસ્ક્રિપ્ટ કોડને વેબપેજમાં સીધા જ એક સ્વતંત્ર વિભાગ સ્વરૂપે મૂકી શકાય છે. HTML પાનામાં <script>...</script> ટેગનો ઉપયોગ કરી જાવાસ્ક્રિપ્ટનો કોડ ઉમેરવામાં આવે છે. <script> અને </script> ટેગની વચ્ચે આવેલી લીટીઓ જાવાસ્ક્રિપ્ટનો કોડ ધરાવે છે. યાદ રાખો કે, જાવાસ્ક્રિપ્ટ એ 'કેસ-સેન્સિટિવ' ભાષા છે.

જાવાસ્ક્રિપ્ટને HTMLના <body> અથવા <head> વિભાગમાં સમાવવામાં આવે છે, પરંતુ કોડને <head> ટેગમાં ઉમેરવામાં આવે તે વધુ ઇચ્છનીય છે. <script> ટેગબ્રાઉઝરને તેની વચ્ચે રહેલાં તમામ લખાણોને એક સ્ક્રિપ્ટ તરીકે અર્થઘટન કરવાનો નિર્દેશ કરે છે. કમ્પોઝરના સૉર્સવ્યૂનો ઉપયોગ કરી એક સરળ જાવાસ્ક્રિપ્ટ કોડ લખીએ. આકૃતિ 2.13 જાવાસ્ક્રિપ્ટ સાથેનો સૉર્સવ્યૂ દર્શાવે છે. ફાઇલનો સંગ્રહ કરી તેને વેબબ્રાઉઝરમાં ખોલો.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
  <meta content="text/html; charset=ISO-8859-1"
 http-equiv="content-type">
  <title>js1</title>
  <script>
document.write("HELLO STUDENTS");
  </script>
</head>
<body>
<br>
</body>
</html>
```

આકૃતિ 2.13 : સૉર્સવ્યૂમાં જાવાસ્ક્રિપ્ટ

અહીં જોઈ શકાય છે કે, <script> ટેગમાં માત્ર એક વિધાન "document.write("HELLO STUDENTS")" લખવામાં આવ્યું છે. અહીં write એક પ્રક્રિયા (method) છે, જે document ઑબ્જેક્ટનો એક ભાગ છે. જે લખાણ દર્શાવવાનું હોય, તેને document.write() પ્રક્રિયાના ચલ તરીકે આપવામાં આવે છે.

આકૃતિ 2.13માં વિધાનના અંતમાં આપવામાં આવેલું અર્ધવિરામ ચિહ્ન (semicolon) જાવાસ્ક્રિપ્ટમાં વૈકલ્પિક છે. અર્ધવિરામ ચિહ્નનો ઉપયોગ સમાપ્તિ (termination) ને બદલે અલગતા (separation) માટે થાય છે. તેથી જો દરેક વિધાન નવી લીટીમાં લખવામાં આવે, તો અર્ધવિરામ મૂકવાની જરૂર નથી. આપણા ઉદાહરણમાં document.write વિધાન પછીનું અર્ધવિરામ કાઢી નાખવામાં આવશે તોપણ પરિણામ સમાન રહેશે. આમ છતાં, અર્ધવિરામનો ઉપયોગ એક સારી પ્રોગ્રામિંગ પ્રથા છે. આકૃતિ 2.14માં કોડના પરિણામને બ્રાઉઝરમાં દર્શાવ્યું છે.

આકૃતિ 2.14 : બ્રાઉઝરમાં પરિણામ

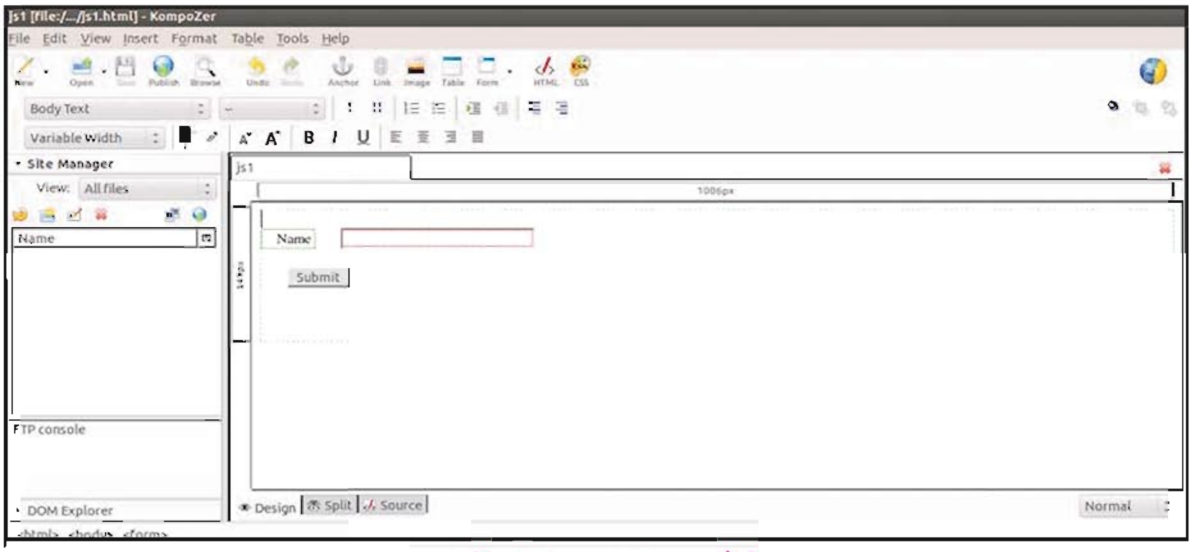
નોંધ : જાવાસ્ક્રિપ્ટ વિધાનોને બ્લૉક સ્વરૂપે એકત્રિત કરી શકાય છે. બ્લૉકની શરૂઆત અને અંત છગડિયા કૌંસ ({ }) દ્વારા કરવામાં આવે છે.

હવે, ફૉર્મની યથાર્થતા માટેની સરળ જાવાસ્ક્રિપ્ટ લખવાનું શીખીએ. પ્રથમ, આપણે બે ફિલ્ડ સાથેનું એક સરળ ફૉર્મ બનાવીએ : name ફિલ્ડ અને submit બટન.

ફૉર્મની યથાર્થતા માટે, જો ઉપયોગકર્તા name ફિલ્ડને ખાલી રાખી submit બટન પર ક્લિક કરે, તો ઉપયોગકર્તા સમક્ષ નામ ઉમેરવા માટેનો સંદેશ પ્રદર્શિત થવો જોઈએ. આ ક્રિયા માટે નીચે જણાવેલ પગલાંને અનુસરો :

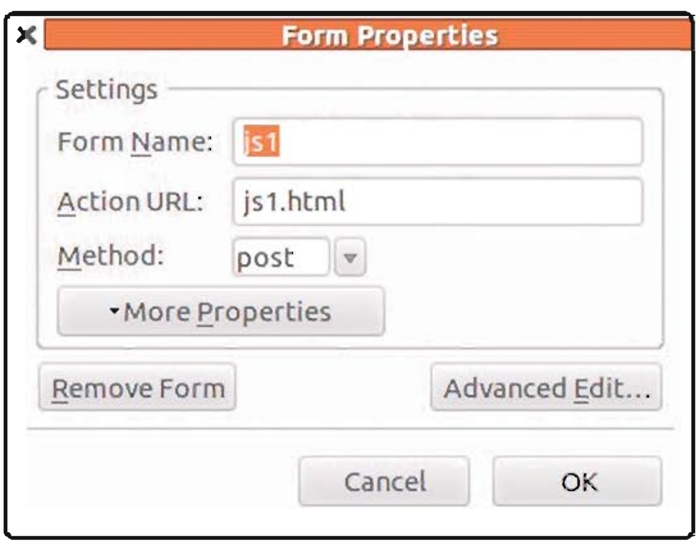
- નવી ફાઇલ બનાવી તેનો સંગ્રહ કરો. આ ઉદાહરણમાં આપણે ફાઇલને js1.html નામ આપી સંગ્રહ કર્યો છે.
- **Form → Define Form** વિકલ્પનો ઉપયોગ કરી નવા ફૉર્મની રચના કરો. ફૉર્મને નામ આપો. આપણા ઉદાહરણમાં ફૉર્મને js1 નામ આપ્યું છે. આકૃતિ 2.15માં દર્શાવ્યા મુજબ નામ માટેનું એક લેબલ અને એક ઇનપુટ ટૅક્સ્ટફિલ્ડ ઉમેરો. એક સબમિટ બટન પણ ઉમેરો.

  નોંધ : ફૉર્મ બનાવતી વખતે આપવામાં આવેલ ફૉર્મ, ઇનપુટ ફિલ્ડ અને કોઈ પણ અન્ય ફિલ્ડનાં નામ સરળ અને યાદ રહે તેવાં હોવાં જોઈએ, કારણકે સ્ક્રિપ્ટમાં વારંવાર તેનો સંદર્ભ આપવાની જરૂર પડશે.



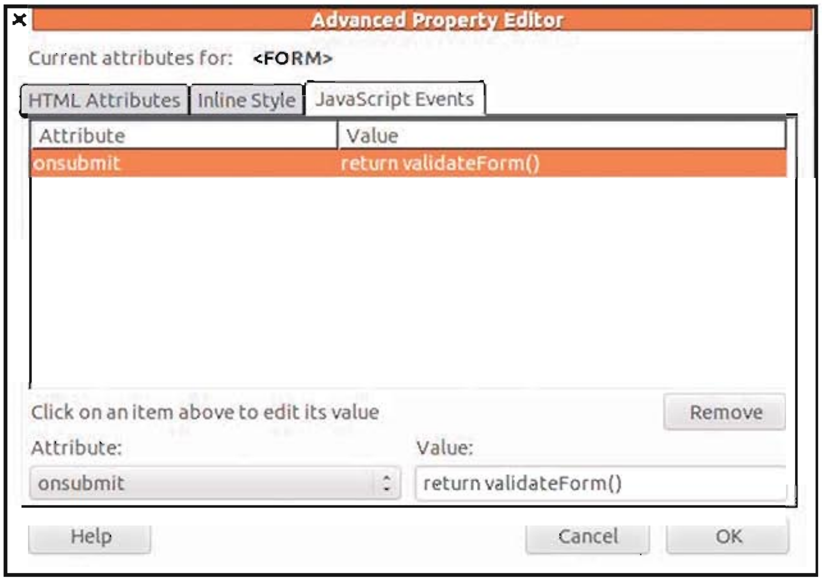
આકૃતિ 2.15 : નમૂનારૂપ ફૉર્મ

- **Form → Define Form** પસંદ કરો. તેથી આકૃતિ 2.16માં દર્શાવવામાં આવેલ Form Properties ડાયલોગબૉક્સ ખૂલશે.



આકૃતિ 2.16 : Form Properties ડાયલોગબૉક્સ

- Advanced Edit બટન પર ક્લિક કરો. તેનાથી આકૃતિ 2.17માં દર્શાવેલ Advanced Property Editor ડાયલોગબૉક્સ ખૂલશે. JavaScript Events વિભાગ પસંદ કરો. આકૃતિ 2.17માં દર્શાવ્યા પ્રમાણે Attribute વિકલ્પમાં "onsubmit" વિકલ્પ પસંદ કરો અને Valueમાં "return validateForm()" લખો. OK બટન પર ક્લિક કરો.



આકૃતિ 2.17 : Advanced Property Editor ડાયલોગબૉક્સ

- હવે, સૉર્સવ્યૂ ખોલો. આકૃતિ 2.18માં દર્શાવ્યા મુજબ <head> ટૅગની અંદર સ્ક્રિપ્ટ ઉમેરો.

આકૃતિ 2.18 : સૉર્સવ્યૂમાં જાવાસ્ક્રિપ્ટ

આકૃતિ 2.8માં દર્શાવેલ સ્ક્રિપ્ટમાં નીચેના ઘટકો વ્યાખ્યાયિત કરવામાં આવ્યા છે :

- validateForm() વિધેય
- *x* ચલ
- if શરતી વિધાન
- alert() આંતરપ્રસ્થાપિત વિધેય
- document ઑબ્જેક્ટ
- focus પદ્ધતિ

હવે, દરેક ઘટકની સંક્ષેપમાં ચર્ચા કરીએ.

ValidateForm વિધેય (Function ValidateForm())

નિશ્ચિત કાર્યને પાર પાડતા પુનઃઉપયોગમાં લઈ શકાય તેવા કોડના બ્લૉકને વિધેય (અથવા રૂટિન) કહે છે. વિધેયને function કી-વર્ડ વડે વ્યાખ્યાયિત કરવામાં આવે છે અને કોડના બ્લૉકને છગડિયા કૌંસ "{ }"માં લખવામાં આવે છે. અહીં validateForm() એક વિધેય છે.

જાવાસ્ક્રિપ્ટ alert() જેવાં કેટલાંક આંતરપ્રસ્થાપિત વિધેય પૂરાં પાડે છે. alert() વિધેય લખાણ સ્વીકારે છે અને તેને એલર્ટ બૉક્સમાં દર્શાવે છે.

ઉદાહરણ તરીકે,

```
function hello()
{
  alert("Hello Students");
}
```

જ્યારે "hello()" વિધેય બોલાવવામાં આવે (called), ત્યારે Hello Students સંદેશ સાથેનું એલર્ટ બૉક્સ દર્શાવવામાં આવશે.

વિધેયનો અમલ કોઈ ઘટના (event) દ્વારા અથવા સૉર્સકોડમાંથી બોલાવવામાં આવે (called) ત્યારે કરવામાં આવે છે. વિધેયને સૉર્સકોડમાં કોઈ પણ સ્થાનેથી બોલાવી શકાય છે. HTML કોડના head કે body વિભાગમાં વિધેયને વ્યાખ્યાયિત કરી શકાય છે. વિધેયને તેનાં નામથી બોલાવવામાં આવે છે. તે કૌંસમાં આપેલી કિંમતો સ્વીકારી શકે છે. આ કિંમતો વિધેયને પ્રાચલ (parameter) તરીકે મોકલવામાં આવે છે. વિધેય કિંમત પરત કરે, તેવું પણ આપણે ક્યારેક ઇચ્છતા હોઈએ. return વિધાનની મદદથી આ શક્ય બને છે. return વિધાનનો ઉપયોગ વિધેયનો અમલ અટકાવી કોઈ નિશ્ચિત કિંમત પરત કરે છે.

આપણા ઉદાહરણમાં validateForm( ) વિધેયને સબમિટ બટન પર ક્લિક આપવાથી ઉદ્ભવતી ઘટના દ્વારા બોલાવવામાં (call) આવે છે. આમ, જ્યારે ઉપયોગકર્તા સબમિટ બટન પર ક્લિક કરશે, જેને ઘટના (event) તરીકે ઓળખવામાં આવે છે, ત્યારે વિધેયને બોલાવી તેમાં આવેલાં વિધાનોનો અમલ કરવામાં આવશે. વિધેય ખોટી (false) કિંમત પરત કરશે.

ઘટના (Events)

ઉપયોગકર્તા દ્વારા કરવામાં આવતી પ્રક્રિયાનો પ્રતિભાવ આપે તેવા સંવાદિત વેબપેજની રચના કરવા માટે જાવાસ્ક્રિપ્ટ ઉપયોગી છે. ઉપયોગકર્તા અને વેબપેજ વચ્ચેના સંવાદન દ્વારા ઘટના (Event) ઉત્પન્ન થાય છે. બીજા શબ્દોમાં કહીએ તો, જ્યારે ઉપયોગકર્તા કંઈ પણ કાર્ય કરે, ત્યારે ઘટના ઉદ્ભવે છે. કોષ્ટક 2.1માં જાવાસ્ક્રિપ્ટ ઘટનાઓની યાદી આપવામાં આવી છે.

ઘટના	સમજૂતી
abort	ચિત્ર દર્શાવવાનું રદ કરવામાં આવે, ત્યારે
blur	રેડિયો-બટન જેવું કોઈ ઘટક નિષ્ક્રિય બને, ત્યારે
click	ઉપયોગકર્તા ફૉર્મના કોઈ ઘટક પર ક્લિક કરે, ત્યારે
change	ઉપયોગકર્તા દ્વારા ફૉર્મના ફિલ્ડની કિંમત બદલવામાં આવે, ત્યારે
error	દસ્તાવેજ કે ચિત્ર દર્શાવતી વખતે ક્ષતિ ઉદ્‌ભવે, ત્યારે
focus	બટન જેવો કોઈ ઘટક સક્રિય બને, ત્યારે
load	દસ્તાવેજ કે ચિત્ર દર્શાવવામાં આવે, ત્યારે
mouseout	માઉસનું પૉઇન્ટર ઘટક પરથી દૂર કરવામાં આવે, ત્યારે
mouseover	માઉસનું પૉઇન્ટર ઘટક પર લાવવામાં આવે, ત્યારે
reset	ફૉર્મના ફિલ્ડને પૂર્વનિર્ધારિત કિંમતો સાથે રીસેટ કરવામાં આવે, ત્યારે
select	ઉપયોગકર્તા ફૉર્મનું ફિલ્ડ પસંદ કરે, ત્યારે
submit	ઉપયોગકર્તા ફૉર્મ સબમિટ કરે, ત્યારે
unload	ઉપયોગકર્તા પાનું છોડી દે, ત્યારે

કોષ્ટક 2.1 : જાવાસ્ક્રિપ્ટ ઘટનાઓ

જ્યારે કોઈ ઘટના ઉદ્‌ભવે છે ત્યારે આપેલ સ્થિતિ અનુસાર એક નિશ્ચિત જાવાસ્ક્રિપ્ટ કોડનો અમલ કરવામાં આવે છે. આ જાવાસ્ક્રિપ્ટ કોડને ઘટના-સંચાલક (event handler) કહે છે. ઇવેન્ટ હૅન્ડલરનું નામ ઘટનાનાં નામ જેવું જ રાખવામાં આવે છે. ઉદાહરણ તરીકે, 'ક્લિક' ઘટના માટે ઇવેન્ટ હૅન્ડલરનું નામ onclick() છે. એ જ રીતે 'સબમિટ' ઘટના માટે onsubmit() ઇવેન્ટ હૅન્ડલરનો ઉપયોગ કરવામાં આવશે.

આપણા ઉદાહરણમાં ઉપયોગકર્તા સબમિટ બટન પર ક્લિક કરે ત્યારે, એટલે કે સબમિટ ઘટના વખતે ફૉર્મની યથાર્થતા ચકાસવાની જરૂર છે. આ માટે આપણે onsubmit ઇવેન્ટ હૅન્ડલરનો ઉપયોગ કરવાની જરૂર છે. આકૃતિ 2.18માં onsumbit() ઇવેન્ટ હૅન્ડલર ઉમેરવામાં આવ્યું છે. આ ઇવેન્ટ હૅન્ડલર validateForm વિધેયને બોલાવશે. 'name' ફિલ્ડમાં કોઈ કિંમત છે કે તે ખાલી (null) છે તેના આધારે True કે False પરત કરશે.

ચલ (Variable)

આપણા ઉદાહરણમાં, validateForm() વિધેયમાં નીચેનું વિધાન લખ્યું છે :

```
var x=document.js1.name.value
```

અહીં આપણે x નામનો ચલ ઘોષિત કર્યો છે. ચલ એ વિગતોનો સંગ્રહ કરતો સંગ્રાહક છે. યાદ રાખો કે, જાવાસ્ક્રિપ્ટમાં ચલ પણ કેસ-સૅન્સિટિવ છે. માટે ચલ "x" અને ચલ "X" સમાન નથી. ચલમાં અંકો, સ્ટ્રિંગ કે લખાણનો સંગ્રહ કરી શકાય છે. જાવાસ્ક્રિપ્ટમાં var કી-વર્ડનો ઉપયોગ કરી ચલની ઘોષણા કરી શકાય છે. ઉદાહરણ તરીકે,

```
var x=3;
var y="Hello";
var z="Hello Students";
```

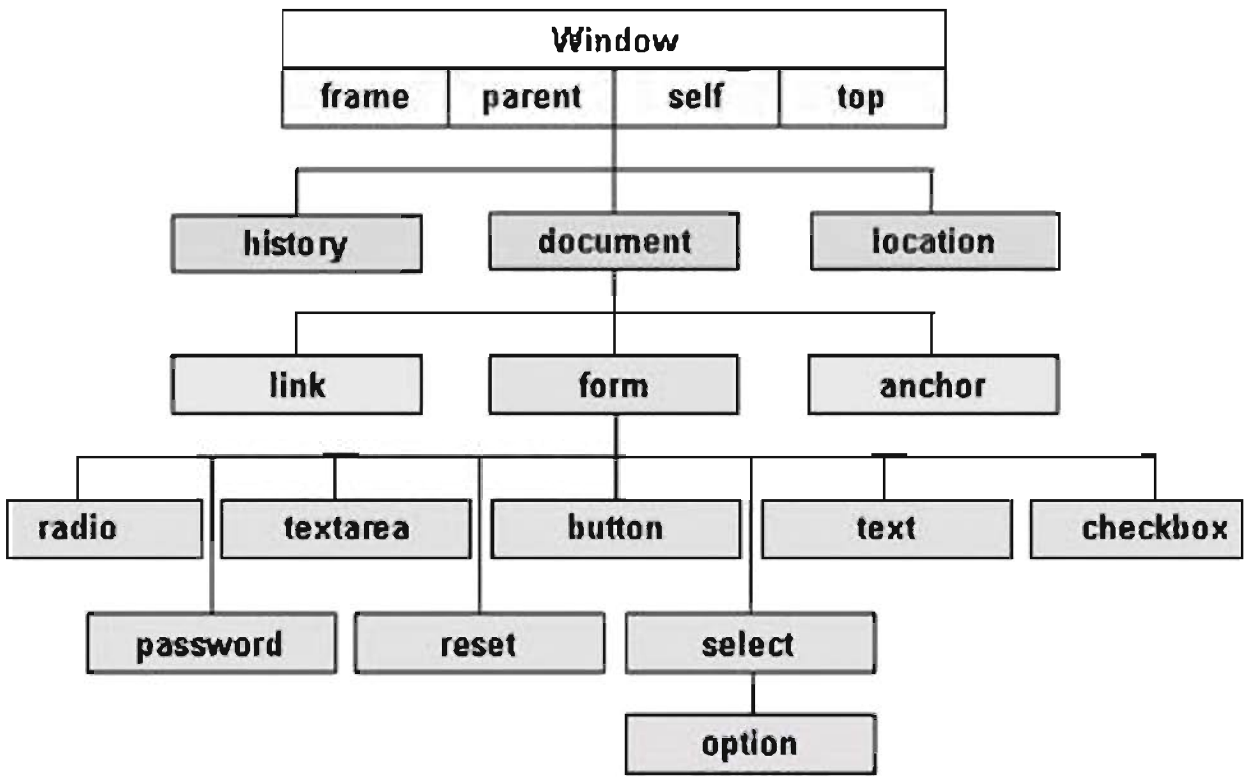
If વિધાન (If Statement)

તમે શરતી વિધાન ifથી સુપરિચિત છો જ. પ્રોગ્રામનો પ્રવાહ બદલવા માટે શરતી વિધાન ifનો ઉપયોગ કરવામાં આવે છે. કોઈ નિશ્ચિત શરતની યથાર્થતા તપાસવા માટે if વિધાન પદાવલીનું મૂલ્યાંકન કરે છે. જો શરત સાચી હોય, તો પ્રોગ્રામ if વિધાનના બ્લૉકમાં પ્રવેશી તેમાં આવેલાં વિધાનોનો અમલ કરે છે. આપણા ઉદાહરણમાં x ચલની કિંમત ખાલી (null) છે કે નહીં તે ચકાસવા માટે if વિધાનનો ઉપયોગ કર્યો છે. જો ચલ xની કિંમત ખાલી હશે, તો if બ્લૉકનો અમલ કરવામાં આવશે.

ડૉક્યુમેન્ટ ઑબ્જેક્ટ મૉડેલ (Document Object Model) - document.js1.name.value

કેટલીક વાર, વેબબ્રાઉઝરના નિયંત્રણ માટે જાવાસ્ક્રિપ્ટનો ઉપયોગ કરવામાં આવે છે. ઉદાહરણ તરીકે, ધારો કે તમારે વેબપેજને બદલવાની કે તેમાં આવેલા ઘટકોનું નિયંત્રણ કરવાની જરૂર છે. વેબ બ્રાઉઝરની વિન્ડો કે વેબપેજના નિયંત્રણ માટે બ્રાઉઝર ઑબ્જેક્ટ મૉડેલ (Browser Object Model - BOM)નો ઉપયોગ કરવામાં આવે છે.

તમામ બ્રાઉઝરને જુદા જુદા વિભાગોમાં અથવા ઘટકોમાં વિભાજિત કરવામાં આવે છે, જેને જાવાસ્ક્રિપ્ટ દ્વારા ઉપયોગમાં લઈ શકાય છે. આ વિભાગોને બ્રાઉઝર ઑબ્જેક્ટ મૉડેલ અથવા BOM તરીકે ઓળખવામાં આવે છે. આકૃતિ 2.19માં દર્શાવ્યા મુજબ આ બ્રાઉઝરના પદાનુક્રમમાં સૌથી ઉપર વિન્ડો ઑબ્જેક્ટ હોય છે.



આકૃતિ 2.19 : બ્રાઉઝર ઑબ્જેક્ટનો પદાનુક્રમ

અહીં ટૂલબાર, મેનુ, સ્ટેટસબાર, વેબપેજ અને અન્ય ઘણા ઘટકો સાથેનો સમગ્ર બ્રાઉઝરનો પદાનુક્રમ રજૂ કરેલો છે. બ્રાઉઝરમાં દેખાતી વિન્ડો કે તેમાં આવેલા ઘટકોને બદલવા માટે બ્રાઉઝર ઑબ્જેક્ટ મૉડેલમાં આવેલા ગુણધર્મો (properties) અને પદ્ધતિઓ (methods)નો ઉપયોગ કરવામાં આવે છે. બ્રાઉઝર ઑબ્જેક્ટ મૉડેલમાં ઑબ્જેક્ટ બનાવવાની જરૂર પડતી નથી; પરંતુ જ્યારે બ્રાઉઝર વેબપેજ દર્શાવે, ત્યારે તેની રચના આપોઆપ થતી હોય છે.

બ્રાઉઝર ઑબ્જેક્ટ મૉડેલમાં સૌથી ઉપરના સ્તરનો ઑબ્જેક્ટ Window છે. Window ઑબ્જેક્ટ બ્રાઉઝરની વિન્ડો અથવા તેમાં આવેલી સ્વતંત્ર ફ્રેમને રજૂ કરે છે. તેની રચના બ્રાઉઝર દ્વારા આપોઆપ કરવામાં આવે છે. Window ઑબ્જેક્ટ વૈશ્વિક (global) ઑબ્જેક્ટ ગણાય છે. કારણકે, મૉડેલના અન્ય તમામ બ્રાઉઝર ઑબ્જેક્ટનો તેની અંદર સમાવેશ કરવામાં આવે છે. ઉદાહરણ તરીકે, Window ઑબ્જેક્ટમાં document ઑબ્જેક્ટનો સમાવેશ થાય છે. Window ઑબ્જેક્ટની પદ્ધતિઓ (methods) અને ગુણધર્મો (properties) દ્વારા વેબબ્રાઉઝરની વિન્ડોને નિયંત્રિત કરવામાં આવે છે, જ્યારે document ઑબ્જેક્ટની પદ્ધતિઓ (methods) અને ગુણધર્મો (properties) દ્વારા વેબપેજને નિયંત્રિત કરવામાં આવે છે.

બ્રાઉઝર ઑબ્જેક્ટ મૉડેલમાં સૌથી અગત્યનો ઑબ્જેક્ટ document ઑબ્જેક્ટ છે. બ્રાઉઝરમાં વેબપેજ રજૂ કરવા માટે તેનો ઉપયોગ કરવામાં આવે છે. વેબપેજના અન્ય તમામ ઘટકો જેવા કે, ફૉર્મ, ચિત્રો, લિંક અને અન્ય ઘટકો document ઑબ્જેક્ટની અંદર સમાવવામાં આવ્યા છે. ઉદાહરણ તરીકે, <form> ઘટક દ્વારા રચવામાં આવેલો form ઑબ્જેક્ટ જાવાસ્ક્રિપ્ટ દ્વારા document ઑબ્જેક્ટની અંદર રજૂ કરવામાં આવે છે. document ઑબ્જેક્ટમાં form ઑબ્જેક્ટનો સમાવેશ કરવામાં આવ્યો છે. તેવી જ રીતે form ઑબ્જેક્ટમાં element ઑબ્જેક્ટને સમાવવામાં આવ્યો છે. ફૉર્મના દરેક ઘટકનો સંદર્ભ મેળવવા element ઑબ્જેક્ટનો ઉપયોગ કરવામાં આવે છે. element ઑબ્જેક્ટ તરીકે રેડિયો-બટન, ટેક્સ્ટબૉક્સ, ચેકબૉક્સ કે અન્ય કોઈ પણ ઘટક હોઈ શકે છે.

focus પદ્ધતિ (focus Method)

સ્ક્રિપ્ટમાં focus નામની પદ્ધતિનો પણ ઉપયોગ કરવામાં આવે છે. ફૉર્મના નિશ્ચિત ઘટક પર નિયંત્રણ લઈ જવા તેને લાગુ પાડવામાં આવે છે. આમ, ઉપયોગકર્તા એલર્ટ સંદેશ મેળવે પછી, focus પદ્ધતિની મદદથી ઘટકની અંદર કર્સર ગોઠવવામાં આવે છે અને તે નિશ્ચિત ઘટક પ્રકાશિત (highlighted) બને છે. અહીં નિયંત્રણને name ઘટક પર લઈ જવામાં આવ્યું છે.

હવે, ફાઇલને બ્રાઉઝરમાં ખોલીએ. આકૃતિ 2.20માં પરિણામ દર્શાવ્યું છે.

આકૃતિ 2.20 : બ્રાઉઝરમાં પરિણામ

name ફિલ્ડને ખાલી રાખી submit બટન પર ક્લિક કરો. આકૃતિ 2.21માં દર્શાવ્યા પ્રમાણે એક એલર્ટ સંદેશ રજૂ કરવામાં આવશે.

આકૃતિ 2.21 : ખાલી name ફિલ્ડ માટેનો એલર્ટ સંદેશ

alertનો સંદેશ એ સોર્સવ્યૂમાં લખવામાં આવેલ સ્ક્રિપ્ટનું પરિણામ છે. આ એવી યથાર્થતા છે, જે ફૉર્મને જાવાસ્ક્રિપ્ટ દ્વારા

પૂરી પાડવામાં આવી છે, આમ, અનેક ફિલ્ડ ધરાવતા ફૉર્મમાં કોઈ પણ ફિલ્ડ ખાલી રાખવામાં આવે, તો તેને ચકાસવા માટે યથાર્થતા ઉમેરી શકાય છે.

ઇન્ટરનેટ પર ફૉર્મ ભરતી વખતે તમે કેટલાંક ફિલ્ડને લાલ રંગની ફૂદડી "*" સાથે દર્શાવેલાં જરૂર જોયાં હશે. તે ઉપયોગકર્તાને નિર્દેશ કરે છે કે આ ફિલ્ડ ફરજિયાત છે અને તેને ખાલી છોડવાં જોઈએ નહીં. જાવાસ્ક્રિપ્ટનો ઉપયોગ કરી આ પ્રકારનાં ફિલ્ડની વિગતો તપાસી શકાય છે તથા ઉપયોગકર્તા સમક્ષ એલર્ટ સંદેશ દર્શાવી શકાય છે.

હવે, ફૉર્મમાં Pincode નામનું અન્ય ફિલ્ડ ઉમેરીએ. આ ફિલ્ડ પર નીચે જણાવેલ યથાર્થતા લાગુ પાડવામાં આવશે :

- ઉપયોગકર્તા આ ફિલ્ડને ખાલી છોડી શકશે નહીં.
- માત્ર અંક માન્ય હશે. (અક્ષર માન્ય રાખવામાં આવશે નહીં).
- પિનકોડ છ અંકોનો હોવો જોઈએ.

જો કોઈ પણ યથાર્થતાનો ભંગ કરવામાં આવશે તો ઉપયોગકર્તા સમક્ષ એલર્ટ સંદેશ દર્શાવવામાં આવશે. આ યથાર્થતા ઉમેરવા નીચે જણાવેલ પગલાંને અનુસરો :

- આ જ ફૉર્મમાં પિનકોડ માટેનું એક લેબલ અને ઇનપુટ ફિલ્ડ ઉમેરો.
- આકૃતિ 2.22માં દર્શાવેલ સ્ક્રિપ્ટ સૉર્સવ્યૂમાં ઉમેરો.

```
<script>
function validateForm()
{
 var x=document.js1.name.value;
 var y=document.js1.pincode.value;
 if (x==null || x=="")
 {
  alert("Please enter the Name");
  document.js1.name.focus();
  return false;
 }

 if (y=="" || isNaN(y) || y.length>6 || y.length<6)
 {
  alert("Please enter the Pincode properly");
  document.js1.pincode.focus();
  return false;
 }
}
</script>
```

આકૃતિ 2.22 : નામ અને પિનકોડની યથાર્થતા ચકાસવા માટેની જાવાસ્ક્રિપ્ટ

સ્ક્રિપ્ટમાં જોઈ શકાય છે તે મુજબ, y નામનો અન્ય એક ચલ ઘોષિત કરવામાં આવે છે. y ચલમાં પિનકોડ ફિલ્ડની કિંમત સમાવવામાં આવી છે. if શરતમાં, isNaN() નામના વિધેયનો ઉપયોગ કર્યો છે. આ વિધેયને સમજીએ.

isNaN()

આંકડાકીય કિંમતો સાથે કાર્ય કરવા માટે જાવાસ્ક્રિપ્ટ આંતરપ્રસ્થાપિત વિધેયોનો ઉપયોગ કરે છે. isNaN() એ સામાન્ય રીતે સૌથી વધુ ઉપયોગમાં લેવાતું આંકડાકીય વિધેય છે. NaN એટલે "Not a Number" આપેલ કિંમત આંકડાકીય પ્રકારની ન હોય, તો તે true પરત કરે છે તથા કિંમત આંકડાકીય હોય, તો તે false પરત કરે છે. ઉદાહરણ તરીકે,

isNaN(123) વિધેય false પરત કરશે, કારણકે "123" એ સંખ્યા છે.

isNaN("hello") વિધેય true પરત કરશે, કારણકે "hello" એ સંખ્યા નથી.

આમ, if વિધાનનો ઉપયોગ કરી તપાસી શકાયું છે કે y ખાલી, અંક ન હોય તેવી અથવા છ અંકોની ન હોય તેવી કિંમત હોવી જોઈએ. જો આમાંથી એક પણ શરતનો ભંગ થશે, તો ઉપયોગકર્તા સમક્ષ એલર્ટ સંદેશ દર્શાવવામાં આવશે.

બ્રાઉઝરમાં ફાઇલ ખોલો. આકૃતિ 2.23માં દર્શાવ્યા મુજબનું પરિણામ જોવા મળશે. name ફિલ્ડમાં નામ ઉમેરો. (name ફિલ્ડને ખાલી છોડશો નહીં, નહીં તો તેના માટેનો એલર્ટ સંદેશ દર્શાવવામાં આવશે.)

આકૃતિ 2.23 : બ્રાઉઝરમાં પરિણામ

Pincode ફિલ્ડને ખાલી રાખી Submit બટન પર ક્લિક કરો. આકૃતિ 2.24માં દર્શાવ્યા મુજબનો એલર્ટ સંદેશ દર્શાવવામાં આવશે.

આકૃતિ 2.24 : ખાલી ફિલ્ડ માટેનો એલર્ટ સંદેશ

હવે, Pincode ફિલ્ડમાં થોડા અક્ષર કે છ અંકોથી વધુ કોઈ આંકડાકીય સંખ્યા ઉમેરો. આમ કરવાથી આકૃતિ 2.24માં દર્શાવેલ આ જ પ્રકારનો એલર્ટ સંદેશ દર્શાવવામાં આવશે.

અહીં, Pincode ફિલ્ડમાં ત્રણે યથાર્થતા માટે આપણે એકસમાન એલર્ટ સંદેશ દર્શાવીએ છીએ. જો યથાર્થતા પ્રમાણે યોગ્ય એલર્ટ સંદેશ દર્શાવવા ઇચ્છતા હોઈએ, તો સ્ક્રિપ્ટમાં સુધારા કરવાની જરૂર પડશે. ઉદાહરણમાં ત્રણે યથાર્થતા માટે એક જ if શરતનો ઉપયોગ કરવામાં આવ્યો છે, જે એલર્ટ સંદેશ દર્શાવે છે. યથાર્થતા મુજબ એલર્ટ સંદેશ દર્શાવવા માટે, ત્રણ if શરતો ઉમેરવી પડે. ત્રણ if શરતો સાથેની સ્ક્રિપ્ટ આકૃતિ 2.25માં દર્શાવી છે.

```
<script>
function validateForm()
{
 var x=document.js1.name.value;
 var y=document.js1.pincode.value;
 if (x==null || x=="")
 {
  alert("Please enter the Name");
  document.js1.name.focus();
  return false;
 }
if (y=="")
 {
  alert("please enter the pincode properly");
  document.js1.pincode.focus();
  return false;
 }
if(isNaN(y))
{
 alert("please enter a number");
 document.js1.pincode.focus();
 return false;
}
if(y.length>6 || y.length<6)
{
 alert("please enter a six digits");
 document.js1.pincode.focus();
 return false;
}
}
</script>
```

આકૃતિ 2.25 : યોગ્ય સંદેશ દર્શાવવા માટેનો જાવાસ્ક્રિપ્ટ કોડ

ફાઇલને બ્રાઉઝરમાં ખોલો. Pincode ફિલ્ડમાં કેટલાક અક્ષરો ઉમેરીને સબમિટ બટન પર ક્લિક કરો. આકૃતિ 2.26માં દર્શાવ્યા મુજબ એલર્ટ સંદેશ રજૂ કરવામાં આવશે.

આકૃતિ 2.26 : આંકડાકીય લખાણ તપાસવા માટેનો એલર્ટ સંદેશ

Pincode ફિલ્ડમાં 6 અંકોથી નાની સંખ્યા ઉમેરો અને સબમિટ બટન પર ક્લિક કરો. આકૃતિ 2.27માં દર્શાવ્યા મુજબનો એલર્ટ સંદેશ રજૂ કરવામાં આવશે.

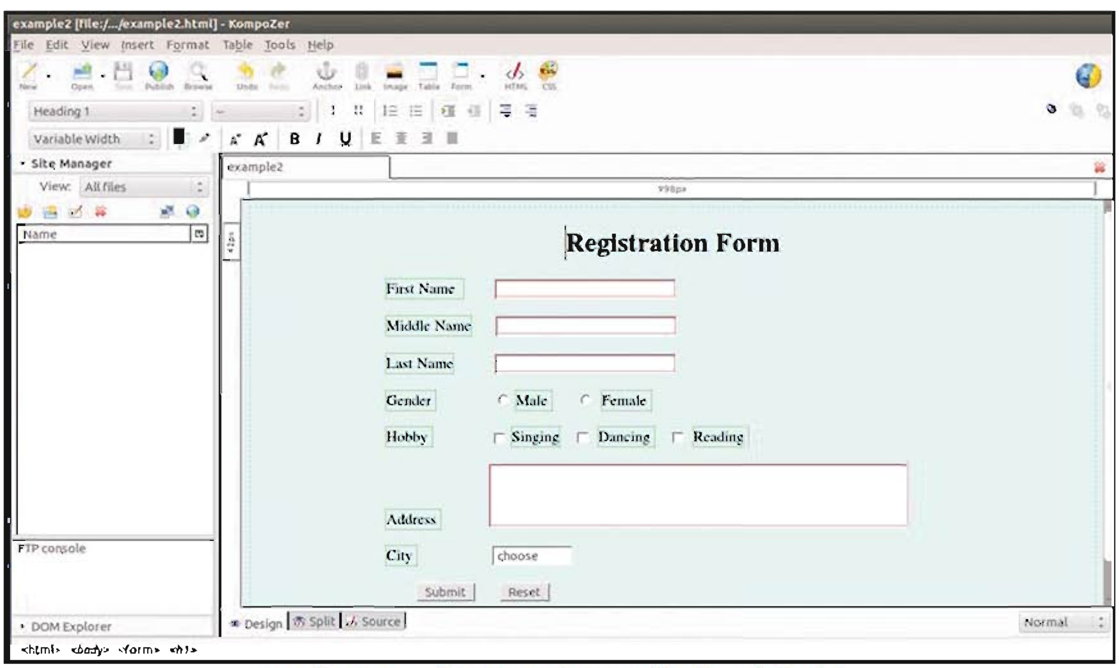
આકૃતિ 2.27 : લંબાઈ તપાસવા માટેનો એલર્ટ સંદેશ

વેબપેજમાં સંવાદન ઉમેરવા માટે જાવાસ્ક્રિપ્ટનો ઉપયોગ આપણે શીખ્યા. હવે, પ્રકરણ-1માં બનાવવામાં આવેલા નોંધણી માટેના ફૉર્મ (Registration Form)માં જાવાસ્ક્રિપ્ટ ઉમેરીએ.

યથાર્થતા લાગુ પાડવા માટે ફૉર્મમાં નીચેની યાદીમાં જણાવેલ સરળ ફેરફાર કરીશું :

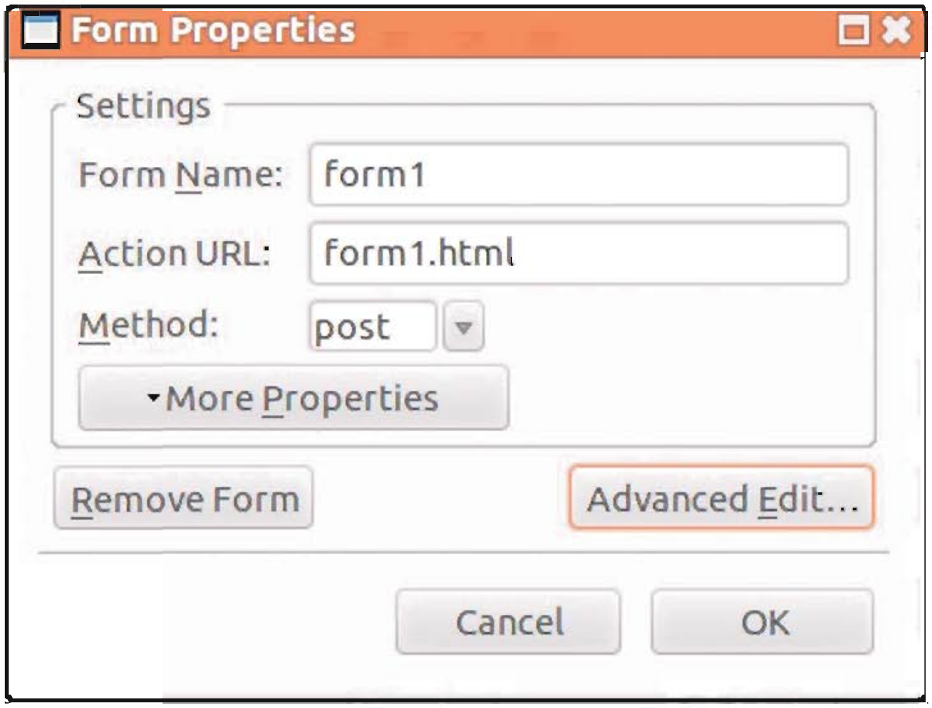
- Gender અને Hobby ફિલ્ડ શરૂઆતમાં પસંદ થયેલાં નહીં હોય.
- Address ફિલ્ડમાં શરૂઆતની કોઈ કિંમત નહીં હોય.
- City ફિલ્ડમાં "choose the city" વિકલ્પ હશે, જેની કિંમત -1 છે. અન્ય તમામ શહેરો, જેવાં કે Ahmedabad, Rajkot અને Suratને અનુક્રમે 1, 2 અને 3 કિંમતો હશે.

ઉપર્યુક્ત ફેરફારો કરવા માટે, ફૉર્મ ખોલી તેમાં સંબંધિત ફિલ્ડ પસંદ કરો. **Form → Form Field** વિકલ્પ પસંદ કરો. આમ કરવાથી જે-તે ફિલ્ડમાં ફેરફાર કરવા માટે તેના ગુણધર્મ દર્શાવવામાં આવશે. ફેરફાર કર્યા બાદ ફૉર્મ આકૃતિ 2.28માં આપેલ ફૉર્મ જેવું દેખાશે.



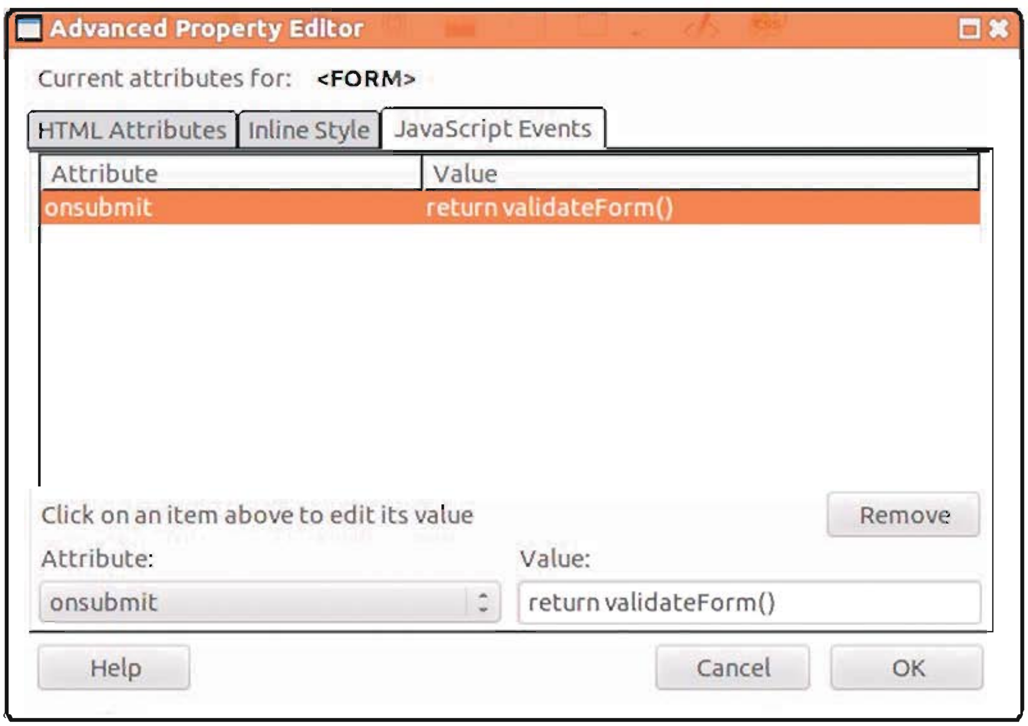
આકૃતિ 2.28 : ફેરફાર કર્યા બાદ નોંધણી માટેનું ફૉર્મ

- **Form → Define Form** વિકલ્પ પસંદ કરો. વૈકલ્પિક રીતે, ફૉર્મમાં રાઇટ ક્લિક કરીને Properties વિકલ્પ પસંદ કરી શકાય. તે આકૃતિ 2.29માં દર્શાવેલ Form Properties ડાયલૉગબૉક્સ રજૂ કરશે.



આકૃતિ 2.29 : Form Properties ડાયલૉગબૉક્સ

- Advanced Edit બટન પર ક્લિક કરો. આમ કરવાથી આકૃતિ 2.30માં આપેલ Advanced Property Editor ડાયલૉગબૉક્સ ખૂલશે. JavaScript Events વિભાગ પસંદ કરો. Attribute વિકલ્પમાં "onsubmit" અને Value વિકલ્પમાં "return validateForm() ઉમેરો. OK બટન પર ક્લિક કરો.



આકૃતિ 2.30 : Advanced Property Editor ડાયલૉગબૉક્સ

- આકૃતિ 2.31માં દર્શાવેલ સ્ક્રિપ્ટ HTMLના <head> વિભાગમાં ઉમેરો.

```
<script>
function validateForm()
{
var x=document.form1.firstname.value;
var y=document.form1.middlename.value;
var z=document.form1.lname.value;
var r=document.form1.address.value;
if (x==null || x=="")
{
alert("Please enter the first name properly");
document.form1.firstname.focus();
return false;
}

if (y==""||y==null)
{
alert("Please enter the middle name properly");
document.form1.middlename.focus();
return false;
}
if(z==""||z==null)
{
alert("Please enter a last name properly");
document.form1.lname.focus();
return false;
}
if(r=="")
{
alert("Please enter address");
document.form1.address.focus();
return false;
}
if (( document.form1.gender[0].checked == false ) && ( document.form1.gender[1].checked ==
false ) )
{
alert ( "Please choose your Gender: Male or Female" );
document.form1.gender[0].focus();
return false;
}
if ((document.form1.hobby[0].checked == false) && (document.form1.hobby[1].checked ==
false ) && (document.form1.hobby[2].checked == false))
{
alert ( "Please choose a hobby" );
document.form1.hobby[0].focus();
return false;
}
if( document.form1.city.value == "-1" )
{
alert( "Please provide your city!" );
document.form1.city.focus();
return false;
}
}
</script>
```

આકૃતિ 2.31 : જાવાસ્ક્રિપ્ટની યથાર્થતા

સ્ક્રિપ્ટમાં દર્શાવ્યા મુજબ x, y, z અને r એમ ચાર ચલ ઘોષિત કરવામાં આવ્યા છે.

var x=document.form1.firstname.value વિધાનમાં form1 એ ફૉર્મના નામનો સંદર્ભ આપે છે, firstname પદ ઘટકને આપેલ નામનો સંદર્ભ આપે છે. ("first name" ઇનપુટ ફિલ્ડને આપવામાં આવેલ નામ). આમ,

- x ચલમાં first nameની કિંમતનો સંગ્રહ કરવામાં આવશે.
- y ચલમાં middle nameની કિંમતનો સંગ્રહ કરવામાં આવશે.
- z ચલમાં last nameની કિંમતનો સંગ્રહ કરવામાં આવશે.
- r ચલમાં addressની કિંમતનો સંગ્રહ કરવામાં આવશે.

સ્ક્રિપ્ટમાં, if શરતનો ઉપયોગ કરી,

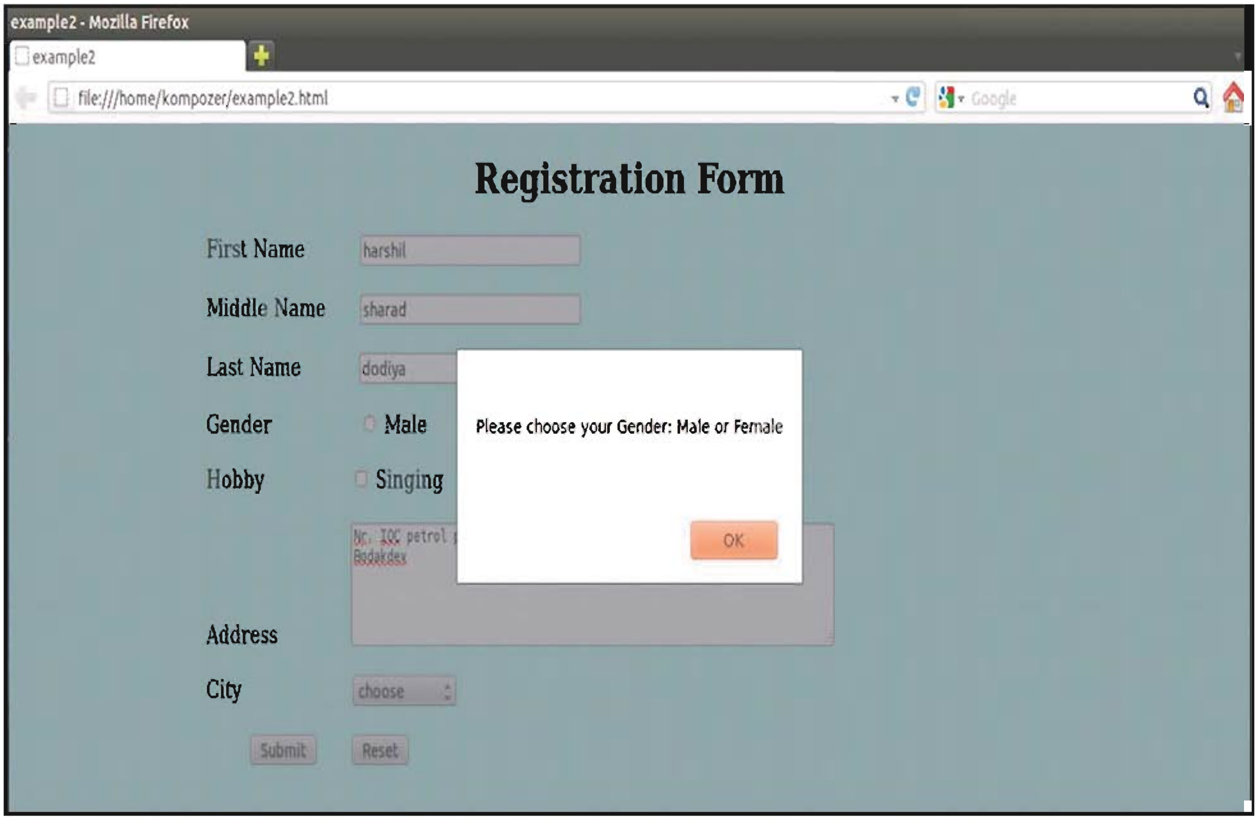
- first name ખાલી છે કે નહીં તે ચલ x દ્વારા ચકાસવામાં આવશે.
- middle name ખાલી છે કે નહીં તે ચલ y દ્વારા ચકાસવામાં આવશે.
- last name ખાલી છે કે નહીં તે ચલ z દ્વારા ચકાસવામાં આવશે.
- address ખાલી છે કે નહીં તે ચલ r દ્વારા ચકાસવામાં આવશે.

જાતિ (gender) ફિલ્ડ માટેનાં રેડિયો-બટનને "gender" નામ દ્વારા જૂથમાં મૂકવામાં આવ્યાં છે. gender બે ઘટક ધરાવતો ઍરે છે. document.form1.gender[0].checkedનો ઉપયોગ કરી gender ઍરેના પ્રથમ ઘટકની ચકાસણી if વિધાન દ્વારા કરવામાં આવે છે તથા document.form1.gender[1].checkedનો ઉપયોગ કરી if વિધાન દ્વારા ઍરેના બીજા ઘટકની ચકાસણી કરવામાં આવે છે. જો આ બન્નેનું પરિણામ false મળે તો ઉપયોગકર્તાએ કોઈ વિકલ્પ પસંદ કરેલ નથી. આથી એલર્ટ સંદેશ દર્શાવવામાં આવશે અને રેડિયો-બટન પર નિયંત્રણ (focus) લઈ જવામાં આવશે.

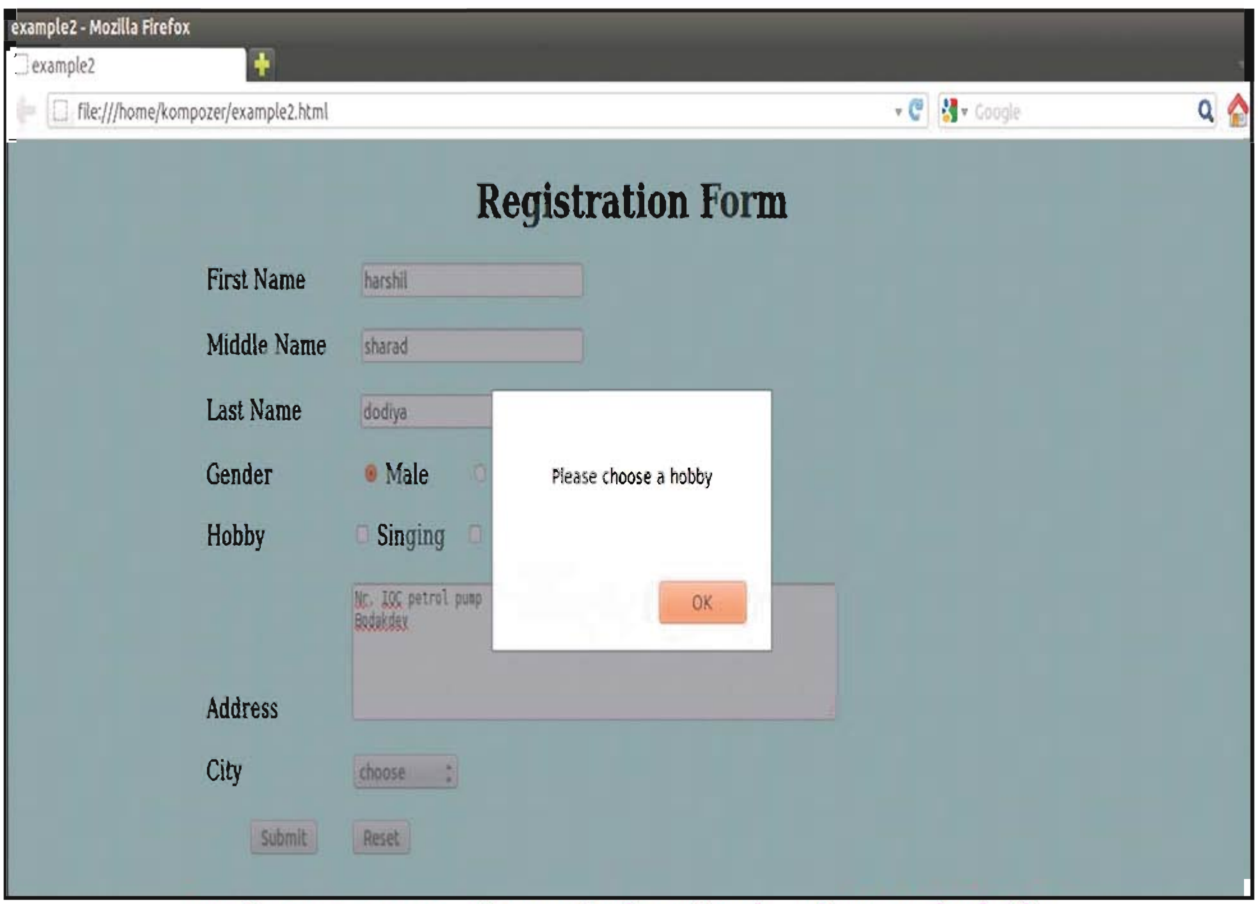
આવી જ રીતે, શોખ ફિલ્ડ માટેનાં ચેકબૉક્સને hobby નામ દ્વારા જૂથમાં મૂકવામાં આવ્યાં છે. hobby ત્રણ ઘટક ધરાવતો ઍરે છે. જેવી રીતે if શરત દ્વારા gender ફિલ્ડની કિંમત તપાસવામાં આવી, તેવી જ રીતે hobby ફિલ્ડની કિંમત ચકાસવામાં આવશે. અને જો તમામ ત્રણ ઘટક false દર્શાવે, તો ઉપયોગકર્તાએ કોઈ વિકલ્પ પસંદ કર્યો નથી. તેથી, ઉપયોગકર્તા સમક્ષ શોખ પસંદ કરવા અંગેનો એલર્ટ સંદેશ દર્શાવવામાં આવશે અને ચેકબૉક્સ પર નિયંત્રણ (focus) લઈ જવામાં આવશે.

શહેર માટેના ફિલ્ડ માટે -1 કિંમત તપાસવામાં આવશે. (**નોંધ :** -1 કિંમત "choose the city" માટે રાખવામાં આવી છે.), તે દર્શાવે છે કે ઉપયોગકર્તા દ્વારા એક પણ કિંમત પસંદ કરવામાં આવી નથી. ત્યારે ઉપયોગકર્તા સમક્ષ એલર્ટ સંદેશ દર્શાવવામાં આવશે. ફિલ્ડ ખાલી છોડવાથી મળતા કેટલાક એલર્ટ સંદેશાઓ આકૃતિ 2.32થી 2.35માં દર્શાવ્યા છે.

આકૃતિ 2.32 : middle name ફિલ્ડ ખાલી છોડવામાં આવે ત્યારે મળતા એલર્ટ સંદેશ

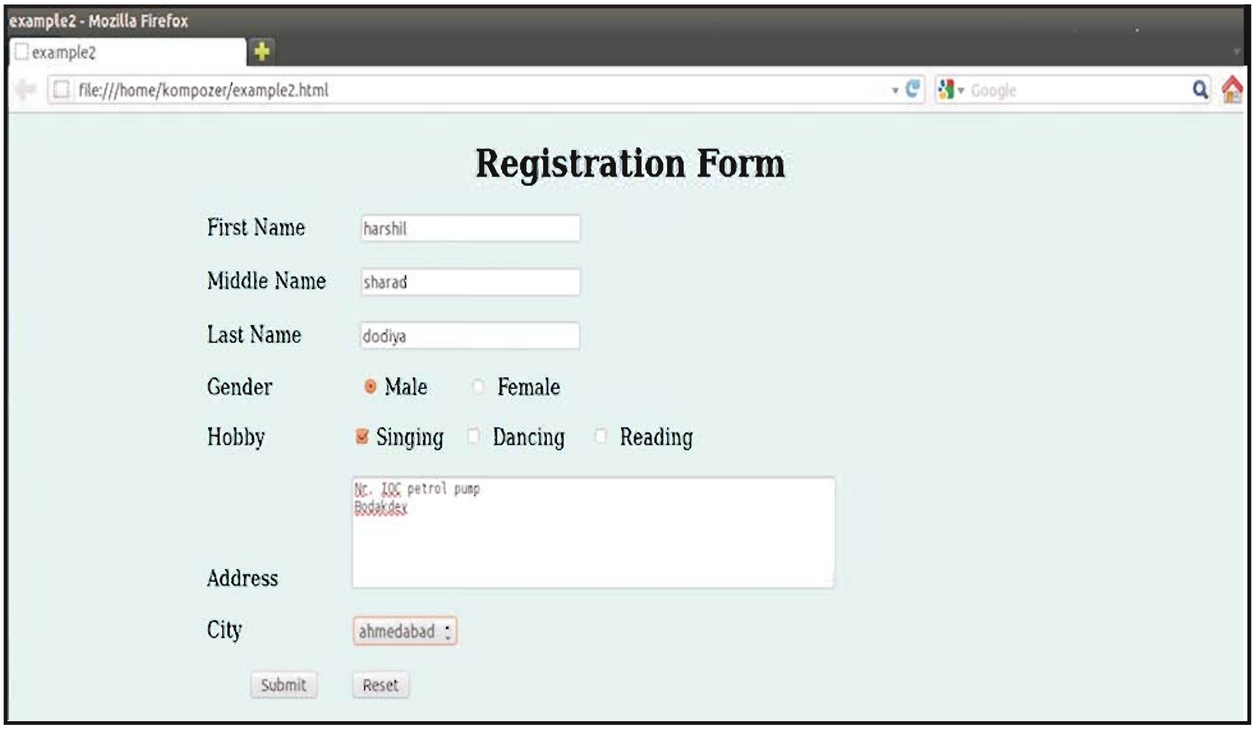


આકૃતિ 2.33 : gender ફિલ્ડ ખાલી છોડવામાં આવે ત્યારે મળતા એલર્ટ સંદેશ



આકૃતિ 2.34 : hobby ફિલ્ડ ખાલી છોડવામાં આવે ત્યારે મળતા એલર્ટ સંદેશ

તમામ ફિલ્ડની કિંમતો ઉમેર્યા પછી ફૉર્મનો અંતિમ દેખાવ આકૃતિ 2.35માં દર્શાવ્યો છે.



આકૃતિ 2.35 : અંતિમ ફૉર્મ

ફૉર્મની યથાર્થતા માટે તથા વેબપેજને વધુ સંવાદિત બનાવવા માટે આપણે જાવાસ્ક્રિપ્ટનો અભ્યાસ કર્યો.

સારાંશ

આ પ્રકરણમાં આપણે કેસ્કેડિંગ સ્ટાઇલશીટ અને જાવાસ્ક્રિપ્ટ વિશે અભ્યાસ કર્યો. વેબસાઇટના દર્શનીય ઘટકો માટેની શૈલી સ્પષ્ટ કરવા CSSનો ઉપયોગ કરવામાં આવે છે. તેની મદદથી દસ્તાવેજની માહિતીને તેના દેખાવ સંબંધી વિગતોથી અલગ રાખવામાં આવે છે. આપણે CSSના ફાયદા અને ગેરફાયદા વિશે ચર્ચા કરી. વેબપેજમાં તાર્કિક પાસું ઉમેરવા માટે જાવાસ્ક્રિપ્ટનો ઉપયોગ કરવામાં આવે છે. સ્ક્રિપ્ટિંગ ભાષા એક સરળ, હળવી પ્રોગ્રામિંગ ભાષા છે, જેમાં અદ્યતન પ્રોગ્રામિંગ કાર્યનો સમાવેશ કરવામાં આવતો નથી. વેબપેજની રૂપરેખા સુધારવા તથા ફૉર્મની યથાર્થતા ચકાસવા માટે તેનો ઉપયોગ કરવામાં આવે છે. તે HTML પાનાંમાં સંવાદન ઉમેરે છે તથા તેને પ્રત્યક્ષ રીતે HTML કોડમાં ઉમેરવામાં આવે છે. HTML પાનામાં જાવાસ્ક્રિપ્ટ ઉમેરવા માટે <script>... </script> ટૅગનો ઉપયોગ કરવામાં આવે છે.

સ્વાધ્યાય

1. કેસ્કેડિંગ સ્ટાઇલશીટનો ઉદ્દેશ જણાવો.
2. વેબપેજમાં શૈલી અને તેમાં આવેલ વિગતોને શા માટે અલગ રાખવી જોઈએ ?
3. CSSની વાક્યરચના સમજાવો.
4. CSSના ફાયદાઓ જણાવો.
5. CSSના ગેરફાયદાઓ જણાવો.
6. HTML પાનામાં જાવાસ્ક્રિપ્ટનો ઉપયોગ શા માટે કરવામાં આવે છે ?

7. HTML પાનામાં જાવાસ્ક્રિપ્ટને કેવી રીતે ઓળખી શકાય ? ઉદાહરણ આપો.

8. આપેલ વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) નીચેનામાંથી કોનો ઉપયોગ વેબસાઇટમાં દર્શનીય ઘટકો માટેની શૈલી સ્પષ્ટ કરવા માટે થાય છે ?

(a) Cascading Style Sheets (b) Webpage

(c) Form (d) Animation

(2) નીચેનામાંથી કોને CSSની વાક્યરચનામાં એક વિશિષ્ટ નિશાની તરીકે ઓળખવામાં આવે છે ?

(a) નિયમ (Rules) (b) પસંદગીકાર (Selector)

(c) ઘોષણા (Declaration) (d) નિવેશ (Input)

(3) નીચેનામાંથી CSS નિયમના મુખ્ય બે વિભાગ ક્યા છે ?

(a) Selector, declaration (b) Select, declaration

(c) Selector, declare (d) Selection, declaration

(4) જેના પર શૈલી લાગુ પાડવાની છે, તેવા HTMLના ઘટકને શું કહે છે ?

(a) declaration (b) selector

(c) select (d) declare

(5) નીચેનામાંથી CSSની વાક્યરચના કઈ છે ?

(a) select {property : value} (b) selector {value : property}

(c) selector {property : value} (d) selection {property : value}

(6) નીચેનામાંથી કઈ સંસ્થાએ જાવાસ્ક્રિપ્ટનો વિકાસ કર્યો ?

(a) Yahoo (b) Google

(c) Wikipedia (d) Netscape

(7) નીચેનામાંથી કઈ સ્ક્રિપ્ટિંગ ભાષાનો ઉપયોગ વેબપેજમાં પ્રોગ્રામિંગ ઉમેરવા માટે કરવામાં આવે છે ?

(a) Action script (b) JavaScript

(c) HTML (d) CSS

(8) નીચેનામાંથી કઈ સરળ, હળવી પ્રોગ્રામિંગ ભાષાને સ્ક્રિપ્ટિંગ ભાષા તરીકે ઓળખવામાં આવે છે કે જેમાં અદ્યતન પ્રોગ્રામિંગ-કાર્યોનો સમાવેશ કરવામાં આવ્યો ન હોય ?

(a) JavaScript (b) HTML

(c) C (d) Java

(9) નીચેનામાંથી ક્યા ટેગનો ઉપયોગ HTML પાનાંમાં જાવાસ્ક્રિપ્ટનો કોડ ઉમેરવા માટે કરવામાં આવે છે ?

(a) <script>… <script> (b) <script>… </script>

(c) <script>… <//script> (d) </script>… </script>

(10) નીચેનામાંથી કઈ નિશાની દ્વારા જાવાસ્ક્રિપ્ટ બ્લૉકની શરૂઆત અને અંત દર્શાવવામાં આવે છે ?

(a) અર્ધવિરામ (semicolon) (b) ચોરસ કૌંસ (square bracket)

(c) છગડિયા કૌંસ (curly bracket) (d) સાદા કૌંસ (round bracket)

(11) નિશ્ચિત કાર્ય કરવા માટે પુનઃઉપયોગમાં લઈ શકાય તે પ્રકારના ઘટકને શું કહે છે ?

(a) Array (b) Code

(c) Program (d) Function

(12) નીચેનામાંથી ક્યા વિધાનનો ઉપયોગ વિધેયમાં કિંમત પરત કરવા માટે થાય છે ?

(a) return (b) function

(c) select (d) send

(13) ઉપયોગકર્તા અને વેબપેજ વચ્ચેના સંવાદનથી શું બને છે ?

(a) વિધેય (Function) (b) પ્રતિભાવ (Response)

(c) ઘટના (Event) (d) કિંમત (Value)

(14) નીચેનામાંથી કોને ઘટના કહી શકાય નહીં ?

(a) Abort (b) Mouseover

(c) Set (d) Load

(15) વિગતોનો સંગ્રહ કરવા માટેના સંગ્રાહકને શું કહે છે ?

(a) ચલ (Variable) (b) પૂર્ણાંક (Integer)

(c) ઘટના (Event) (d) ઘટના-સંચાલક (Event handler)

(16) BOM એટલે શું ?

(a) Browser of Model (b) Browser Object Model

(c) Browser Object Modelling (d) Browse Object Model

(17) બ્રાઉઝર ઑબ્જેક્ટ મૉડેલમાં સૌથી ઉપરના સ્તરે રહેલ ઑબ્જેક્ટ કયો છે ?

(a) Window (b) Document

(c) Page (d) Location

(18) NaN એટલે શું ?

(a) Not a Numeric (b) Not a Number

(c) Not a Noun (d) Not an Numeric

પ્રાયોગિક સ્વાધ્યાય

1. નીચે આપેલ નિયમોના ઉપયોગ દ્વારા H1 ટેગ માટે CSSની રચના કરી તેને લખાણ પર લાગુ કરો :
   - Font : Times New Roman
   - Color : Red
   - Case : Lowercase
   - Fontstyle : Italic
   - Alignment : Centre
   - Text decoration : Underline
   - Background color : Light Grey
   - Border : Dotted
2. આપેલ ઘટકો સાથેનું ફૉર્મ બનાવો : name, email address, phone number અને submit button. તેમાં નીચે દર્શાવ્યા મુજબ યથાર્થતા ઉમેરો :
   - ફિલ્ડ ખાલી ન હોવાં જોઈએ.
   - ફોન માટેના ફિલ્ડમાં માત્ર અંક માન્ય છે.
   - ફોનનંબર દસ અંકોનો હોવો જોઈએ.
3. વિદ્યાર્થીની અંગત વિગતો ઉમેરવા માટેના પ્રકરણ-1માં બનાવવામાં આવેલ ફૉર્મમાં યથાર્થતા ઉમેરો.
4. પ્રતિસાદ (feedback) માટેના પ્રકરણ-1માં બનાવવામાં આવેલ ફૉર્મમાં યથાર્થતા ઉમેરો.

કમ્પોઝરનો ઉપયોગ કરી સરળ વેબસાઇટની રચના 3

આધુનિક યુગમાં કોઈ પણ વ્યવસાયમાં વેબસાઇટ ખૂબ અગત્યનો ભાગ ભજવે છે. તે વિશ્વકક્ષાએ વ્યવસાયની રજૂઆતમાં મદદરૂપ બને છે. વ્યવસાયના વિકાસમાં, ઉત્પાદનના વેચાણમાં અને ગ્રાહકોને બહોળી સંખ્યામાં આકર્ષવામાં તે મદદરૂપ બને છે. તેથી જ, વેબસાઇટની રચના શ્રેષ્ઠ તક્નિક દ્વારા કરવામાં આવી હોય તે જરૂરી છે, જેથી સંસ્થા મહત્તમ સંખ્યામાં મુલાકાતીઓ મેળવી શકે અને ઉચ્ચતમ નફો રળી શકે.

વેબસાઇટ એ વિશિષ્ટ હેતુ માટે પરસ્પર જોડાયેલાં વેબપેજનો સમૂહ છે. સંવાદિત (interactive), ઉપયોગકર્તા માટે સરળ (user friendly) તથા ચોક્કસ (accurate) અને ઉપયોગી માહિતી ધરાવતી વેબસાઇટ બનાવવી એ પડકારજનક કાર્ય છે. વેબસાઇટની રચના એવી હોવી જોઈએ, જેથી ઉપયોગકર્તાને તે માહિતીપ્રદ લાગે તથા ઉપયોગકર્તા વારંવાર તેની મુલાકાત લે. જો સાઇટ વ્યાવસાયિક હોય, તો તે ગ્રાહકોની સંખ્યામાં વધારો કરે. વેબસાઇટની રચના કરતી વખતે પૂરતું ધ્યાન આપવામાં આવે, તો આ હેતુ સિદ્ધ થઈ શકે. આ પ્રકરણમાં વેબસાઇટનું આયોજન કરવા માટે ધ્યાનમાં રાખવાના સામાન્ય મુદ્દાઓની ચર્ચા કરવામાં આવી છે. આપણે કમ્પોઝર (KompoZer) પ્રોગ્રામની મદદથી વેબસાઇટ બનાવી તેને ઇન્ટરનેટ પર પ્રકાશિત (publish) કરીશું.

વેબસાઇટનું આયોજન (Planning for the Website)

સારી વેબસાઇટનો વિકાસ એ કોઈ પણ અન્ય યોજના જેવી જ એક યોજના છે અને તેના માટે વિસ્તૃત આયોજન જરૂરી છે. આયોજન જેટલું બહેતર હશે, વેબસાઇટની ઉપયોગિતાના સંદર્ભે સફળતાની તક પણ તેટલી જ વધુ રહેશે. બીજા શબ્દોમાં કહીએ તો વેબસાઇટ બનાવવા માટેનું ધ્યેય સિદ્ધ કરી શકાશે. સારી વેબસાઇટ બનાવવા માટે આયોજનની પ્રક્રિયામાં ધ્યાનમાં રાખવા માટેના કેટલાક અગત્યના મુદ્દા નીચે દર્શાવ્યા છે :

હેતુ (Purpose)

વેબસાઇટનો હેતુ સ્પષ્ટપણે વ્યાખ્યાયિત થયેલો હોય તે જરૂરી છે. વેબસાઇટની રચના કરતાં પહેલાં તેની વ્યાખ્યા અને ધ્યેય સ્પષ્ટ હોવાં જરૂરી છે. યોગ્ય રીતે રચના પામેલી વેબસાઇટ સંસ્થાને મદદરૂપ બને છે. વેબસાઇટની રચનાનો હેતુ વ્યક્તિઓના સમૂહને માહિતી પૂરી પાડવાનો, નવા ગ્રાહકોને આકર્ષવાનો કે ઉત્પાદનને ઑનલાઇન વેચવાનો હોઈ શકે. હેતુ નક્કી થયા પછી વેબસાઇટના વિષયવસ્તુ (content) અને રૂપરેખા (layout)ને યોગ્ય રીતે વિકસાવી શકાય.

પ્રેક્ષકગણ (Audience)

વેબસાઇટના રચનાત્મક વિભાગનું કાર્ય શરૂ કરતાં પહેલાં અપેક્ષિત ઉપયોગકર્તા અંગેની જાણકારી હોવી જરૂરી છે. વેબસાઇટમાં સામાન્ય અને વિસ્તૃત (detailed) એમ બંને પ્રકારની માહિતી હોવી જોઈએ. તેમાં વિશિષ્ટ અને સરળ માહિતીનો સમાવેશ પણ થયો હોવો જોઈએ. માહિતીના વિવિધ પ્રકાર, વિસ્તાર અને વિશેષતાના વ્યાપનો આધાર અપેક્ષિત પ્રેક્ષકોની રુચિ પર રહેલો છે. વેબસાઇટ પ્રત્યેની ઉપયોગકર્તાની અપેક્ષાઓ જાણવી જરૂરી છે, કારણકે તે જાણકારી વેબસાઇટના યોગ્ય વિષયવસ્તુ અને રૂપરેખાના વિકાસમાં મદદરૂપ બને છે અને વેબસાઇટ ઉપયોગી તથા સફળ હોવાની ખાતરી આપે છે.

એક વધુ ધ્યાન આપવા લાયક બાબત છે ઉપયોગકર્તાના ઇન્ટરનેટ જોડાણની ઝડપ. જો વેબસાઇટ પર વિશાળ ચિત્રાત્મક ફાઇલો રાખવામાં આવી હોય, તો તે ડાઉનલોડ થવામાં લાંબો સમય લે છે. કેટલીક વાર ઉપયોગકર્તા આવી વિશાળ ફાઇલોની રાહ જોવામાં અધીરો બની સાઇટ છોડી દે એવું પણ બને. માટે, વેબસાઇટમાં જરૂરી હોય તેટલા પ્રમાણમાં જ ચિત્રો અને મલ્ટિમીડિયા ફાઇલનો સમાવેશ કરવામાં આવે તે ઇચ્છનીય છે.

વિષયવસ્તુ (Content)

વેબસાઇટમાં સંપૂર્ણ અને સુસંગત માહિતી હોવી અત્યંત જરૂરી છે. અસંગત માહિતીનો વધુ ઉપયોગ ઉપયોગકર્તાને નિરાશ બનાવી શકે છે, કારણકે તે સ્પષ્ટ વિગતો પ્રાપ્ત કરી શકતો નથી. વળી, જો પૂરી પાડવામાં આપેલી માહિતી અપૂર્ણ હોય, તો ઉપયોગકર્તા વેબસાઇટ છોડી દઈ શકે છે. યોગ્ય અને સુસંગત માહિતી મૂકવાથી ઉપયોગકર્તાને સંતોષ પ્રાપ્ત થાય છે તથા વેબસાઇટમાં તેનો રસ જળવાઈ રહે છે. વેબસાઇટના વિષયવસ્તુને સામાન્ય અને વિસ્તૃત વર્ગોમાં વહેંચવું જરૂરી છે.

સામાન્ય વિષયવસ્તુ ઉપયોગકર્તા સમક્ષ સાઇટ, સંસ્થા, ઉત્પાદન અને સેવાઓ તથા અન્ય વસ્તુઓનું વિહંગાવલોકન (overview) પૂરું પાડે છે. માહિતીની સંક્ષિપ્ત સમજૂતી ઉપયોગકર્તાની શોધમાં મદદરૂપ નીવડે છે. તે યોગ્ય અને સ્પષ્ટ વિગતોનો નિર્દેશ કરે છે તથા સાઇટમાં આવેલ અન્ય ઉપલબ્ધ વસ્તુઓ માટેની જાણકારી પૂરી પાડે છે.

વિસ્તૃત વિષયવસ્તુ ઉપયોગકર્તાને ઉત્પાદન અને સેવાઓ અંગેની વિસ્તૃત માહિતી પૂરી પાડે છે. વેબસાઇટની રૂપરેખા એવી હોવી જોઈએ કે ઉપયોગકર્તા તેને ઉપયોગી એવી વિશિષ્ટ માહિતી સરળતાથી મેળવી શકે.

લખાણના ફકરા હંમેશાં યોગ્ય લંબાઈના રાખવા જોઈએ. વેબપેજ પર વિપુલ પ્રમાણમાં વિષયવસ્તુ દર્શાવવાનું હોય તો તેને લખાણના નાના નાના વિભાગોમાં વહેંચી નાખવું જોઈએ. આમ કરવાથી ઉપયોગકર્તા પોતાને જરૂરી માહિતી સરળતાથી મેળવી શકે છે. વેબસાઇટમાં નેવિગેશન (navigation) શક્ય એટલું સરળ હોવું જોઈએ. એક પાના પરથી અન્ય પાના પરનું સ્થાનાંતરણ સરળ હોવું જોઈએ, જેથી ઉપયોગકર્તાને ખબર પડે કે તે ક્યાં છે અને હોમપેજ પર કેવી રીતે પાછા ફરી શકાશે.

સામાન્ય અને વિશિષ્ટ એમ બંને પ્રકારની જરૂરી માહિતી સરળતાથી ઉપલબ્ધ બને, તો જ વેબસાઇટની રચનાનો ઉદ્દેશ્ય પાર પડે છે. સારા વિષયવસ્તુથી સભર વેબસાઇટ ઉદ્યોગ કે અન્ય સંસ્થા માટે એક કિંમતી સાધન બની રહે છે. અન્ય સાધનોની જેમ વેબસાઇટ પણ સંસ્થા માટે શ્રેષ્ઠ કાર્ય કરી શકે છે.

માધ્યમ (Medium)

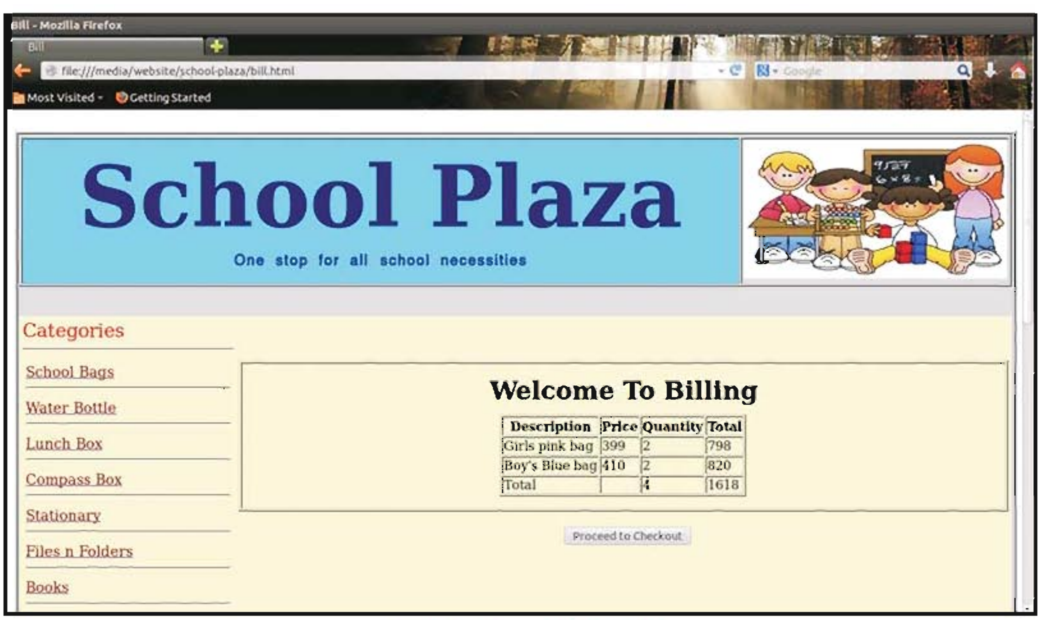
વધુ ને વધુ વ્યક્તિઓ હવે સ્માર્ટફોન અને ટેબ્લેટ દ્વારા ઇન્ટરનેટનો ઉપયોગ કરી રહી છે. વેબસાઇટની રૂપરેખાનું કદ કમ્પ્યૂટર, સ્માર્ટફોન અને ટેબ્લેટ જેવાં તમામ સાધનોને અનુરૂપ હોવું જોઈએ. આ ઉપરાંત વેબસાઇટની રૂપરેખા તૈયાર કરતી વખતે એ પણ ધ્યાનમાં રાખવું જોઈએ કે તેને મોઝિલા ફાયરફોક્સ, ક્રોમ, ઓપેરા અને ઇન્ટરનેટ એક્સપ્લોરર જેવાં તમામ પ્રચલિત વેબબ્રાઉઝરમાં યોગ્ય રીતે દર્શાવવામાં આવે.

કમ્પોઝરનો ઉપયોગ કરી સરળ વેબસાઇટ બનાવવી (Creating a Simple Website Using KompoZer)

અગાઉના પ્રકરણમાં આપણે કમ્પોઝરની મદદથી વેબપેજની રચના તથા સંપાદન માટે જાવા સ્ક્રિપ્ટના ઉપયોગ વિશે ચર્ચા કરી હતી. વેબસાઇટ બનાવવા માટે સામાન્ય રીતે ધ્યાનમાં લેવાતા મુદ્દાઓ પણ આપણે જોયા હતા. હવે આપણે કમ્પોઝરની મદદથી એક નાની ઇ-કૉમર્સની વેબસાઇટ કેવી રીતે બનાવી શકાય, તેનો અભ્યાસ કરીશું.

આપણે "School Plaza" નામની એક સરળ ઇ-કૉમર્સ વેબસાઇટની રચના કરીએ, જે શાળામાં જતા વિદ્યાર્થીઓની જરૂરિયાતો પૂરી કરતી હોય. આ વેબસાઇટ પર પુસ્તકો, સ્ટેશનરી, શાળાની બૅગ, પાણીની બૉટલ, નાસ્તાનો ડબ્બો, ફાઇલ, ફોલ્ડર અને તેના જેવી શાળાને સંબંધિત અન્ય વસ્તુઓ ઉપલબ્ધ હશે અને તેનો ઑર્ડર પણ આપી શકાશે. આપણે એક શૉપિંગ-કાર્ટની પણ રચના કરીશું, જે ખરીદવામાં આવેલી તમામ વસ્તુઓની યાદી અને કુલ રકમ દર્શાવશે. આપણે આયોજન કરેલી વેબસાઇટનું હોમપેજ અને શૉપિંગકાર્ટનો દેખાવ આકૃતિ 3.1 અને 3.2માં દર્શાવ્યો છે.

આકૃતિ 3.1 : વેબસાઈટનું હોમપેજ

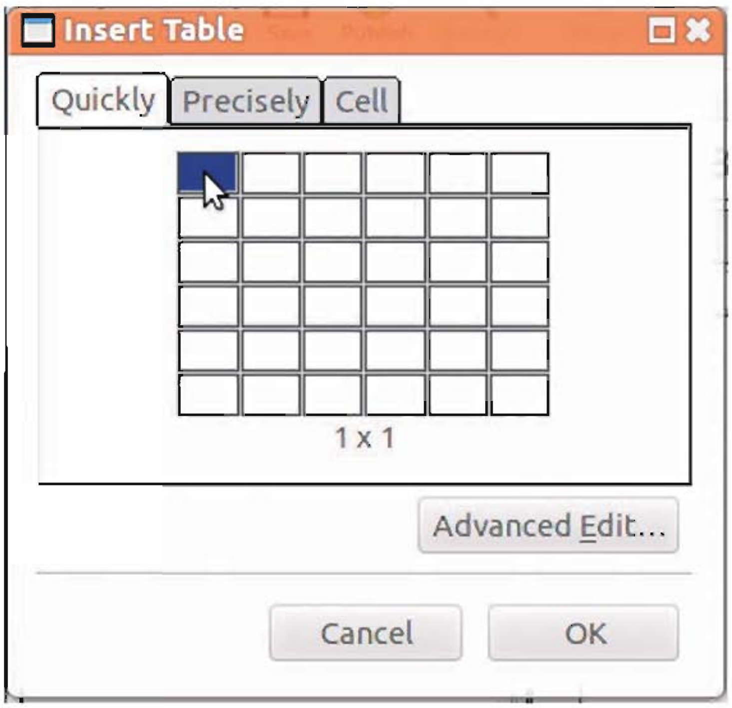


આકૃતિ 3.2 : શૉપિંગકાર્ટ

આપણે ક્રમિક પગલાં દ્વારા આ વેબસાઇટની રચના કરીએ. સૌપ્રથમ આપણે વેબસાઇટનું હોમપેજ બનાવીશું. જ્યારે ઉપયોગકર્તા વેબબ્રાઉઝરના ઍડ્રેસબારમાં URL સરનામું ઉમેરે, ત્યારે વેબસાઇટના ખોલવામાં આવતાં સૌપ્રથમ પાનાને હોમપેજ કહે છે. તમામ ઉપલબ્ધ વસ્તુઓના વર્ગ (Category)ની યાદી આપણી વેબસાઇટના હોમપેજમાં દર્શાવવી જોઈએ.

કોઈ પણ વર્ગ પર ક્લિક કરવાથી તે વર્ગમાં આવેલી વસ્તુઓ દર્શાવતું અન્ય વેબપેજ રજૂ કરવામાં આવશે. વેબસાઇટ બનાવવા માટે ટેમ્પ્લેટ (Template) નામે ઓળખાતી વિવિધ રૂપરેખાઓના નમૂના ઇન્ટરનેટ પર નિઃશુલ્ક ઉપલબ્ધ છે. આ ટેમ્પ્લેટ સરળતાથી ડાઉનલોડ કરી જરૂર મુજબ ઉપયોગમાં લઈ શકાય છે.

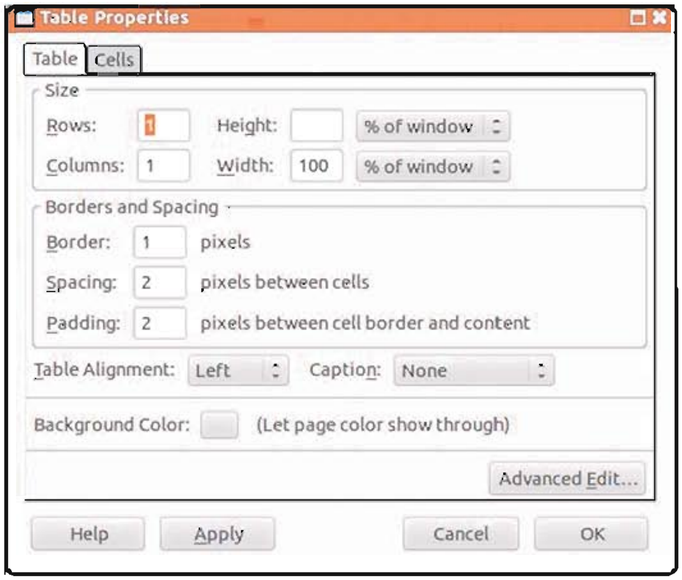
હોમપેજની રચના દ્વારા વેબસાઇટની શરૂઆત કરવા માટે કમ્પોઝર ખોલી નવું વેબપેજ બનાવો. વેબપેજમાં તમામ વિષયવસ્તુ અને ચિત્રોની ગોઠવણ કરવા માટે આકૃતિ 3.3માં દર્શાવ્યા મુજબ એક ખાનું (cell) ધરાવતું કોષ્ટક ઉમેરીશું. કોષ્ટક ઉમેરવા માટે **Insert → Table** પસંદ કરો. એક સેલ ધરાવતું કોષ્ટક પસંદ કરી OK બટન પર ક્લિક કરો.



આકૃતિ 3.3 : 1 × 1 કોષ્ટક ઉમેરવું

યોગ્ય શીર્ષક આપી વેબપેજનો સંગ્રહ કરો. શીર્ષક વેબબ્રાઉઝરના ટાઇટલબારમાં દર્શાવવામાં આવશે. અહીં, આપણે "School Plaza" શીર્ષક આપ્યું છે. "index.html" નામ આપી ફાઇલનો સંગ્રહ કરો. સામાન્ય રીતે હોમપેજને index.html નામ આપવામાં આવે છે. વેબસર્વર પર દરેક વેબસાઇટ ડિરેક્ટરીમાં બનાવવામાં આવે છે અને દરેક વેબપેજ એ વેબસર્વર પરની સ્વતંત્ર ફાઇલ છે. જ્યારે ઉપયોગકર્તા www.schoolplaza.com જેવું વેબસાઇટનું કોઈ URL બ્રાઉઝરમાં લખે છે, ત્યારે તે ખોલવામાં આવનાર ફાઇલની સ્પષ્ટતા કરતો નથી, પરંતુ વેબસર્વરને થોડું વિષયવસ્તુ દર્શાવવા માટે એક પૂર્વનિર્ધારિત ફાઇલની જરૂર પડે છે. આ પૂર્વનિર્ધારિત ફાઇલનું નામ index.html છે. આમ, જ્યારે ફાઇલના નામ વગર URL ઉમેરવામાં આવે છે, ત્યારે સર્વર પૂર્વનિર્ધારિત ફાઇલને ધ્યાનમાં લઈ તેને આપોઆપ દર્શાવે છે.

હવે, આપણે ઉમેરેલા કોષ્ટકને બેકગ્રાઉન્ડ કલર આપીશું. કોષ્ટક પસંદ કરી ડબલ ક્લિક કરો. વૈકલ્પિક રૂપે કોષ્ટક પર રાઇટ ક્લિક કરી 'Table Cell Properties' વિકલ્પ પસંદ કરી શકાય. આમ કરવાથી આકૃતિ 3.4માં દર્શાવેલું Table Properties ડાયલોગબોક્સ ખૂલશે.

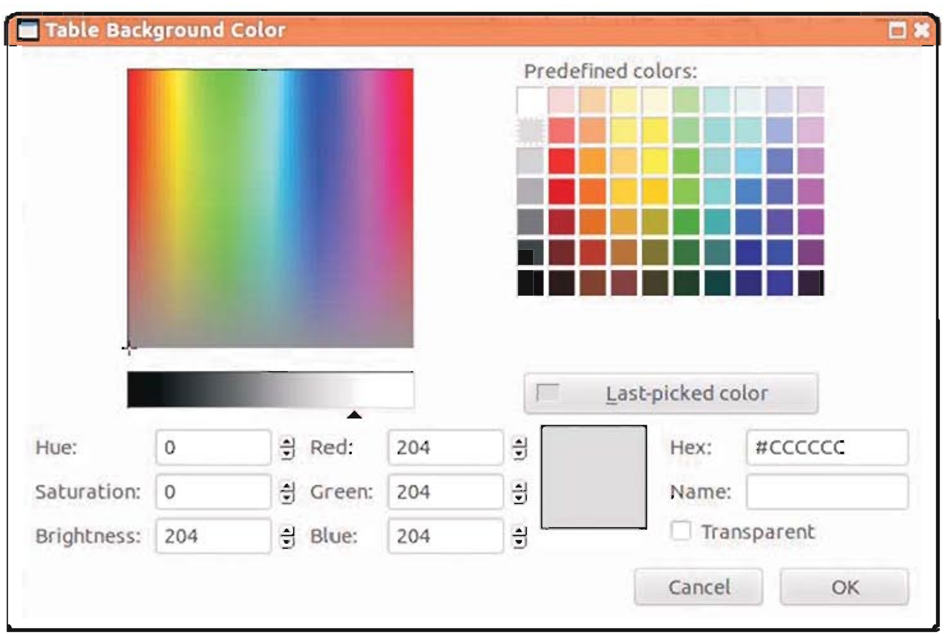


આકૃતિ 3.4 : Table Properties ડાયલોગબૉક્સ

અહીં 'Table' અને 'Cells' બે વિભાગ (Tab) જોવા મળશે, જેના દ્વારા ઉપયોગકર્તા કોષ્ટક કે સ્વતંત્ર સેલનાં વિવિધ પાસાંને નિયંત્રિત કરી શકશે. કોષ્ટકને સંબંધિત સેલ અહીં જોઈ શકાશે. Table વિભાગમાં આવેલા વિકલ્પ નીચે દર્શાવ્યા છે :

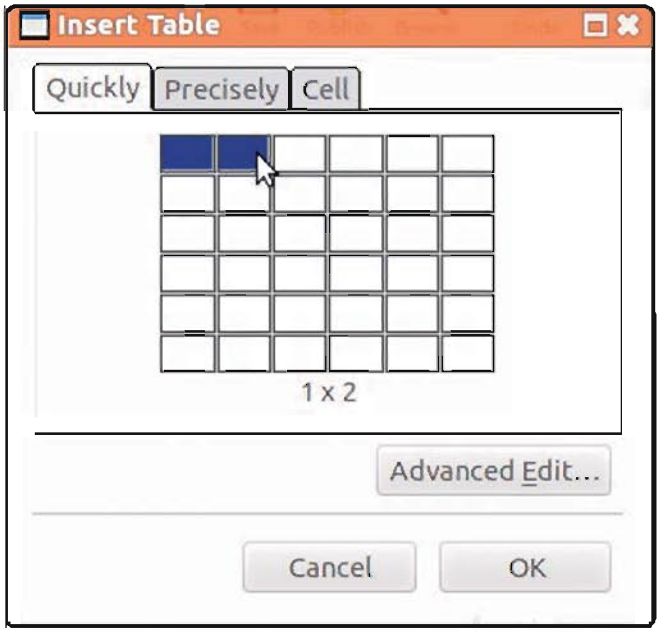
1. **Size :** તે હરોળ (row) અને સ્તંભ (column)ની સંખ્યા તથા કોષ્ટકની ઊંચાઈ અને પહોળાઈ દર્શાવે છે. ઊંચાઈ અને પહોળાઈ પિક્સેલ અથવા વિન્ડોના ટકાના સ્વરૂપે હોઈ શકે.
2. **Borders and Spacing :** કોષ્ટકને સીમારેખા (border) આપવાની હોય, તો તે માટેના વિકલ્પ અહીં દર્શાવવામાં આવે છે. સેલ વચ્ચેની જગ્યા નિશ્ચિત કરવા માટે spacingનો ઉપયોગ કરવામાં આવે છે. સેલની કિનારી અને તેમાં આવેલ લખાણ વચ્ચેની જગ્યા નિશ્ચિત કરવા paddingનો ઉપયોગ કરવામાં આવે છે.
3. **Table Alignment :** કોષ્ટકને ડાબી બાજુ, જમણી બાજુ કે વચ્ચે ગોઠવે છે.
4. **Caption :** જરૂર જણાય, તો કોષ્ટકને શીર્ષક આપી શકાય છે.
5. **Background Color :** કોષ્ટકના બૅકગ્રાઉન્ડનો કલર દર્શાવે છે.

આકૃતિ 3.5માં દર્શાવ્યા મુજબ તમારી ઇચ્છા મુજબ બૅકગ્રાઉન્ડ કલર પસંદ કરો.



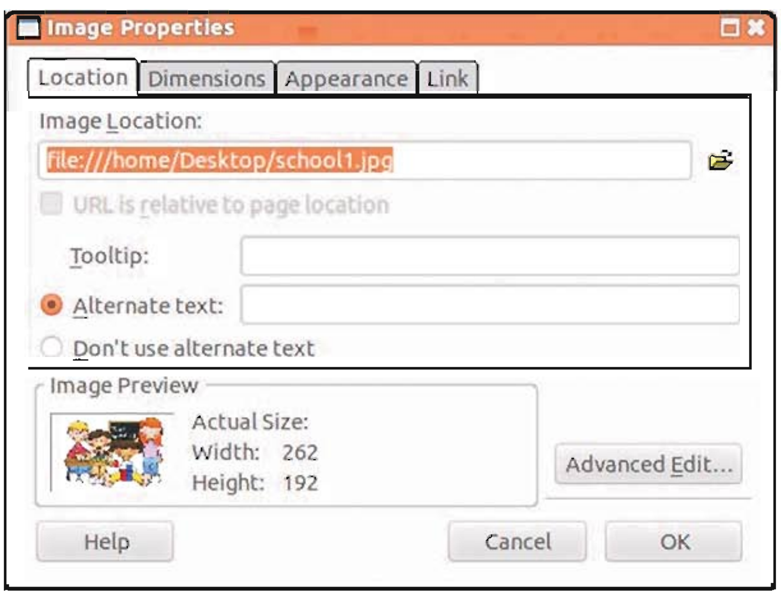
આકૃતિ 3.5 : Table Background Color ડાયલોગબૉક્સ

હવે, "School Plaza" લખાણ અને તેને સંબંધિત ચિત્ર ઉમેરવા માટે એક હરોળ અને બે સ્તંભ ધરાવતું અન્ય કોષ્ટક ઉમેરીએ. એક સ્તંભ લખાણ દર્શાવશે અને બીજામાં ચિત્ર દર્શાવવામાં આવશે. **Insert → Table** વિકલ્પ પસંદ કરો. આકૃતિ 3.6માં દર્શાવ્યા મુજબ 1 × 2 કોષ્ટક પસંદ કરો અને OK બટન પર ક્લિક કરો.



આકૃતિ 3.6 : 1×2 કોષ્ટક ઉમેરવું

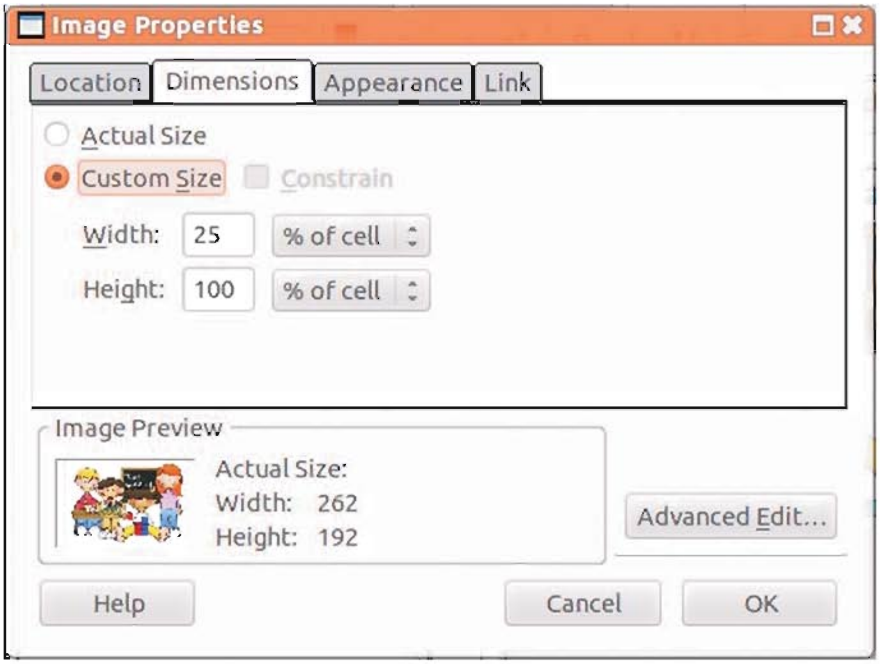
કોષ્ટકને બૅકગ્રાઉન્ડ કલર આપો. હવે કોષ્ટકના ડાબી બાજુના સેલમાં લખાણ અને જમણી બાજુના સેલમાં ચિત્ર ઉમેરીએ. ચિત્ર ઉમેરવા માટે **Insert → Image** વિકલ્પ પસંદ કરો. વૈકલ્પિક રીતે, ટૂલબાર પર આવેલા ચિત્રના આઇકન પર ક્લિક પણ કરી શકાય. આમ કરવાથી આકૃતિ 3.7માં દર્શાવેલ ડાયલોગબૉક્સ ખોલવામાં આવશે. ચિત્રનું સ્થાન શોધો. ઇનપુટ બૉક્સમાં વૈકલ્પિક લખાણ ઉમેરો અને જો વૈકલ્પિક લખાણ ઉમેરવા માંગતા ન હો, તો "Don't use alternate text" રેડિયો બટન પસંદ કરો. ચિત્રને બ્રાઉઝરમાં દર્શાવી શકાય તેમ ન હોય, ત્યારે તેના સ્થાને આ વૈકલ્પિક લખાણ દર્શાવવામાં આવશે.



આકૃતિ 3.7 : Image Properties ડાયલોગબૉક્સ

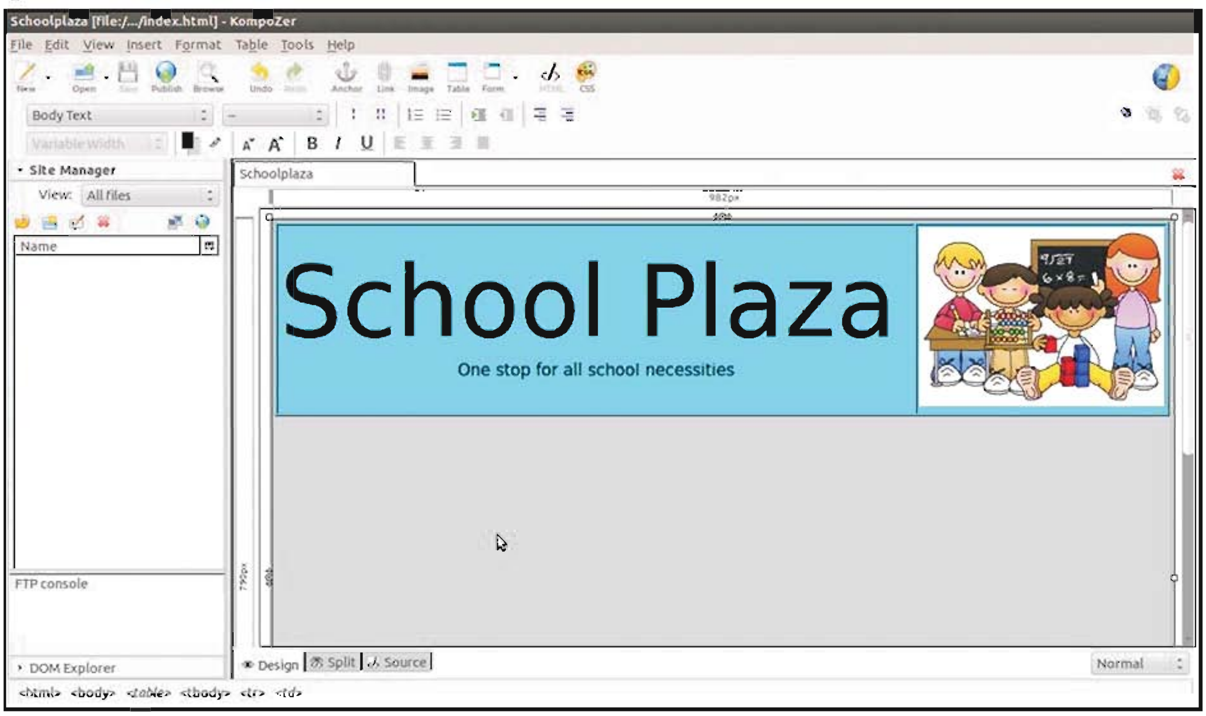
નોંધ : આપણે અહીં school1.jpg ચિત્રનો ઉપયોગ કર્યો છે. વિદ્યાર્થીઓ પોતાની પસંદગીનું ચિત્ર બનાવી શકે છે અથવા અન્ય મનપસંદ ચિત્રનો ઉપયોગ કરી શકે છે.

હવે, Dimension વિભાગ પસંદ કરી તેમાં આવેલ Custom Size વિકલ્પ પસંદ કરો. ચિત્રને આખા સેલમાં દર્શાવવાનું હોવાથી આકૃતિ 3.8માં દર્શાવ્યા મુજબ ઊંચાઈ અને પહોળાઈ 100% રાખીશું. અન્ય સેલમાં આકૃતિ 3.9માં દર્શાવ્યા મુજબ લખાણ ઉમેરો.



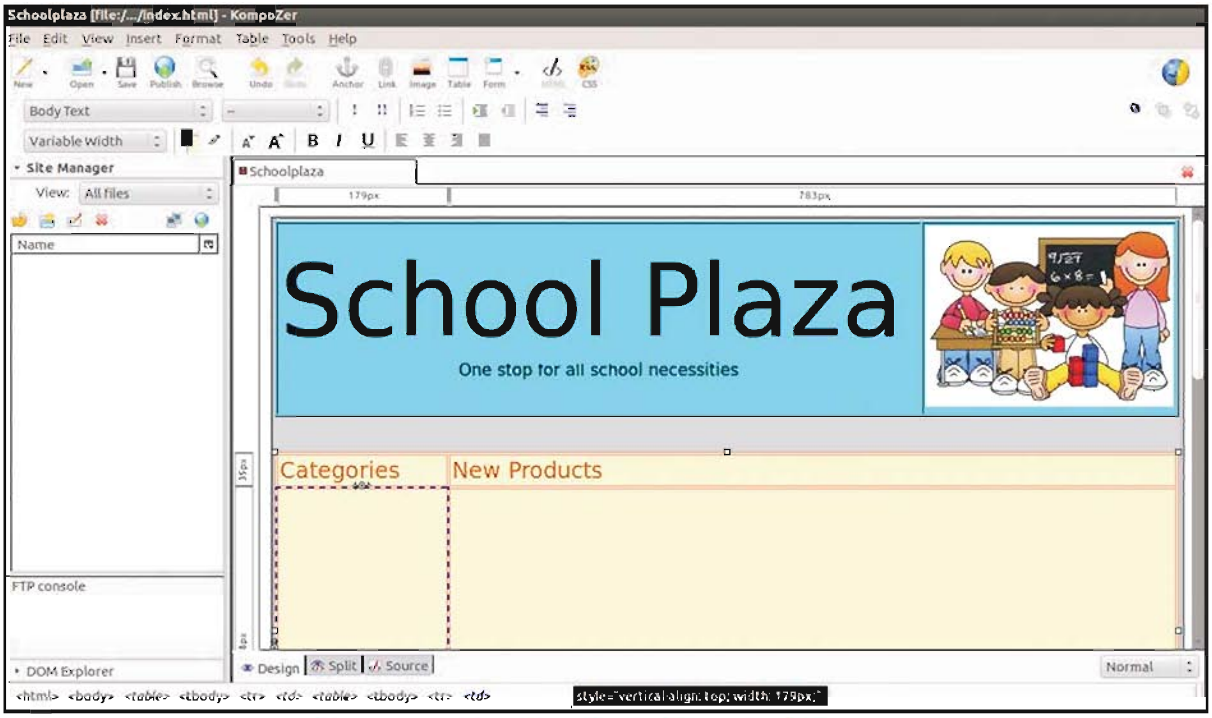
આકૃતિ 3.8 : Image Properties ડાયલોગબોક્સમાં Dimensions વિભાગ

આકૃતિ 3.9માં ઉમેરવામાં આવેલ લખાણ અને ચિત્ર દર્શાવ્યાં છે.



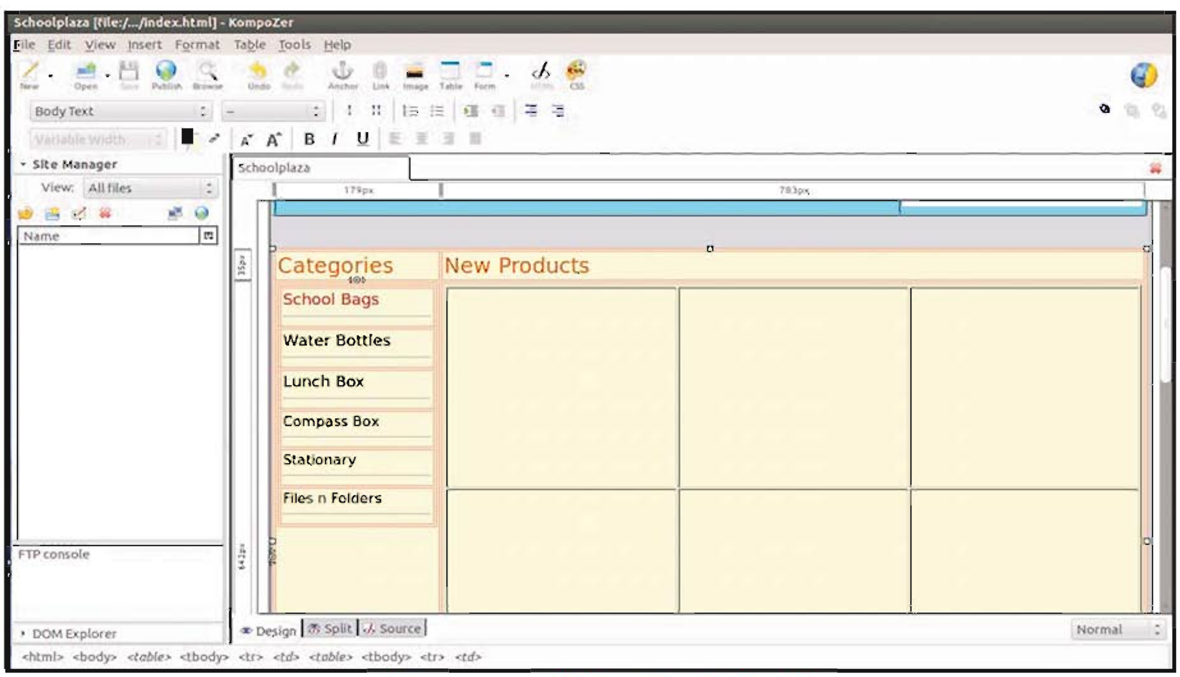
આકૃતિ 3.9 : ચિત્ર અને લખાણ સાથેનું કોષ્ટક

હવે, વર્ગો (Categories) અને નવાં ઉત્પાદનો (New Products) દર્શાવવા માટે 2 હરોળ અને 2 સ્તંભ ધરાવતું અન્ય કોષ્ટક ઉમેરીશું. પ્રથમ હરોળના સેલમાં "Categories" અને "New Products" લખાણ ટાઈપ કરો. **Format → Text color**નો ઉપયોગ કરી લખાણને યોગ્ય રંગ આપો. આકૃતિ 3.10માં લખાણ સાથેની હરોળ દર્શાવી છે.



આકૃતિ 3.10 : કોષ્ટકની હરોળમાં ઉમેરવામાં આવેલ લખાણ

ત્યાર પછી, બીજી હરોળમાં categoriesની નીચે એક સ્તંભ અને વધુ હરોળ સાથેનું એક અન્ય કોષ્ટક ઉમેરીશું. આ કોષ્ટકમાં સ્કૂલબૅગ, પાણીની બૉટલ અને અન્ય વસ્તુ માટેના દરેક વર્ગ એક-એક હરોળમાં ઉમેરીશું. લખાણને યોગ્ય રંગ આપો. અહીં વર્ગ છૂટા પાડવા માટે આપણે આડી લીટી પણ ઉમેરીશું. આડી લીટી ઉમેરવા માટે **Insert → Horizontal line** વિકલ્પનો ઉપયોગ કરવામાં આવે છે. વળી, વર્ગોને એવી રીતે લિંક આપવામાં આવશે કે જે આપણને પસંદ કરેલ વર્ગના સંબંધિત પાના પર લઈ જશે. વર્ગ પરની લિંક આપણે પછીથી બનાવીશું. હવે આપણે નવા ઉત્પાદનના સેલની નીચે વસ્તુની છબી અને તેને સંબંધિત વિગતો ઉમેરવા કોષ્ટકની રચના કરીએ. પાનાંમાં દર્શાવવા ઇચ્છતા હોઈએ તે વસ્તુ આધારિત કોષ્ટક ઉમેરો. અહીં આપણે 3 હરોળ અને 3 સ્તંભ સાથેનું કોષ્ટક ઉમેર્યું છે. વેબપેજમાં ઉમેરેલું કોષ્ટક આકૃતિ 3.11માં દર્શાવ્યું છે.



આકૃતિ 3.11 : વર્ગોની યાદી દર્શાવતી સ્ક્રીન

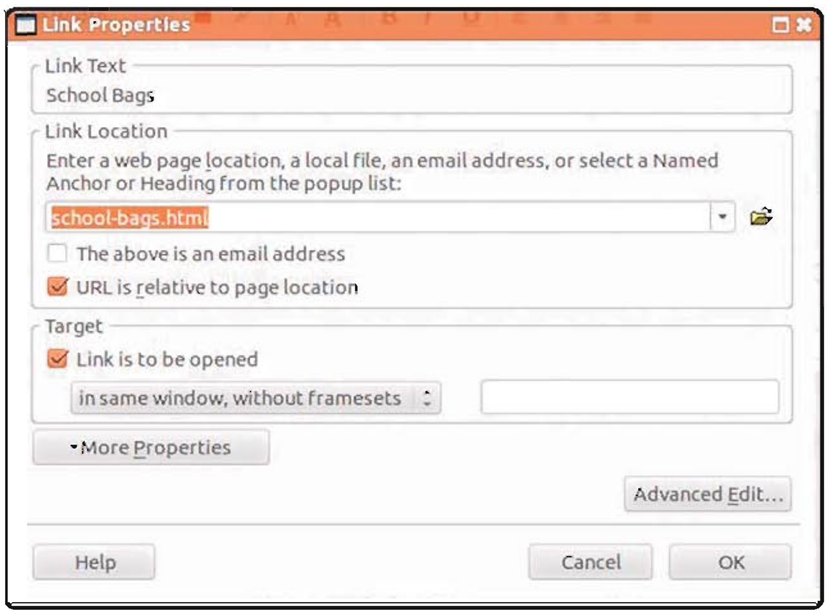
ત્યાર બાદ, પાનાના નીચેના ભાગમાં "About school plaza", "Feedback", "Contact us" અને "Site map"ની વિગતો માટે અન્ય એક કોષ્ટક ઉમેરીશું. કોષ્ટકની રચના કરી લખાણ ઉમેરો. આ લખાણ પણ લિંક બનશે.

આપણે લગભગ આખું હોમપેજ બનાવી લીધું છે. જ્યારે કોઈ પણ વર્ગ પર ક્લિક કરવામાં આવશે ત્યારે, પાનાંની ગોઠવણ સમાન રહેશે, પરંતુ તે વર્ગને અનુરૂપ ચિત્ર બદલાઈ જશે. આમ, હવે નવા વેબપેજની રચના કરવાની જરૂર નથી, પરંતુ હોમપેજની જ એકથી વધુ નકલો કરી જરૂર મુજબ તેમાં ફેરફાર કરી શકાશે.

Save As વિકલ્પ પસંદ કરી ફાઇલનો જુદાં-જુદાં નામથી સંગ્રહ કરો. અહીં ફાઇલનો સંગ્રહ waterbottles.html, school-bags.html અને lucnchbox.html નામ સાથે કરવામાં આવ્યો છે. હવે જ્યારે નિશ્ચિત કોઈ વર્ગના લિંક પર ક્લિક કરવામાં આવે, ત્યારે તેને સંબંધિત વેબપેજ ખૂલશે. તમે એ ધ્યાનમાં લીધું હશે કે બનાવેલાં તમામ વેબપેજને હોમપેજ જેવું જ શીર્ષક મળેલું છે. એક પછી એક વેબપેજ ખોલો અને **Format → Page Title** અને **Properties** વિકલ્પનો ઉપયોગ કરી દરેક વેબપેજનું શીર્ષક બદલો. હવે, દરેક વેબપેજ ખોલી તેના વર્ગને અનુરૂપ ચિત્રો ઉમેરો. હોમપેજમાં ચિત્ર ઉમેરવાની પદ્ધતિને અનુસરવાથી અન્ય તમામ વેબપેજમાં પણ ચિત્રો ઉમેરી શકાશે.

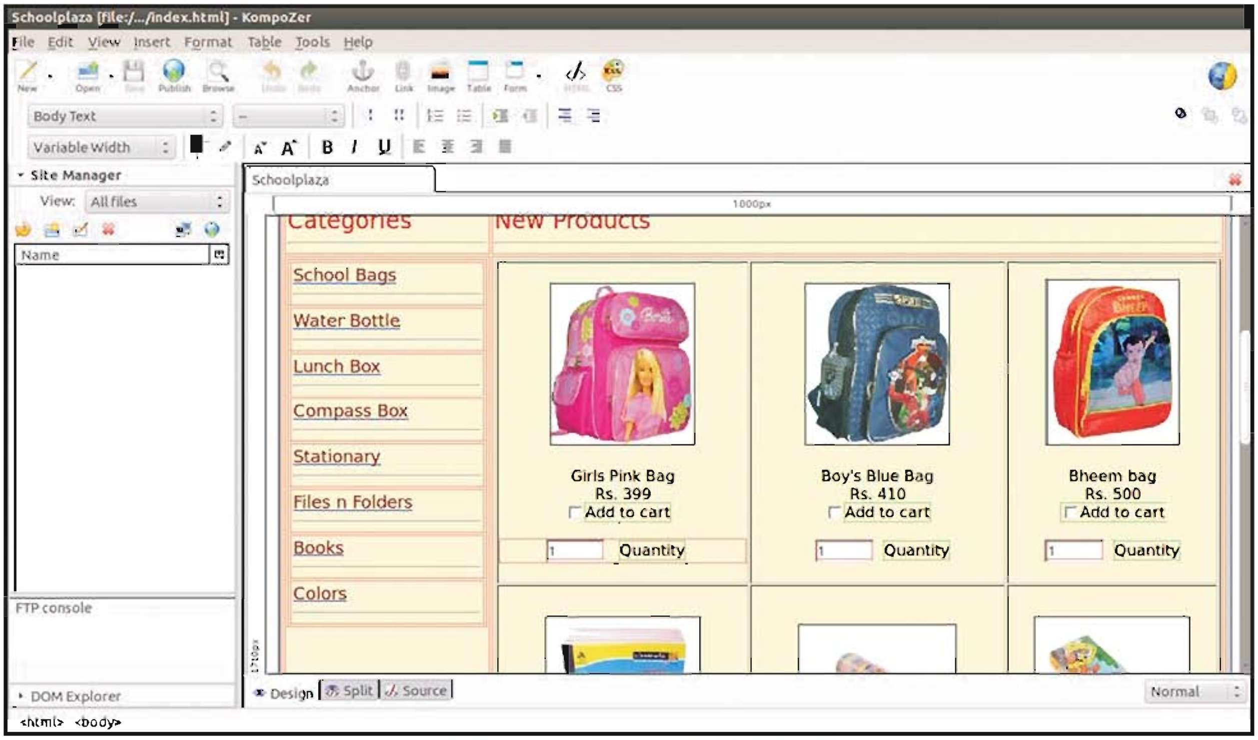
હોમપેજમાં વસ્તુઓનાં ચિત્ર અને તેની વિગતો ઉમેરતાં પહેલાં જુદા-જુદા વર્ગો માટે લિંકની રચના કરીએ. વર્ગનાં નામ પર ક્લિક કરવાથી તેને સંબંધિત વેબપેજ ખૂલશે.

index.html ફાઇલ ખોલો. લિંકની રચના કરવા માટે લખાણ (School-Bags) પસંદ કરો અને **Insert → Link** વિકલ્પ પસંદ કરો. તે આકૃતિ 3.12માં દર્શાવ્યા મુજબનું ડાયલોગબૉક્સ રજૂ કરશે. Link Location વિકલ્પ હેઠળ, જે ફાઇલ ખોલવાની છે, તે દર્શાવો (school-bags.html). OK બટન પર ક્લિક કરો.



આકૃતિ 3.12 : Link Properties ડાયલોગબૉક્સ

હવે, હોમપેજમાં નવા ઉત્પાદનોની વિગતો ઉમેરીએ. કોષ્ટકના દરેક સેલમાં વસ્તુનું ચિત્ર, વસ્તુનું નામ, કિંમત તથા Add to Cart ચેકબૉક્સ અને વસ્તુના જથ્થા માટેનું ટૅક્સ્ટબૉક્સ ઉમેરીશું. અગાઉ ઉપયોગમાં લીધેલી પદ્ધતિ મુજબ ચિત્ર ઉમેરો. ઉત્પાદનનું નામ અને કિંમત ઉમેરો. હવે, ફૉર્મફિલ્ડ દ્વારા એક ચેકબૉક્સ ઉમેરો. ચેકબૉક્સ માટે b1 નામની "id" HTML લાક્ષણિકતા ઉમેરીશું, જેનો પછીથી જાવાસ્ક્રિપ્ટમાં ઉપયોગ કરવામાં આવશે. "Add to cart" નામનું લેબલ ઉમેરો. પછીથી, શરૂઆતની કિંમત 1 અને "q1" id સાથે ટૅક્સ્ટબૉક્સ ઉમેરો. "Quantity" લેબલ ઉમેરો. જ્યારે ઉપયોગકર્તા Add to Cart ચેકબૉક્સ પસંદ કરશે, ત્યારે વસ્તુ અને તેના જથ્થાનો શૉપિંગ કાર્ટમાં સમાવેશ કરવામાં આવશે. ઉમેરવામાં આવેલ ઉત્પાદનોની વિગતો આકૃતિ 3.13માં દર્શાવી છે.



આકૃતિ 3.13 : ઉત્પાદન અંગેની વિગતો

નોંધ : આપણે બનાવેલ વર્ગ માટેનાં ચિત્રો વેબપેજમાં ઉમેરો. બાકીની વિશેષતાઓ સમાન રહેશે. અહીં અન્ય વેબપેજ દર્શાવવામાં આવ્યાં નથી.

પાનાંના નીચેના ભાગમાં "Purchase" બટન ઉમેરીશું, જે આપણને બિલની વિગતોના વેબપેજ તરફ લઈ જશે. આ પાનું ખરીદવામાં આવેલ કુલ વસ્તુઓની સંખ્યા અને કુલ રકમ દર્શાવે છે. બટનની onclick ઘટના સમયે કોડ લિસ્ટિંગ 3.1માં દર્શાવેલ જાવાસ્ક્રિપ્ટ ઉમેરીએ. તે oncart() વિધેયનો અમલ કરશે અને ત્યાર પછી oncart() વિધેય બિલ માટેનું વેબપેજ દર્શાવશે.

```
<script>
function oncart()
{
var check=false;
var cookievalue="";

if(document.getElementById("b1").checked)
{
check=true;
var quantity=document.getElementById("q1").value;
var price = 399;
var total=(price*quantity);
cookievalue+=":Girls pink bag,"+ price + "," + quantity + "," + total;
}

if(document.getElementById("b2").checked)
{
```

```
check=true;
var quantity=document.getElementById("q2").value;
var price = 410;
var total=(price*quantity);
cookievalue+=":Boy's Blue bag,"+ price + "," + quantity + "," + total;
}

if(document.getElementById("b3").checked)
{
check=true;
var quantity=document.getElementById("q3").value;
var price=500;
var total=(500*quantity);
cookievalue+=":Bheem bag,"+ price + "," + quantity + "," + total;
}

if(document.getElementById("nb1").checked)
{
check=true;
var quantity=document.getElementById("q4").value;
var total=(50*quantity);
cookievalue+=":NoteBook (set of 3),"+total;
}

if(document.getElementById("cp1").checked)
{
check=true;
var quantity=document.getElementById("q5").value;
var total=(45*quantity);
cookievalue+=":Color pencils,"+total;
}

if(document.getElementById("p1").checked)
{
check=true;
var quantity=document.getElementById("q6").value;
var total=(80*quantity);
cookievalue+=":Pencil Box,"+total;
}

if(document.getElementById("l1").checked)
{
```

```
check=true;
var quantity=document.getElementById("q7").value;
var total=(110*quantity);
cookievalue+=":Angry Bird Lunch box,"+total;
}

if(document.getElementById("l2").checked)
{
check=true;
var quantity=document.getElementById("q8").value;
var total=(120*quantity);
cookievalue+=":cartoon lunch box,"+total;
}

if(document.getElementById("e1").checked)
{
check=true;
var quantity=document.getElementById("q9").value;
var total=(50*quantity);
cookievalue+=":Eraser (set of 4),"+total;
}

if(!check)
{
alert("No item selected");
}
else
{
document.cookie=cookievalue;
window.location="bill.html";
}
}
  </script>
```

કોડલિસ્ટિંગ 3.1 : યથાર્થતા માટેની જાવાસ્ક્રિપ્ટ

કોડલિસ્ટિંગ 3.1માં oncart() વિધેય દર્શાવ્યું છે. પ્રથમ if વિધાન ચકાસે છે કે item1 માટેનું ચેકબૉક્સ પસંદ કરવામાં આવ્યું છે કે નહીં. આ પહેલાં આપણે item1ના ચેકબૉક્સને "b1" નામથી id આપ્યું છે. document.getElementById("b1").checked વિધાનનો ઉપયોગ કરી તપાસવામાં આવ્યું છે કે ઉપયોગકર્તાએ વસ્તુ પસંદ કરી છે કે નહીં. જો વસ્તુ પસંદ કરવામાં આવી હોય, તો તેનો જથ્થો ચલમાં સાચવી કુલ રકમની ગણતરી કરવામાં આવશે. અંતમાં, સ્ટ્રિંગ સ્વરૂપે વસ્તુનું નામ, કિંમત, જથ્થો અને કુલ રકમનો ચલમાં સંગ્રહ કરીશું. અહીં દરેક વિગતને અલ્પવિરામ (,) દ્વારા છૂટી પાડવામાં આવી છે. દરેક ઉત્પાદનની વિગતોને વિસર્ગ (:) દ્વારા છૂટી પાડવામાં આવી છે. યાદીમાં આવેલ દરેક વસ્તુ માટે સ્ક્રિપ્ટને આગળ વધારવામાં આવે છે.

જો ઉપયોગકર્તાએ એક પણ વસ્તુ પસંદ કરી ન હોય અને purchase બટન પર ક્લિક કરવામાં આવે, તો "No item is selected" લખાણ સાથેનો સંદેશ દર્શાવવામાં આવશે. નહીં તો, વસ્તુને 'કૂકી' (cookie) તરીકે સંગ્રહવામાં આવશે. ઉપયોગકર્તાના કમ્પ્યૂટરમાં સંગ્રહ કરવામાં આવતા ચલને 'કૂકી' કહે છે. તે માર્ગ યાદ રાખી મુલાકાતીના વધુ સારા અનુભવ માટે અથવા સાઇટની આંકડાવિષયક માહિતી માટે જરૂરી પસંદગીઓ, ખરીદી, વળતર અને અન્ય માહિતી પૂરી પાડે છે. જાવાસ્ક્રિપ્ટમાં document ઑબ્જેક્ટના Cookies ગુણધર્મ દ્વારા કૂકીનો ઉપયોગ કરી શકાય છે. કૂકીને વાંચી શકાય છે, રચી શકાય છે, સુધારી શકાય છે તથા દૂર કરી શકાય છે.

જ્યારે ઉપયોગકર્તા Purchase બટન પર ક્લિક કરે છે, ત્યારે જાવાસ્ક્રિપ્ટ દ્વારા "bill.html" પાનું દર્શાવવામાં આવે છે. bill.html પાનામાં જાવાસ્ક્રિપ્ટ દ્વારા પસંદ કરેલ વસ્તુઓ માટે કુલ વસ્તુઓની સંખ્યા સાથે ચૂકવવાની કુલ રકમની ગણતરી કરવામાં આવે છે. કુલ રકમની ગણતરી કરવા માટેની જાવાસ્ક્રિપ્ટ કોડલિસ્ટિંગ 3.2માં દર્શાવી છે.

અહીં myfun() નામનું વિધેય વ્યાખ્યાયિત કરવામાં આવ્યું છે. આ વિધેય કૂકી પાસેથી વિગતો લઈ તેને ઍરેમાં સંગૃહીત કરે છે. ત્યાર પછી વિગતોને parseFloat() વિધેય દ્વારા અંકોમાં ફેરવી સરવાળો કરવામાં આવે છે. અંતમાં સરવાળાને દર્શાવવામાં આવે છે.

```
function myfun()
{
var cookiearr=new Array();
cookiearr=document.cookie.split(":");
document.writeln("<center><h1>Welcome To Billing</h1></center>");
document.write("<center><table
border=3><thead><tr><th>Description</th><th>Price</th><th>Quantity</th><th>Total</th></tr></
thead>");
var total2=0;
var total3=0;

for(var i=1;i<cookiearr.length;i++)
{
var cookiedata=cookiearr[i].split(",");
document.writeln("<tr><td>"+cookiedata[0]+"</td><td>"+cookiedata[1]+"</td><td>"+cookiedata[2]
+"</td><td>"+cookiedata[3]+"</td></tr>");

total2+=parseFloat(cookiedata[2]);
total3+=parseFloat(cookiedata[3]);
}
document.write("<tr><td>Total</td><td>"+" "
+"</td><td>"+total2+"</td><td>"+total3+"</td></tr>");
document.write("</tbody></table></center>");
}
 </script>
```

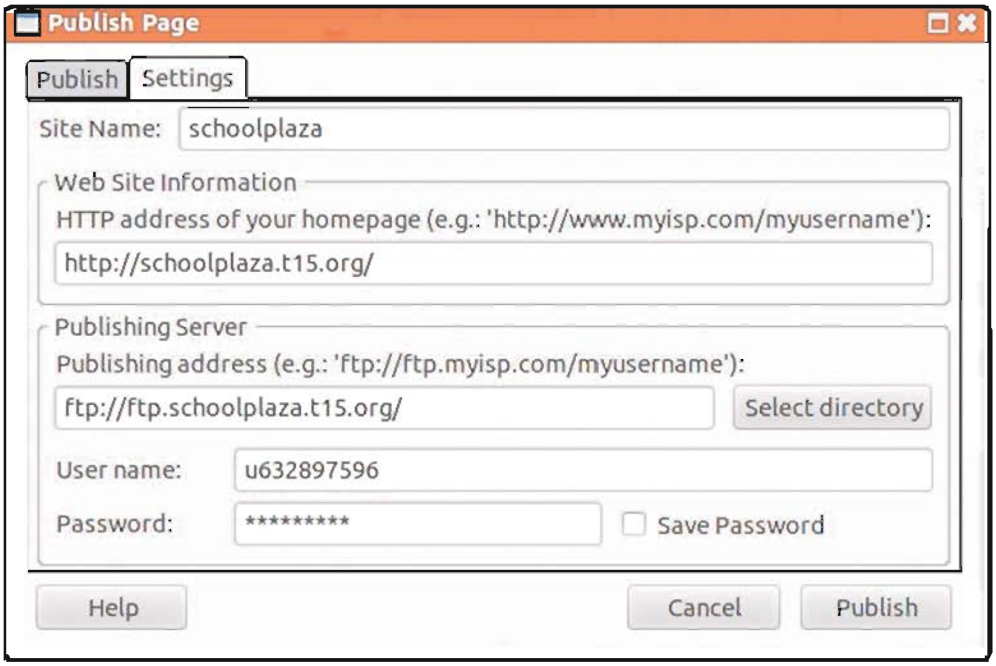
કોડલિસ્ટિંગ 3.2 : કુલ રકમ શોધવા માટેની જાવાસ્ક્રિપ્ટ

અંતમાં, ઉપયોગકર્તા બિલના વેબપેજ પર આવેલા Proceed to Checkout બટન પર ક્લિક કરશે. આ સમયે જો ઉપયોગકર્તાએ લોગ-ઇન કરેલું નહીં હોય, તો તેને તે વિશે પૂછવામાં આવશે અને જો ઉપયોગકર્તાની નોંધણી થયેલ ન હોય, તો તેને નવું ખાતું (account) બનાવવા માટે કહેવામાં આવશે. આ માટે આગળના પ્રકરણમાં ચર્ચા કરી તે મુજબ નોંધણી માટેનું ફૉર્મ (Registration form) બનાવી શકાય છે. ત્યાર પછી સંસ્થાના માલિક દ્વારા નક્કી કરવામાં આવેલ ચૂકવણીની વિગતો ઉમેરી શકાય. સામાન્ય રીતે, આ પ્રકારની વેબસાઇટના વ્યવસ્થાપન માટે ડેટાબેઝની રચના કરવાની જરૂર પડે છે, જેમાં ઉત્પાદન, નોંધણી થયેલાં ઉપયોગકર્તાઓ, લોગ-ઇન નામ, પાસવર્ડ અને અન્ય સંબંધિત વિગતોનો સંગ્રહ કરવામાં આવ્યો હોય છે.

વેબસાઈટ પ્રકાશિત કરવી (Publishing a Website)

અત્યાર સુધી આપણે કમ્પોઝરની મદદથી સરળ વેબસાઇટની રચના વિશે શીખ્યા. હવે વેબસાઇટને કેવી રીતે પ્રદર્શિત (publish) કરવી તે જોઈએ. વેબસાઇટને પ્રદર્શિત કે પ્રકાશિત કરવી એટલે કે ઉપયોગકર્તા સાઇટનાં વેબપેજ, ચિત્રો અને સ્ટાઇલશીટનો ઉપયોગ કરી શકે તે માટે તેને વેબસર્વર પર ખસેડવી. આ પ્રક્રિયાને સામાન્ય રીતે 'અપલોડિંગ' (Uploading) તરીકે ઓળખવામાં આવે છે.

જો વેબસર્વર પર આપણું ખાતું હોય, તો વેબસાઇટને અપલોડ કરી શકાય છે. ઇન્ટરનેટની સેવા પૂરી પાડનાર સંસ્થા (Internet Service Provider - ISP) વેબસર્વર પર મર્યાદિત જગ્યા પૂરી પાડે છે. વ્યાવસાયિક રીતે હોસ્ટિંગ પૂરું પાડતી સંસ્થાઓ પાસેથી જગ્યા ખરીદી પણ શકાય છે. વૈકલ્પિક રીતે, વેબસાઇટને અપલોડ કરવા માટે ઘણા વેબહોસ્ટ (web host) છે, જે ઉપયોગકર્તાને મર્યાદિત જગ્યા નિઃશુલ્ક પૂરી પાડે છે. આ નિઃશુલ્ક જગ્યા મર્યાદિત સમયમર્યાદા માટે પણ હોઈ શકે છે. આપણે બનાવેલ વેબસાઇટના ઉદાહરણમાં આવી જ નિઃશુલ્ક જગ્યા પૂરી પાડતી સાઇટનો ઉપયોગ કરી વેબસાઇટ અપલોડ કરી છે. સાઇટને પ્રદર્શિત કરવા સિસ્ટમની ગોઠવણી માટે તે જગ્યાનાં સેટિંગ જાણવાં જરૂરી છે. index.html વેબપેજ ખોલો અને **File → Publish** વિકલ્પ પસંદ કરો. આકૃતિ 3.14 દર્શાવ્યા મુજબ Publish Page ડાયલોગબૉક્સ રજૂ કરવામાં આવશે.



આકૃતિ 3.14 : Publish Pag e ડાયલોગબૉક્સ

Publish Page ડાયલોગબૉક્સમાં આપેલ ગોઠવણના વિભાગોમાં નીચેની વિગતો પૂરી પાડવામાં આવે છે :

1. Site name ફિલ્ડમાં વેબસાઇટનું નામ ઉમેરો. વેબસાઇટનો સંદર્ભ મેળવવા માટે આ નામનો ઉપયોગ કમ્પોઝર (KompoZer) દ્વારા માત્ર આંતરિક હેતુ માટે કરવામાં આવે છે.

2. Website informationમાં "HTTP address of your homepage" ફિલ્ડ વેબસાઇટનું પ્રત્યક્ષ સરનામું અથવા URL રજૂ કરે છે.

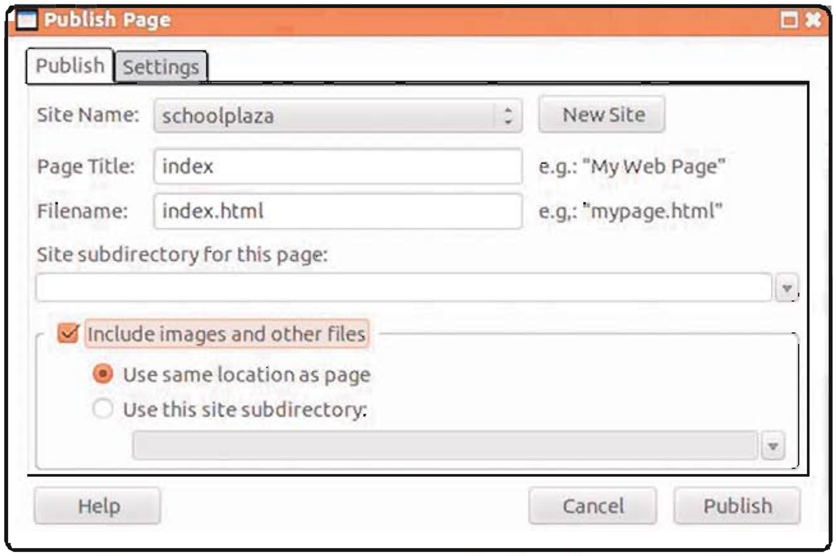
   નોંધ : આપણે ઇન્ટરનેટ પર ડોમેઇનની નોંધણી કરાવી શકીએ છીએ. ઇન્ટરનેટ પર એવી ઘણી વેબસાઇટ ઉપલબ્ધ છે, જેની મદદથી આપણે ડોમેઇન નામની નોંધણી કરી શકીએ છીએ. ઉદાહરણ તરીકે, સામાન્ય ખર્ચ ચૂકવી www.schoolplaza.com જેવું નામ મેળવી શકાય છે.

3. જગ્યા ખરીદવામાં આવી હોય તે ISP કે વેબહોસ્ટ દ્વારા વેબસાઇટ પ્રદર્શિત કરવા માટે વિગતો પૂરી પાડવામાં આવે છે. પ્રકાશન માટેનું સરનામું ટાઇપ કરો. જો FTP દ્વારા જોડાણ કરવામાં આવ્યું હોય, તો FTP સરનામું ટાઇપ કરો. આપણા કમ્પ્યૂટર પરથી વેબહોસ્ટ પર ફાઇલોનાં સ્થાનાંતરણ માટે File Transfer Protocol (FTP)નો ઉપયોગ

કરવામાં આવે છે. અગાઉ ચર્ચા કરી તે મુજબ, આ પ્રક્રિયાને અપલોડિંગ કે પ્રકાશન (publishing) કહે છે. FTP યૂઝરનેમ અને પાસવર્ડ આપો.

નોંધ : વેબહોસ્ટ દ્વારા પૂરી પાડવામાં આવેલી નિ:શુલ્ક જગ્યાની વિગતો આકૃતિ 3.14માં દર્શાવી છે.

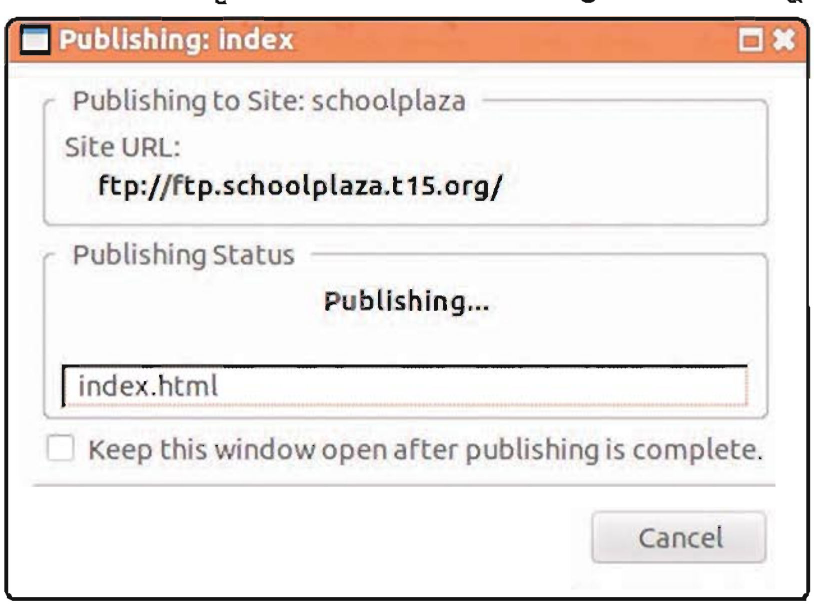
આકૃતિ 3.14માં આવેલા Publish વિભાગ પર ક્લિક કરો. આમ કરવાથી આકૃતિ 3.15માં દર્શાવેલ ડાયલોગબૉક્સ રજૂ થશે.



આકૃતિ 3.15 : Publish Page ડાયલોગબૉક્સનો Publish વિભાગ

હવે, નીચે દર્શાવેલ પગલાંને અનુસરો :

1. જો પસંદ થયેલ ન હોય, તો ડ્રૉપડાઉન મેનુમાંથી સાઇટનું નામ પસંદ કરો.
2. Page Title ઉમેરો, તે index.htmlના શીર્ષકનો સંદર્ભ આપે છે.
3. ફાઇલનું નામ આપો. અહીં index.html ફાઇલ ખૂલેલી હોવાને કારણે તે આપોઆપ ફાઇલનું નામ લઈ લેશે.
4. જો વેબપેજ સાઇટની પેટા ડિરેક્ટરીમાં હોવું જરૂરી હોય, તો તેને "Site subdirectory" ફિલ્ડમાં ઉમેરો. કેટલીક હોસ્ટ સંસ્થાઓની એ જરૂરિયાત હોય છે કે વેબપેજને public_html જેવી સ્વતંત્ર ડિરેક્ટરીમાં રાખવામાં આવે.
5. ચિત્રો માટેની ડિરેક્ટરી નિશ્ચિત કરો. સામાન્ય રીતે ચિત્રોને તે જ ડિરેક્ટરીમાં રાખવામાં આવે છે.
6. Publish બટન પર ક્લિક કરો. આકૃતિ 3.16માં દર્શાવેલ Publishing ડાયલોગબૉક્સ રજૂ કરવામાં આવશે.



આકૃતિ 3.16 : Publishing ડાયલોગબૉક્સ

હવે કમ્પોઝર (KompoZer) ફાઈલોને અપલોડ કરશે અને અંતમાં Pop-up સંદેશ પ્રદર્શિત કરશે. અપલોડિંગ થઈ ગયા બાદ, બ્રાઉઝરમાં URL ટાઈપ કરી વેબસાઇટને તપાસી લો.

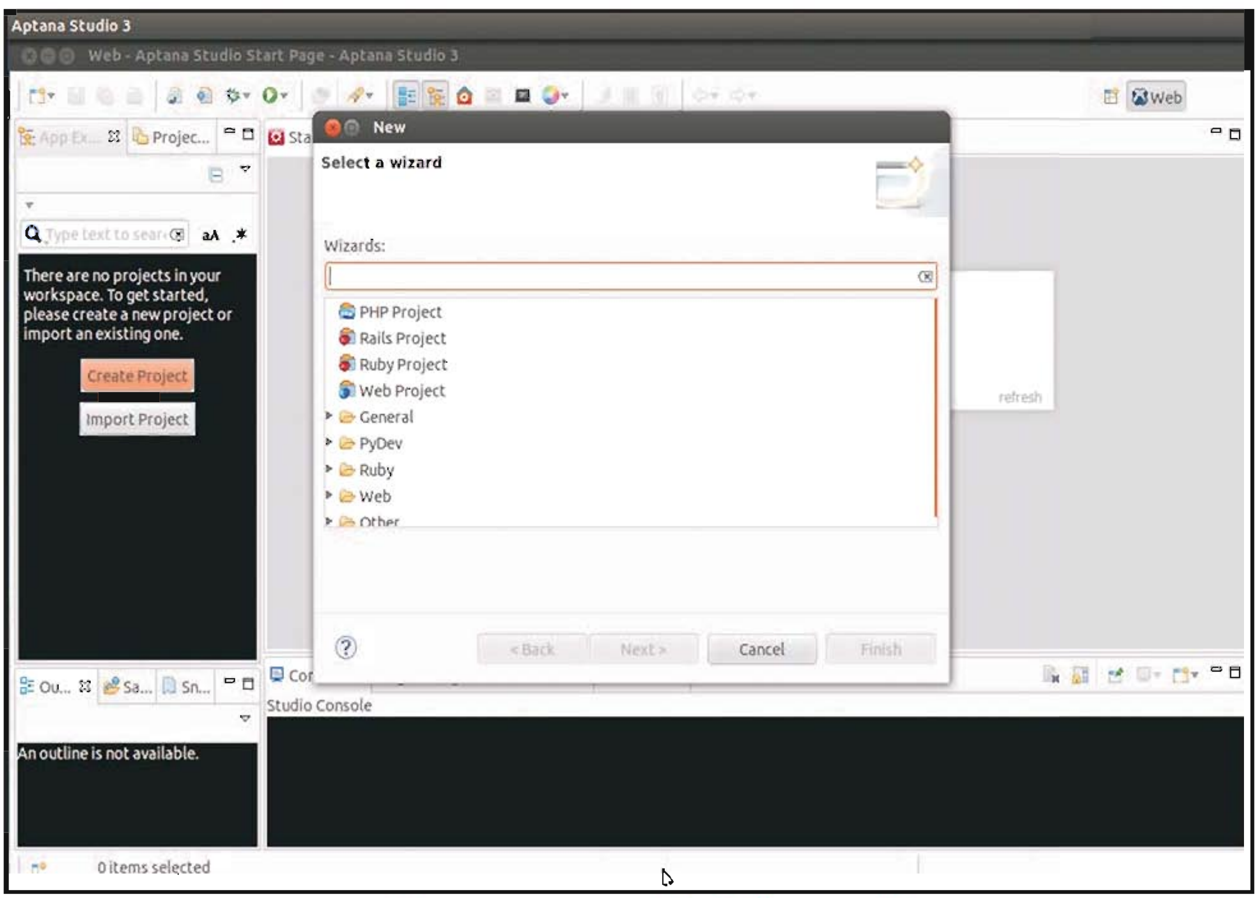
વેબવિકાસ માટેનાં અન્ય ઑપનસૉર્સ ટૂલ્સ (Other Open Source Web Development Tools)

કમ્પોઝરની મદદથી વેબસાઇટની રચના કેટલી સરળતાપૂર્વક કરી શકાય, તેનો આપણે અભ્યાસ કર્યો. આપણે અગાઉ જોયું તે મુજબ કમ્પોઝર એ વેબવિકાસ માટેનું એક નિ:શુલ્ક અને ઑપનસૉર્સ IDE છે. કમ્પોઝરની જેમ જ વેબવિકાસ માટેનાં ઘણાં નિ:શુલ્ક ટૂલ્સ ઇન્ટરનેટ પર ઉપલબ્ધ છે.

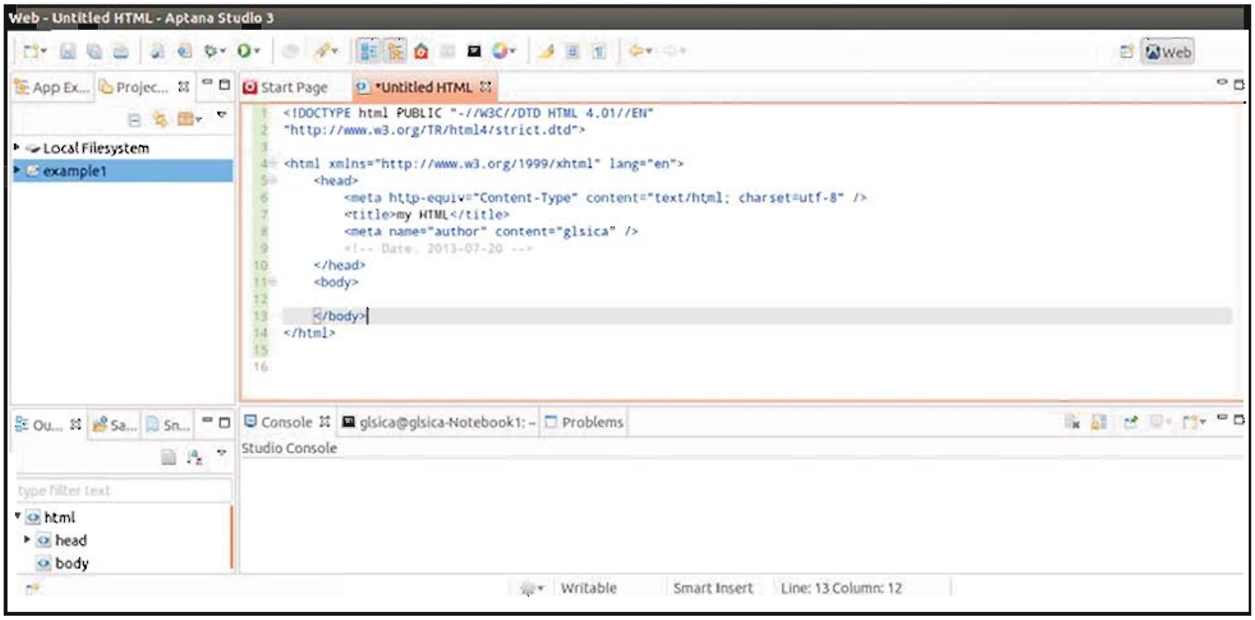
નેટ પર સરળતાથી ઉપલબ્ધ હોય અને વેબ ડેવલપર દ્વારા વેબસાઇટના વિકાસ માટે સરળતાથી ઉપયોગમાં લઈ શકાય તેવાં કેટલાંક ઑપનસૉર્સ ટૂલ્સ વિશે ચર્ચા કરીએ.

અપ્ટાના સ્ટુડિયો (Aptana Studio)

વેબવિનિયોગ બનાવવા માટે અપ્ટાના સ્ટુડિયો એક સક્ષમ ઑપનસૉર્સ IDE (Integrated Development Environment) છે. તે HTML, CSS, JavaScript, Ruby, Rails, PHP, Python અને અન્ય ઘણી ભાષાઓને સમર્થન આપતું એક સંપૂર્ણ વેબવિકાસ માટેનું વાતાવરણ પૂરું પાડે છે. તે વિપુલ સંખ્યામાં વધારાના પ્લગ-ઇન સાથે આપવામાં આવે છે. અપ્ટાના સ્ટુડિયો 2.0.5 (જે અપ્ટાના સ્ટુડિયો 2 પણ કહેવાય છે)નો ઉપયોગ HTML, CSS અને JavaScriptની મદદથી વેબવિનિયોગોના વિકાસ માટે થાય છે. www.aptana.com વેબસાઇટ પરથી તે સરળતાથી ડાઉનલોડ કરી શકાય છે. અપ્ટાના સ્ટુડિયોનો દેખાવ આકૃતિ 3.17માં દર્શાવ્યો છે. આકૃતિમાં દર્શાવ્યા મુજબ તેની મદદથી PHP, Rails, Ruby કે Webમાં પ્રોજેક્ટ બનાવી શકાય છે. આ ઇન્ટરફેસમાં લખેલ HTML કોડ આકૃતિ 3.18માં દર્શાવેલ છે.



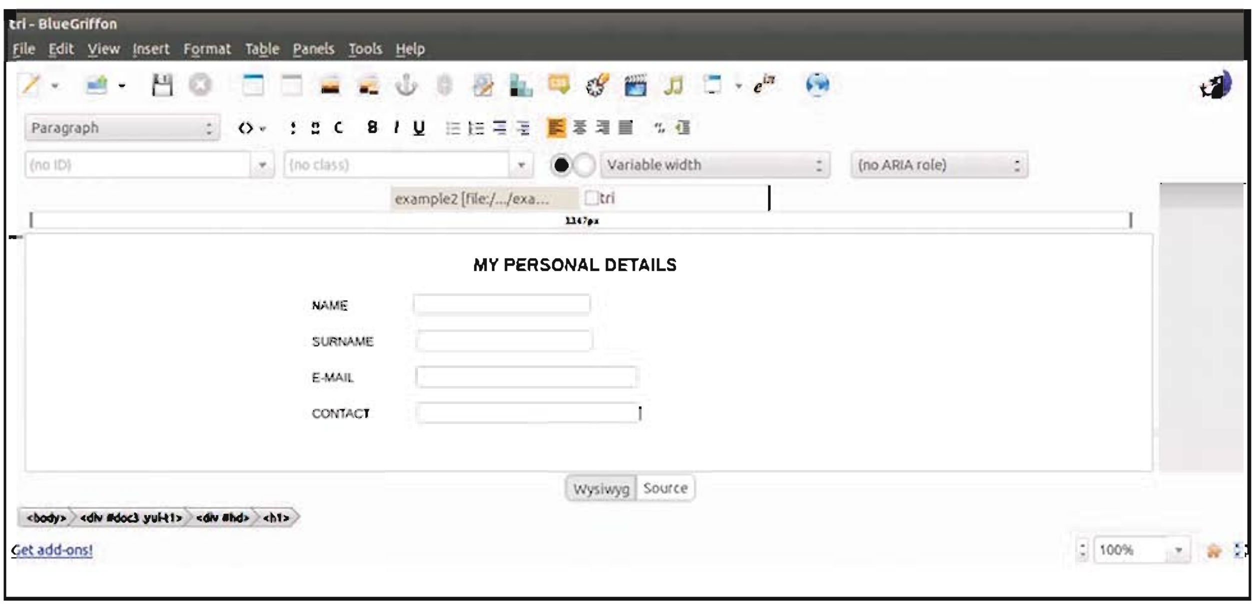
આકૃતિ 3.17 : અપ્ટાના સ્ટુડિયોમાં પ્રોજેક્ટ બનાવવો



આકૃતિ 3.18 : અપ્ટાના સ્ટુડિયોનો ઇન્ટરફેસ

બ્લૂગ્રિફોન (BlueGriffon)

બ્લૂગ્રિફોન એક અન્ય ઑપનસૉર્સ WYSIWYG પ્રકારનું HTML એડિટર છે. તેને www.bluegriffon.org પરથી સરળતાથી ડાઉનલોડ કરી શકાય છે. તે અંગ્રેજી, ડચ, જર્મન, ચાઈનીઝ અને અન્ય ઘણી ભાષાઓનું સમર્થન કરે છે. વેબ-ધારાધોરણના ગહન તક્નિકી જ્ઞાનના અભાવમાં પણ આકર્ષક વેબસાઇટની રચના કરી શકાય, તેવું સરળ ઇન્ટરફેસ પૂરી પાડતો એક સાહજિક વિનિયોગ છે. બ્લૂગ્રિફોનની આવૃત્તિ 1.6.2નો ઇન્ટરફેસ આકૃતિ 3.19માં દર્શાવ્યો છે. તેમાં ટૂલ્સનો ઉપયોગ કરી સરળ ફૉર્મની રચના કરવામાં આવી છે.

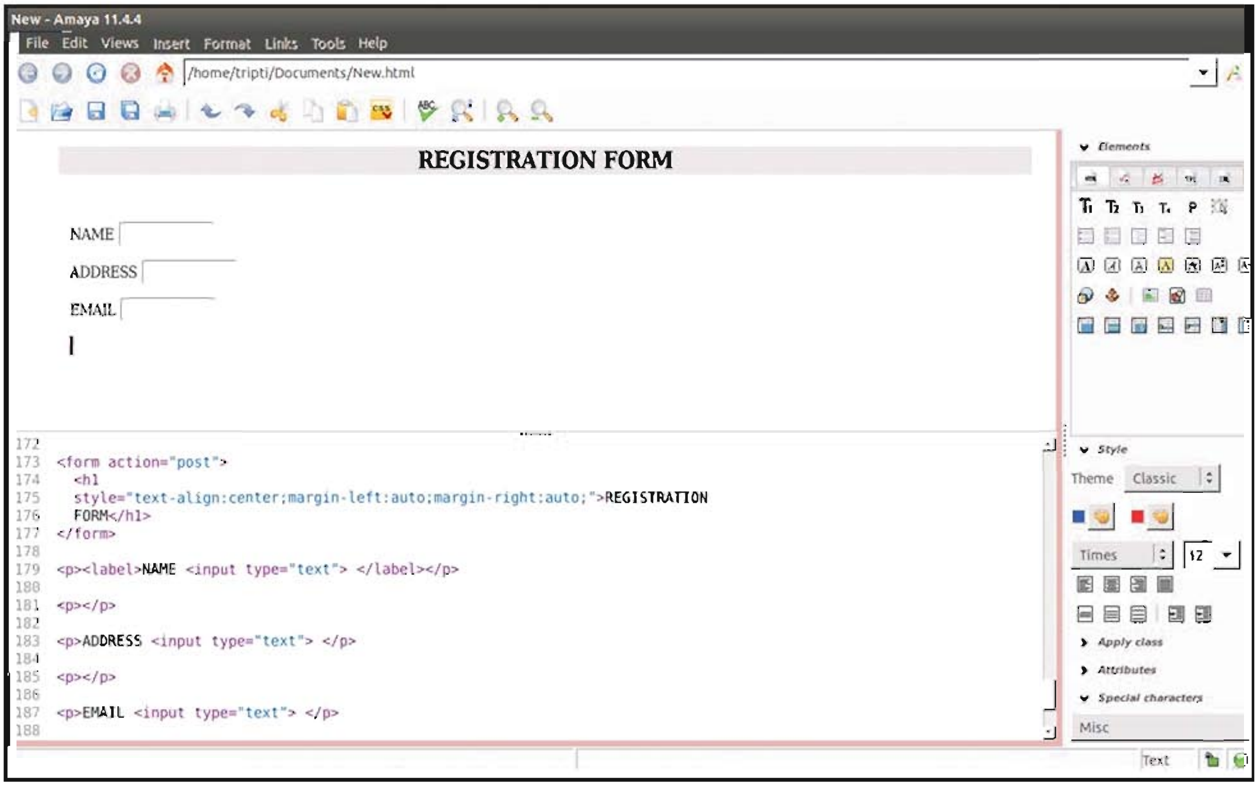


આકૃતિ 3.19 : બ્લૂગ્રિફોનનો ઉપયોગ કરી વેબપેજની રચના

આમાયા (Amaya)

World Wide Web Consortium (W3C) દ્વારા વિકસિત આમાયા એક અન્ય નિઃશુલ્ક, ઑપનસૉર્સ WYSIWYG પ્રકારનું વેબ-એડિટર છે. તેની શરૂઆત HTML/CSS એડિટર તરીકે કરવામાં આવી હતી અને હવે તે ઘણી

XML-આધારિત સિસ્ટમ માટેના એડિટર તરીકે વિકાસ પામ્યું છે. www.w3.org/Amaya પરથી તેને સરળતાથી ડાઉનલોડ કરી શકાય છે. આકૃતિ 3.20માં આમાયા દ્વારા રચિત વેબપેજ દર્શાવ્યું છે. વેબપેજની નીચે જ તેના સૉર્સને પણ જોઈ શકાય છે.



આકૃતિ 3.20 : આમાયાનો ઉપયોગ કરી રચવામાં આવેલું વેબપેજ

સારાંશ

આ પ્રકરણમાં આપણે કમ્પોઝરની મદદથી સરળ વેબસાઇટની રચના વિશે ચર્ચા કરી. વેબસાઇટ બનાવ્યા પછી તેને ઇન્ટરનેટ પર અપલોડ કરવી જરૂરી છે. વેબસાઇટને પ્રદર્શિત કરવી એટલે તેનાં પાનાં, ચિત્રો અને સ્ટાઇલશીટને વેબસર્વર પર ખસેડવાં, જેથી ઉપયોગકર્તા તેનો ઉપયોગ કરી શકે. આ પ્રક્રિયા 'અપલોડિંગ' તરીકે ઓળખાય છે. વેબસર્વર પર ખાતું (account) હોય, તો વેબસાઇટને અપલોડ કરી શકાય છે. ઇન્ટરનેટની સેવા પૂરી પાડનાર સંસ્થા (Internet Service Providers - ISP) વેબસર્વર પર મર્યાદિત જગ્યા પૂરી પાડે છે. વ્યાવસાયિક હોસ્ટિંગ સંસ્થાઓ પાસેથી જગ્યા ખરીદી પણ શકાય છે. આપણે Aptana studio, BlueGriffon અને Amaya જેવાં જુદાં-જુદાં નિ:શુલ્ક, ઑપનસૉર્સ વેબવિકાસ માટે ઉપલબ્ધ IDE વિશે પણ ચર્ચા કરી.

સ્વાધ્યાય

1. વ્યવસાયમાં વેબસાઇટ શા માટે અગત્યનો ભાગ ભજવે છે ?
2. સારી વેબસાઇટના આયોજનની પ્રક્રિયામાં ધ્યાનમાં રાખવા જેવા મુદ્દાઓની યાદી બનાવો.

3. વેબસાઇટ બનાવવા માટેનો હેતુ શા માટે સ્પષ્ટપણે વ્યાખ્યાયિત થયેલ હોવો જોઈએ ?

4. વેબસાઇટની રચના કરતી વખતે અપેક્ષિત પ્રેક્ષકગણ વિશેનું જ્ઞાન કઈ રીતે મદદરૂપ બને છે ?

5. વેબસાઇટની રચના વખતે કયા વિષયવસ્તુનો સમાવેશ કરવો જોઈએ ?

6. વેબસાઇટના હોમપેજને શા માટે index.html નામ આપવું જોઈએ ?

7. કોષ્ટકની લાક્ષણિકતાઓ ગોઠવવા માટે કયા વિકલ્પ ઉપલબ્ધ છે ?

8. કૂકી એટલે શું ?

9. વેબસાઇટનું પ્રકાશન (publishing) એટલે શું ?

10. વેબવિકાસ માટે ઉપલબ્ધ કેટલાંક ઓપનસોર્સ ટૂલ્સનાં નામ આપો.

11. આપેલ વિકલ્પમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) વ્યવસાયને આગળ વધારવા, ઉત્પાદન વેચવા અને વિશાળ પ્રમાણમાં ગ્રાહકોને આકર્ષવા માટે નીચેનામાંથી શું મદદરૂપ બને છે ?

(a) વેબસાઇટ (b) વેબપેજ (c) ફૉર્મ (d) CSS

(2) વેબસાઇટની વિકાસપ્રક્રિયામાં નીચેનામાંથી કયો મુદ્દો અગત્યનો નથી ?

(a) હેતુ (b) પ્રેક્ષકગણ (c) વિષયવસ્તુ (d) નિવેશ

(3) વેબસાઇટમાં કેવી માહિતી હોવી જોઈએ ?

(a) સંપૂર્ણ, સંબંધિત (b) સંપૂર્ણ, અસંબંધિત

(c) અપૂર્ણ, અસંબંધિત (d) અપૂર્ણ, સંબંધિત

(4) નીચેનામાંથી કયા પ્રકારની વિષયવસ્તુ વેબસાઇટ, સંસ્થા, ઉત્પાદન, સેવા અને અન્ય વસ્તુઓનું વિહંગાવલોકન પૂરું પાડે છે ?

(a) વિસ્તૃત (b) લાંબી (c) સામાન્ય (d) ટૂંકી

(5) વેબપેજના આંતરજોડાણને શું કહે છે ?

(a) વેબપેજ (b) ફૉર્મ (c) કમ્પોઝર (d) વેબસાઇટ

(6) બ્રાઉઝરના ઍડ્રેસબારમાં URL ઉમેરવામાં આવે, ત્યારે ખોલવામાં આવતા પ્રથમ પાનાંને શું કહે છે ?

(a) હોમપેજ (b) અંતિમ પેજ (c) વેબપેજ (d) પ્રથમ પેજ

(7) વેબસાઇટના હોમપેજને કયું નામ આપી સંગ્રહ કરવો જોઈએ ?

(a) first.html (b) index.html (c) home.html (d) one.html

(8) ઉપયોગકર્તાના કમ્પ્યૂટરમાં સંગ્રહ કરવામાં આવતા ચલને શું કહે છે ?

(a) Integer (b) HTML (c) Cookie (d) Java

(9) FTPનું પૂરું નામ શું છે ?

(a) File Truncate Protocol (b) File Transfer Process

(c) Fine Tune Protocol (d) File Transfer Protocol

પ્રાયોગિક સ્વાધ્યાય

1. કમ્પોઝરનો ઉપયોગ કરી તમારી શાળાની વેબસાઇટ બનાવો.
2. કમ્પોઝરનો ઉપયોગ કરી 'ગરવી ગુજરાત' વિષય પર વેબસાઇટ બનાવો.
3. કમ્પોઝરનો ઉપયોગ કરી 'ગુજરાત ટૂરિઝમ' વિષય પર વેબસાઇટ બનાવો.
4. કમ્પોઝરનો ઉપયોગ કરી રમકડાંની દુકાન માટે ઇ-કૉમર્સની વેબસાઇટ બનાવો.

ઇ-કૉમર્સનો પરિચય 4

આપણે માહિતીના યુગમાં જીવી રહ્યા છીએ. રેડિયો, ટેલિવિઝન, વર્તમાનપત્ર અને ઇન્ટરનેટ જેવા સ્રોત પરથી માહિતી ઉપલબ્ધ થતી હોય છે. ઇન્ટરનેટ અને મોબાઇલ સાધનોએ માહિતીનો ઉપયોગ કરવાની આપણી રીતો બદલી નાખી છે. લોકોએ દિવસના લગભગ દરેક કલાકે ઇન્ટરનેટનો ઉપયોગ કરવાનું શરૂ કર્યું છે. પહેલાંની વેબસાઇટનો મુખ્ય ઉપયોગ ઉત્પાદન અને સંસ્થા અંગેની માહિતીના પ્રસારણ માટે થતો. જેમકે, શૈક્ષણિક સંસ્થાની વેબસાઇટ વિવિધ પાઠ્યક્રમ, પાઠ્યક્રમનું વિષયવસ્તુ તથા વિદ્યાર્થીઓ અને શિક્ષકો અંગેની માહિતી પૂરી પાડે છે. એ જ રીતે વ્યાવસાયિક સંસ્થાઓ તેમનાં ઉત્પાદનો, ઉત્પાદનની વિશેષતાઓ, તેમના પુરવઠાકાર (supplier), ઑર્ડર આપવાની રીત વગેરે માહિતી પૂરી પાડે છે.

છેલ્લા કેટલાક દાયકાઓમાં ટેલિકૉમ્યુનિકેશન માળખાં તથા સૉફ્ટવેર તક્નિકમાં થયેલા ધરખમ વિકાસને કારણે ઇન્ટરનેટ વ્યાવસાયિક સંસ્થાઓ જેટલું જ વ્યક્તિગત રીતે પણ ઘણું લોકપ્રિય બન્યું છે. આજકાલ ઇન્ટરનેટે વ્યવસાયના સંચાલનની પદ્ધતિમાં ક્રાંતિકારી ફેરફારો કર્યા છે. બિલની ચુકવણી, બૅન્કને લગતાં કાર્યો અને ખરીદી જેવા વિવિધ હેતુઓ માટે લોકો ઇન્ટરનેટનો ઉપયોગ કરતા થયા છે. ઉત્પાદનનું માર્કેટિંગ અને વેચાણ, સૂચિપત્રો દર્શાવવા, શૅરની લે-વેચ અને ગ્રાહક-સેવા જેવી પ્રવૃત્તિઓ માટે વ્યાવસાયિક સંસ્થાઓ ઇન્ટરનેટનો ઉપયોગ કરે છે, જેને ઇ-કૉમર્સ તરીકે ઓળખવામાં આવે છે.

ઇન્ટરનેટ જેવા ઇલેક્ટ્રૉનિક માધ્યમનો ઉપયોગ કરી ઉત્પાદનનું ખરીદ-વેચાણ, સેવા અને માહિતી પૂરી પાડવાના કાર્યને પણ ઇ-કૉમર્સ તરીકે વ્યાખ્યાયિત કરી શકાય. ઇ-કૉમર્સ એ સંસ્થાઓ, વેપારીઓ અને ગ્રાહકોને તેમના ખર્ચમાં રાહત આપી માલ અને સેવાની ગુણવત્તા વધારવામાં સહાયભૂત થનારી આધુનિક પદ્ધતિ છે. વિતરણની ઝડપમાં પણ તે વધારો કરે છે. તે લોકોને સમય અને અંતરના અવરોધથી ઉગારી વૈશ્વિક બજાર અને વ્યાવસાયિક તકોનો લાભ લેવાની સુવિધા પૂરી પાડે છે.

વેબસાઇટ દ્વારા ઉત્પાદનનું વેચાણ એ વિશ્વભરમાં સૌથી ઝડપી પ્રગતિ કરી રહેલ વેપારની પદ્ધતિ છે. પુસ્તકો, ઇલેક્ટ્રૉનિક સાધનો, મોટરકાર, પ્રવાસન યોજનાઓ (holiday packages) અને બીજાં અનેક પ્રકારનાં ઉત્પાદનો અને સેવાઓનો વ્યાપાર ઑનલાઇન થાય છે. ઘણી વ્યાવસાયિક સંસ્થાઓએ તેમની ઇ-કૉમર્સ આધારિત પ્રવૃત્તિઓ માટેની વેબસાઇટ સ્થાપિત કરી છે. ઇ-કૉમર્સે આખા વિશ્વને એક વૈશ્વિક સ્થાન બનાવી દીધું છે, જ્યાંથી કોઈ પણ વ્યક્તિ કોઈ પણ સ્થળેથી કંઈ પણ ખરીદી શકે છે.

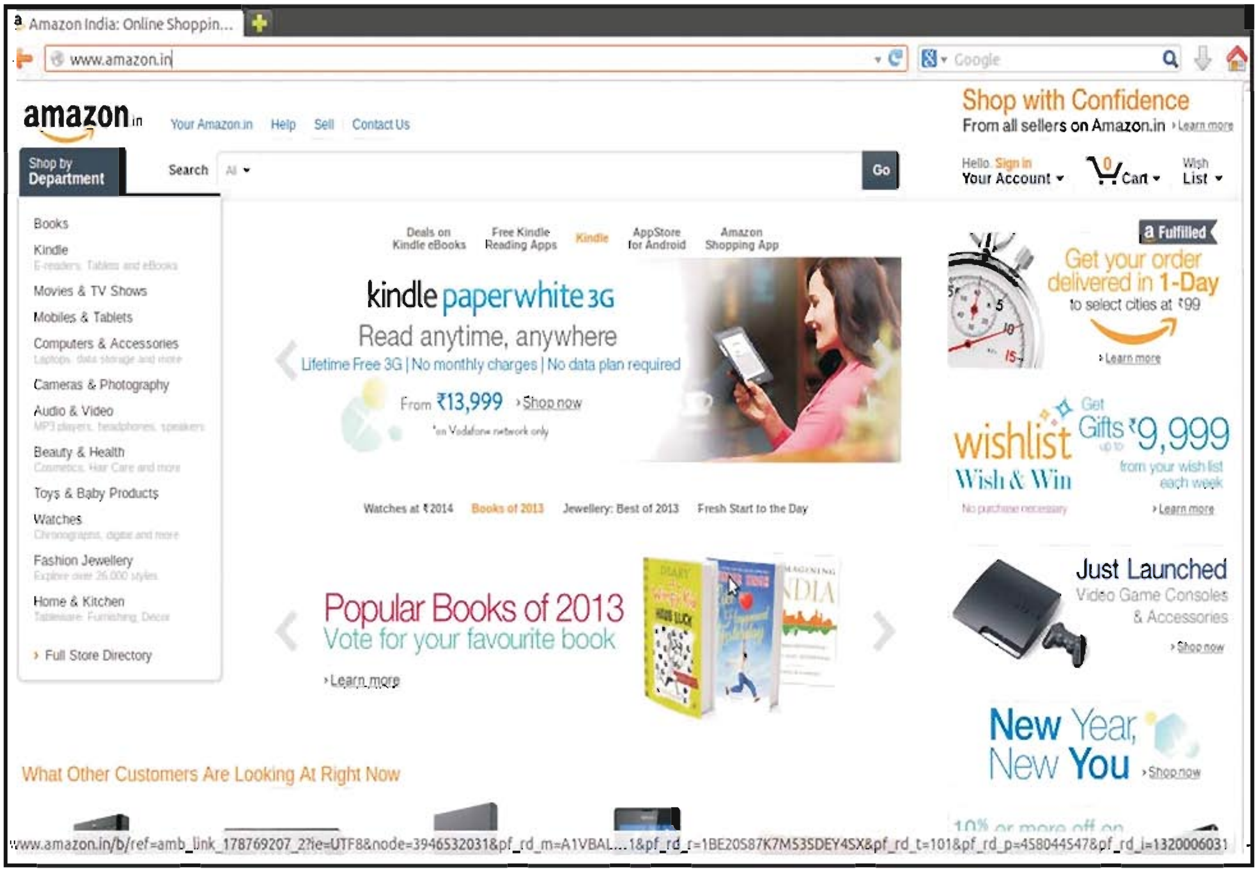
ઇ-કૉમર્સના વિનિયોગ (Applications of E-commerce)

આજકાલ જે વ્યાવસાયિક પ્રવૃત્તિઓમાં ઇ-કૉમર્સનો બહોળો ઉપયોગ કરવામાં આવે છે, તેમાં માલનો વેપાર એટલે કે, માર્કેટિંગ અને વેચાણ, માલની હરાજી તથા બૅંકિંગ અને વીમા જેવી નાણાકીય પ્રવૃત્તિઓનો સમાવેશ થાય છે. આ પ્રવૃત્તિઓની ટૂંકમાં ચર્ચા કરીએ. ઇ-કૉમર્સ અહીં ચર્ચા કરવામાં આવેલ પ્રવૃત્તિઓ પૂરતું જ મર્યાદિત નથી, પરંતુ વ્યવસાય અને અન્ય પ્રવૃત્તિઓ સાથે સંકળાયેલા અન્ય વિકાસશીલ વિસ્તારોમાં પણ વિસ્તર્યું છે.

ઇન્ટરનેટ પર પુસ્તકની દુકાન (Internet Bookshops)

ઇન્ટરનેટ પર ઇ-કૉમર્સનો આ સૌપ્રથમ વિનિયોગ હતો. ગ્રાહકો ઇન્ટરનેટ પર પુસ્તકો ખરીદવાનું પસંદ કરે છે, કારણ કે પુસ્તકોને ભૌતિક રીતે તપાસવાની જરૂર પડતી નથી અને તેનું સરળતાથી વર્ણન કરી શકાય છે. પુસ્તકોને ગ્રાહકો સુધી ક્ષતિરહિત કે અલ્પ ક્ષતિસહિત સરળતાપૂર્વક પહોંચાડી શકાય છે. ઑનલાઇન પુસ્તક-વિક્રેતાને પુસ્તકોની વર્ગીકૃત

યાદી, મુખપૃષ્ઠનાં ચિત્ર, પુસ્તકનું વર્ણન, પાનાંની કુલ સંખ્યા, પુસ્તકની કિંમત, વળતર અને ગ્રાહકોના પ્રતિભાવ દર્શાવવા સારી વેબસાઇટની જરૂર પડે છે. પુસ્તકનાં શીર્ષક, લેખકનાં નામ કે પ્રકાશકનાં નામ દ્વારા પુસ્તકને શોધી શકાય છે. પુસ્તકોનો ઑનલાઇન વિક્રેતા ગ્રાહકોની અભિરુચિની નોંધ રાખે છે અને તેમનો રસ જળવાઈ રહે તે માટે તેમને નવાં પુસ્તકોની જાણકારી આપતા રહે છે. ઑનલાઇન પુસ્તકોની દુકાનોમાંથી પ્રથમ એવી www.amazon.comનું હોમપેજ આકૃતિ 4.1માં દર્શાવ્યું છે.



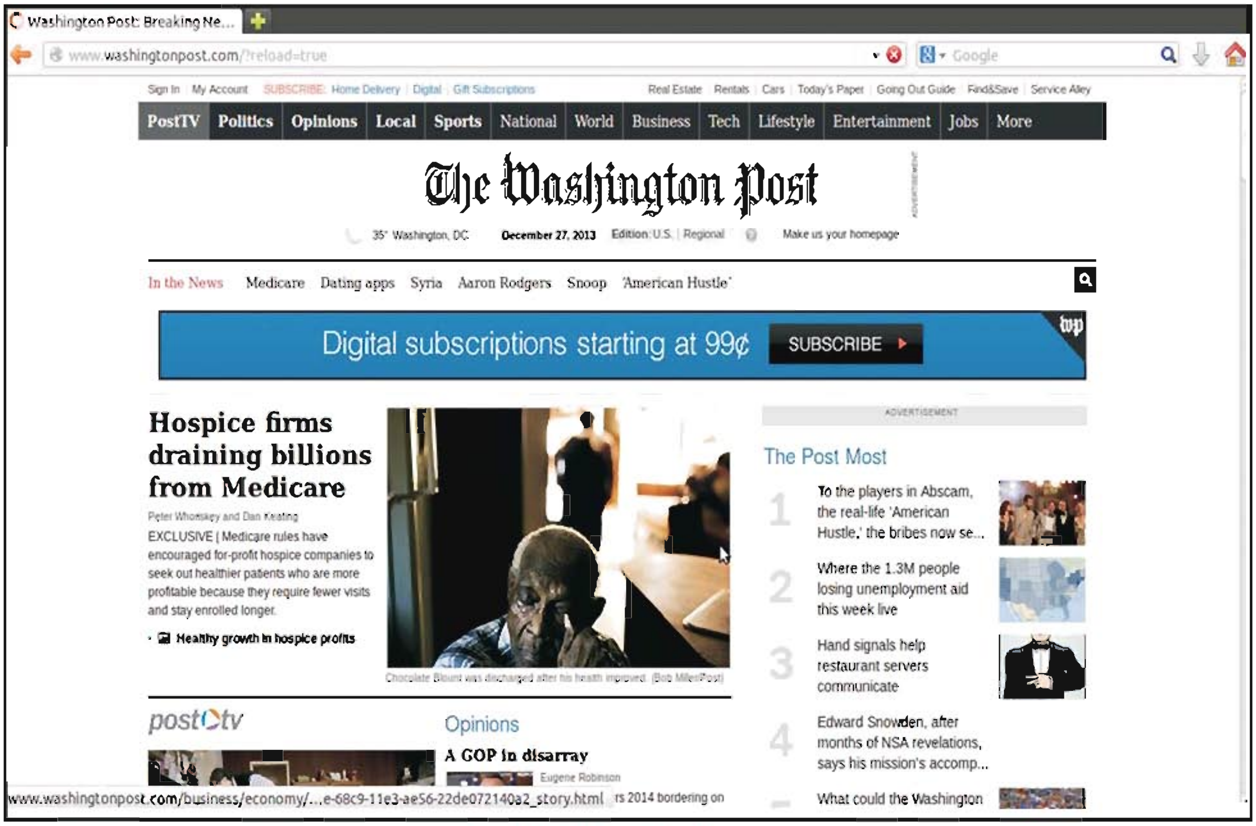
આકૃતિ 4.1 : પુસ્તકની ઑનલાઇન દુકાન

પુસ્તકોની વિશાળ ઑનલાઇન દુકાન ધરાવતી કેટલીક વેબસાઇટ નીચે દર્શાવી છે :

- www.amazon.com
- shopping.indiatimes.com
- www.buybooksindia.com
- www.bookshopofindia.com

ઇલેક્ટ્રૉનિક વર્તમાનપત્ર (Electronic newspaper)

ઇન્ટરનેટ પર ડિજિટલ સ્વરૂપે ઉપલબ્ધ વર્તમાનપત્રને ઇલેક્ટ્રૉનિક વર્તમાનપત્ર કે ઇ-ન્યૂઝપેપર (E-newspaper) તરીકે ઓળખવામાં આવે છે. મુદ્રિત વર્તમાનપત્ર કે ટેલિવિઝન અને રેડિયો દ્વારા પ્રસારિત કરવામાં આવતા સમાચારો કરતાં તે વધુ લાભદાયી છે. વિશ્વસ્તરે થતી ઘટનાઓના તત્કાલ સમાચાર તે આપી શકે છે. ડિજિટલ તકનિકની સહાયથી દર્શકની અભિરુચિને અનુરૂપ બ્રાઉઝરની સંરચના દ્વારા સમાચારોને પ્રદર્શિત કરી શકાય છે. અને મુદ્રણ-પ્રક્રિયામાંથી મુક્તિ મેળવી શકાય છે, જે કિંમત ઘટાડવામાં મદદરૂપ બને છે. મોટા ભાગનાં પ્રતિષ્ઠિત વર્તમાનપત્રો હવે વાચકોને ઇ-વર્તમાનપત્ર પૂરાં પાડે છે. આકૃતિ 4.2માં વૉશિંગ્ટન-પોસ્ટના ઇ-વર્તમાનપત્રનું હોમપેજ દર્શાવ્યું છે.



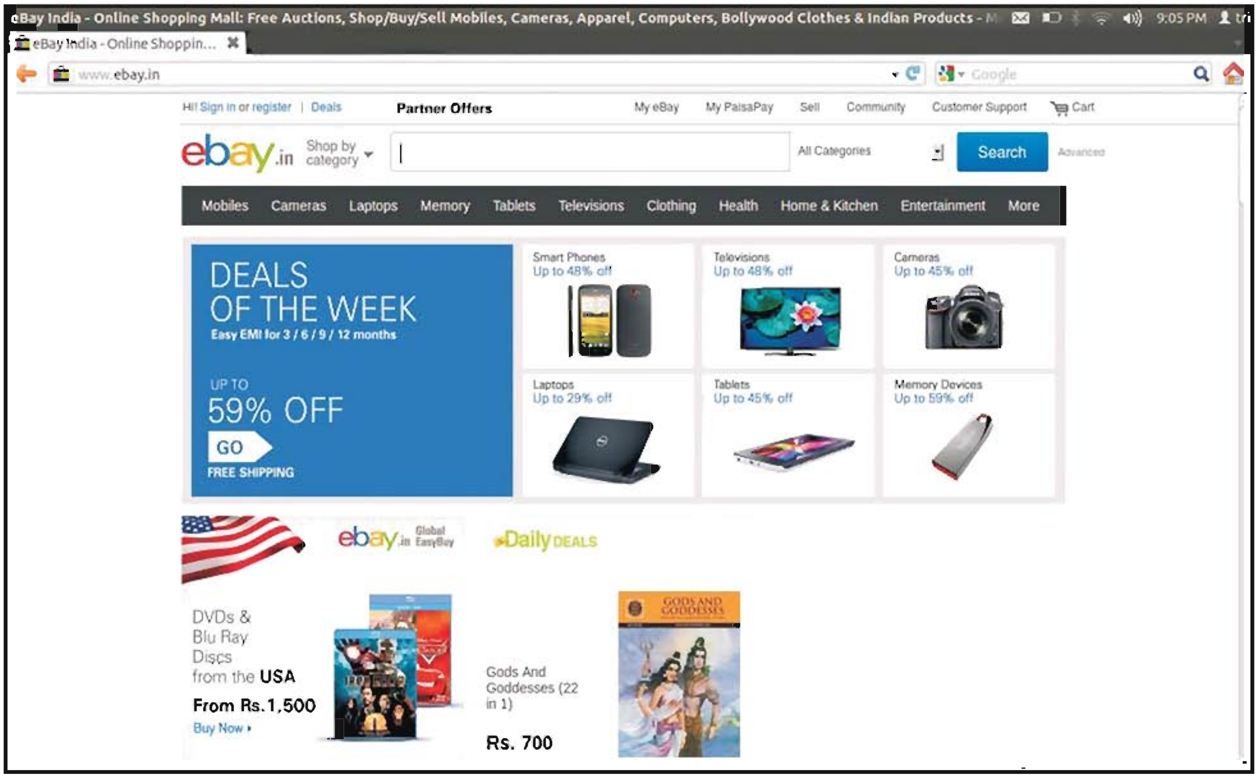
આકૃતિ 4.2 : ઈ-વર્તમાનપત્ર

ઑનલાઈન હરાજી (Online Auctions)

હરાજી ખરીદ અને વેચાણની એવી પ્રક્રિયા છે જે ગ્રાહકને ઉત્પાદનની બોલી બોલવાની તક આપી, ઉચ્ચ બોલી બોલનારને વસ્તુ ખરીદવાની સુવિધા પ્રદાન કરે છે. પરંપરાગત હરાજીમાં મર્યાદિત લોકો ભાગ લે છે. હવે હરાજીની આ પ્રક્રિયાને ઈ-કૉમર્સ તક્નિકની મદદથી અમલમાં મૂકી શકાય છે, જે લોકોને ઇન્ટરનેટ પર બોલી બોલવાની સુવિધા આપે છે. તેને ઑનલાઈન હરાજી (Online Auction) તરીકે ઓળખવામાં આવે છે. માલની જીવંત હરાજી પૂરી પાડતી ઘણી વેબસાઈટ ઉપલબ્ધ છે. આ વેબસાઈટ વેચનાર અને બોલી લગાવનાર બંનેને આધાર પૂરો પાડે છે. જ્યારે તમે આ પ્રકારની સાઈટ પર વસ્તુ મૂકો છો, ત્યારે તમે 'વેચનાર' છો. આ જ સમયે તમે અન્ય વેચનાર દ્વારા સાઈટ પર મૂકેલા ઉત્પાદન માટે બોલી લગાવી શકો છો, તો હવે તમે 'બોલી લગાવનાર' છો.

હરાજીની ઑનલાઈન સાઈટ પર વસ્તુ વેચવા માટે પ્રથમ તમારે તે સાઈટ પર નોંધણી કરાવવી જરૂરી છે. તમે વેચેલી કે બોલી લગાવેલી વસ્તુનું સંધાન (track) મેળવવા, સ્વીકૃત બોલીને શોધવા અને વેચનાર તથા બોલી લગાવનારના પ્રતિસાદનો ડેટાબેઝ બનાવવા નોંધણી જરૂરી બને છે. વેચાણ પહેલાં સભ્યોએ પોતાની મૂળભૂત સંપર્ક-માહિતી પણ પૂરી પાડવી જરૂરી છે. ત્યાર પછી, તેઓ હરાજી માટે મૂકવા માગતા હોય તે વસ્તુ માટે સાઈટ પર આપેલી સૂચનાઓને ક્રમાનુસાર અનુસરે તે જરૂરી છે. સામાન્ય રીતે વિક્રેતા વસ્તુના ડિજિટલ ચિત્ર સહિતની ટૂંકી માહિતી પ્રસ્તુત કરે છે.

હરાજી માટે વેબસાઈટ પર માલ પ્રદર્શિત કરવાથી વ્યક્તિ તેની સારી કિંમત ઉપજાવી શકે છે અને અપેક્ષિત કિંમત સુધી હરાજી પહોંચી જાય કે સમયમર્યાદા પૂરી થાય, ત્યારે વસ્તુ મહત્તમ બોલી લગાવનારને મોકલવામાં આવે છે. બોલી બોલનારને ચુકવણીના વિવિધ વિકલ્પ પણ પૂરા પાડવામાં આવે છે. વિક્રેતા તેના ઉત્પાદન માટે શ્રેષ્ઠ કિંમત મેળવી શકે છે તથા બોલી બોલનાર પોતાની પસંદગીનું ઉત્પાદન ઓછા સમયમાં મેળવી શકે છે. આકૃતિ 4.3માં ઑનલાઈન હરાજીની સાઈટ www.ebay.comનું હોમપેજ દર્શાવ્યું છે.



આકૃતિ 4.3 : ઑનલાઈન હરાજી માટેની સાઈટ

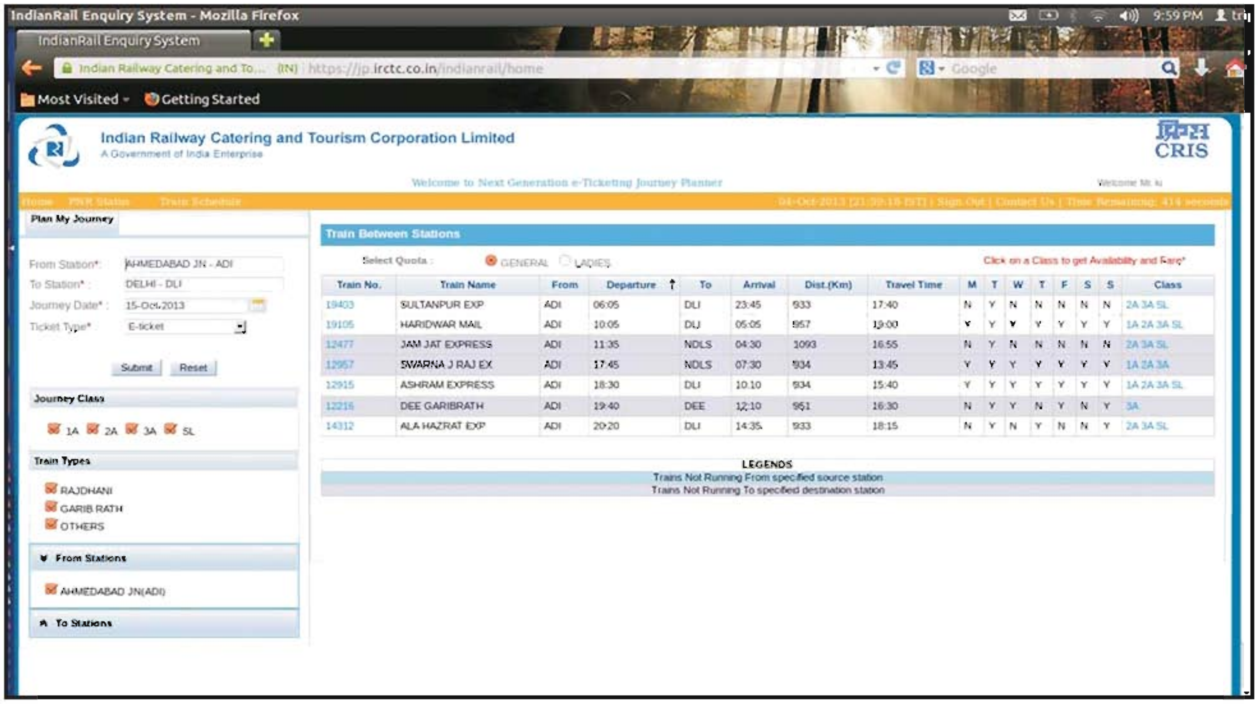
ઑનલાઇન હરાજી માટેની કેટલીક પ્રચલિત વેબસાઇટની યાદી નીચે આપવામાં આવી છે :

- www.ebay.com
- www.onlineauction.com
- www.mybids.in
- www.ubid.com

માર્કેટિંગ અને વેચાણ (Marketing and Selling)

હાલમાં ઘણી સંસ્થાઓ તેમના ઉત્પાદનના પ્રચાર અને સેવાના વ્યવસાયનું સંચાલન વેબસાઇટ દ્વારા કરે છે. વધુ સારા માર્કેટિંગ માટે તેઓ તેમના ઉત્પાદનની સૂચિ (catalogue) ઇન્ટરનેટ પર ઑનલાઇન પૂરી પાડે છે. સૂચિ જુદી જુદી શ્રેણીઓમાં સચિત્ર તો ક્યારેક વીડિયો દ્વારા ઉત્પાદનનું સંક્ષિપ્ત વિવરણ અને વિશેષતાઓ દર્શાવે છે. ગ્રાહક સૂચિને જોઈ શકે છે તથા પોતાની પસંદગીના ઉત્પાદનને શૉપિંગકાર્ટ (shopping cart)માં ઉમેરે છે. ઑનલાઇન શૉપિંગકાર્ટ એ દુકાનમાં આપવામાં આવેલ મૂળભૂત શૉપિંગકાર્ટ જેવું જ હોય છે. સ્થાનિક દુકાનમાં ગ્રાહક ઉત્પાદનને પસંદ કરી શૉપિંગકાર્ટમાં મૂકતો હોય છે. આ પ્રક્રિયા પૂર્ણ થયા બાદ ગ્રાહક દ્વારા ખરીદવામાં આવેલ તમામ ઉત્પાદનોનું શૉપિંગકાર્ટમાંથી બિલ બનાવવામાં આવે છે. આવી જ રીતે, ગ્રાહક ઑનલાઇન શૉપિંગકાર્ટમાં ઉત્પાદન ઉમેરી શકે છે, પસંદ કરેલ ઉત્પાદનને જોઈ શકે છે, તેમાં જરૂરી ફેરફારો કરી શકે છે કે ઉમેરી શકે છે અને અંતમાં ચેકઆઉટ (checkout) સમયે ઉત્પાદનનો ઑર્ડર આપી શકે છે. ઉપયોગકર્તાએ વસ્તુ મેળવવા માટે પોતાના સ્થળની વિગતો (shipping detail) આપવી જરૂરી છે. ખરીદીના બિલની ચુકવણી પણ ઇન્ટરનેટ દ્વારા કરી શકાય છે.

આજે ઇન્ટરનેટ પરથી કરિયાણું, રમકડાં, કમ્પ્યૂટરનાં સાધનો, મોબાઇલ અને અન્ય ઘણાં ઉત્પાદનો ખરીદી શકાય છે. કેટલીક વેબસાઇટ હવાઈ ટિકિટ સાથે પર્યટન-યોજનાઓ પૂરી પાડે છે. હવાઈ કે રેલવે-ટિકિટ હવે સરળતાથી ઑનલાઇન ખરીદી શકાય છે. ઉદાહરણ તરીકે, ભારતીય રેલવેની વેબસાઇટ www.irctc.co.in વિવિધ ટ્રેન અંગેની માહિતી પૂરી પાડે છે તથા ટિકિટની ઑનલાઇન નોંધણી અને ચુકવણી પણ થઈ શકે છે. ટિકિટની નોંધણી થઈ ગયા બાદ તેની ઇ-કૉપી (E-copy) ગ્રાહકના ઇ-મેઇલ પર તથા તેના મોબાઇલમાં SMS દ્વારા મોકલવામાં આવે છે. રેલવે-ટિકિટની ઑનલાઇન નોંધણી આકૃતિ 4.4માં દર્શાવી છે.



આકૃતિ 4.4 : રેલવે-ટિકિટની ઑનલાઇન નોંધણીની સુવિધા

માર્કેટિંગ અને વેચાણ માટેની કેટલીક અન્ય પ્રચલિત સાઇટની યાદી આ મુજબ છે :

- www.homeshop18.com
- www.flipkart.com
- www.myntra.com
- www.makemytrip.com

ઑનલાઇન બિલિંગ (Online Billing)

ઑનલાઇન બિલિંગમાં સંસ્થાઓ પોતાના બિલ ઇ-મેઇલ દ્વારા મોકલે છે. બિલ મેળવ્યા બાદ ગ્રાહક સંસ્થાની વેબસાઇટનો ઉપયોગ કરી ક્રેડિટકાર્ડ કે નેટબૅંકિંગની સુવિધા દ્વારા ઑનલાઇન ચુકવણી કરી શકે છે. જે સંસ્થાઓ બહોળી સંખ્યાના તેમના ગ્રાહકોને સમયાંતરે બિલ મોકલતી હોય, તે આ સુવિધાનો ઉપયોગ કરી શકે છે. ઉદાહરણ તરીકે, BSNL તેના ગ્રાહકોને ઑનલાઇન બિલ મોકલે છે અને તેના ગ્રાહકો ઇન્ટરનેટનો ઉપયોગ કરી ઑનલાઇન ચુકવણી પણ કરી શકે છે.

માહિતીસેવા (Information Services)

ઘણી સંસ્થાઓ પોતાના કર્મચારીઓ કે સભ્યોને અદ્યતન માહિતી પૂરી પાડવા માટે ઇન્ટરનેટનો ઉપયોગ કરે છે. તેમાં શૈક્ષણિક સંસ્થાઓ અને યુનિવર્સિટીઓનો સમાવેશ થાય છે, જે પરીક્ષાનાં પરિણામ, પ્રવેશફૉર્મ, પરીક્ષાનો કાર્યક્રમ, બેઠક-વ્યવસ્થા અને અગત્યની સૂચનાઓ ઇન્ટરનેટ દ્વારા પૂરી પાડે છે. વિદ્યાર્થીઓ કોઈ પણ સ્થળેથી પોતાનું પરિણામ જાણી શકે છે. સંસ્થાઓ કે બૅન્કો દ્વારા ગ્રાહકોને મોકલવામાં આવતી સૂચનાઓ અને સ્મરણપત્રો (reminders) માહિતીસેવાનાં અન્ય ઉદાહરણ છે.

અનેક સંસ્થાઓ વેબસાઇટ પર વિવિધ ફૉર્મ પૂરાં પાડે છે, જેને ગ્રાહક સરળતાથી ડાઉનલોડ કરી ઉપયોગમાં લઈ શકે છે. ઑનલાઇન ફૉર્મ ભરાવીને સંસ્થાઓ પોતાના ગ્રાહકોની માહિતી કે વિગતો પણ મેળવી શકે છે.

સહાયસેવાઓ (Support Services)

છેલ્લા દાયકામાં થયેલા વિશાળ તક્નિકી ફેરફારોને કારણે સહાયસેવાઓનું મહત્ત્વ વધવા પામ્યું છે. આજે સરળતમ ઇલેક્ટ્રૉનિક ઉત્પાદનમાં પણ જ્યારે ક્રાંતિ ઉદ્ભવે, ત્યારે તેના નિવારણ અને જાળવણી માટે વિશિષ્ટ જ્ઞાન અને તક્નિકી ક્ષમતાની

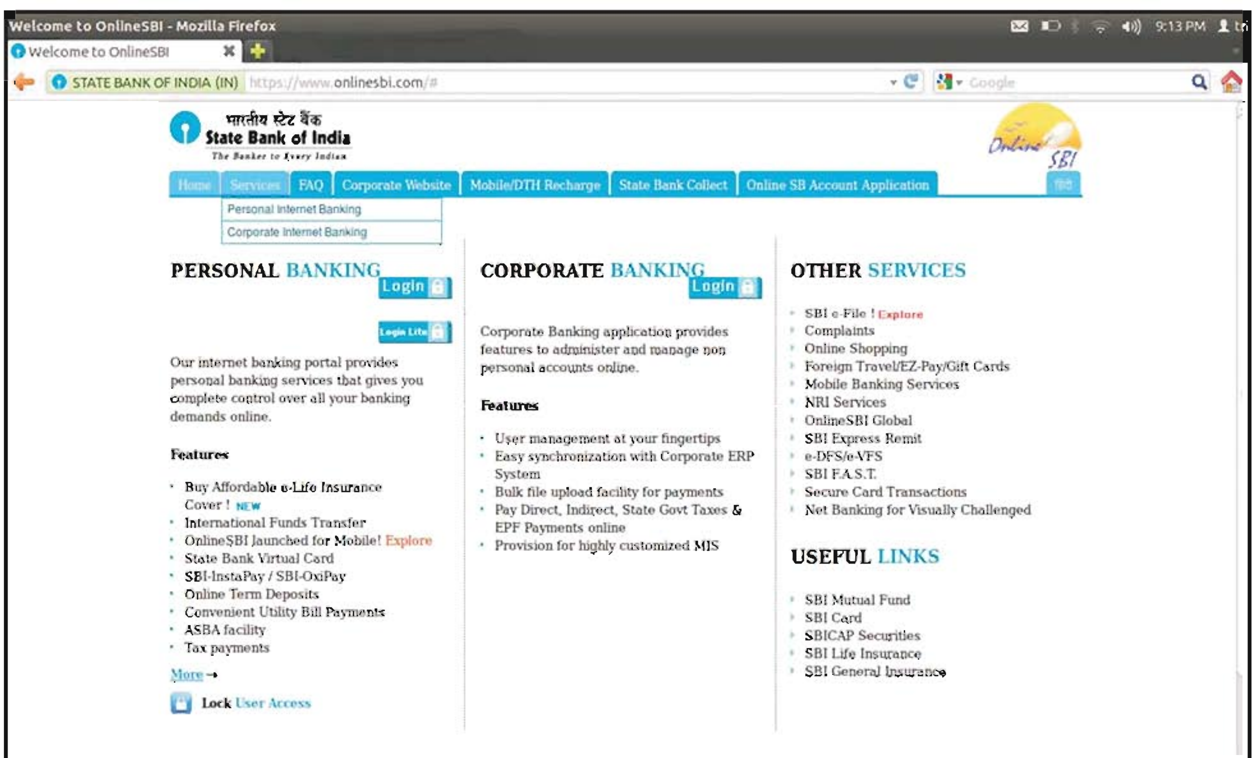
જરૂર પડે છે. ઉત્પાદનના વેચાણ પછી સંસ્થા ગ્રાહકોને ઑનલાઇન સહાય પૂરી પાડે છે. ઉદાહરણ તરીકે, ઇલેક્ટ્રૉનિક ઉત્પાદનનું વેચાણ કરતી સંસ્થા તેના ગ્રાહકોને ફરિયાદની ઑનલાઇન નોંધણી કરવાની સુવિધા પૂરી પાડે છે, જે ત્યાર પછી સહાયક ઇજનેરને મોકલવામાં આવે છે. ગ્રાહકો ઑનલાઇન મૂકેલી ફરિયાદની સ્થિતિ પણ જાણી શકે છે. સૉફ્ટવેર સંસ્થાઓ સૉફ્ટવેરનાં સ્થાપન, સંરચના કે ઉપયોગ સંબંધી કોઈ પણ મુશ્કેલી સંદર્ભે ઑનલાઇન સહાય પૂરી પાડે છે. સૉફ્ટવેર-વિક્રેતા પોતાના લાઇસન્સ-ધારક ગ્રાહકને સૉફ્ટવેરના અદ્યતન સુધારા (updates) ડાઉનલોડ કરવાની સુવિધા પૂરી પાડે છે. હાર્ડવેર-વિક્રેતા તેમનાં સાધનો માટે સૉફ્ટવેર-ડ્રાઇવર પૂરાં પાડે છે. આ ડ્રાઇવરને ગ્રાહક તેના ઉત્પાદનના પ્રકાર અને મૉડેલ પસંદ કરી ડાઉનલોડ કરી શકે છે.

નેટબૅંકિંગ (Net Banking)

આધુનિક યુગમાં નેટબૅંકિંગ કે ઇલેક્ટ્રૉનિક બૅંકિંગ ઘણું પ્રચલિત થઈ રહ્યું છે. ક્યારેક એવું પણ બને કે કોઈક કારણસર ગ્રાહક બૅન્કમાં ગયા વિના તાત્કાલિક ચુકવણી કરવા ઇચ્છતો હોય કે તેના ખાતાની સિલક જાણવા ઇચ્છતો હોય. આના ઉકેલ માટે ઑનલાઇન બૅંકિંગ મદદરૂપ બની શકે છે. ઇન્ટરનેટ દ્વારા બૅન્કના વ્યવહારોનો અમલ કરવાની પ્રક્રિયાને ઑનલાઇન બૅંકિંગ કહે છે. આજે મોટા ભાગની પ્રતિષ્ઠિત બૅન્કોએ પોતાના ગ્રાહકોને ઑનલાઇન બૅંકિંગની સુવિધા પૂરી પાડવાનું શરૂ કર્યું છે. ઑનલાઇન બૅંકિંગની મદદથી ગ્રાહકોને નીચે જણાવેલ સેવાઓ પૂરી પાડવામાં આવે છે :

- કોઈ પણ સમયે ખાતાની સિલકની જાણકારી
- એક ખાતામાંથી અન્ય ખાતામાં રકમની લેવડદેવડ
- કોઈ પણ આવક કે જાવક માટેનાં પત્રક (statement)
- નાણાકીય વ્યવહારની સ્થિતિ વિશે જાણકારી
- બૅન્કમાં ગયા વગર ટેલિફોન, ઇલેક્ટ્રિસિટી અને અન્ય બિલની ચુકવણી

ગ્રાહકને ઑનલાઇન સેવાઓ માટે પાસવર્ડ આપવામાં આવે છે, જેના દ્વારા કમ્પ્યૂટર કે મોબાઇલની મદદથી તે બૅન્કની સાઇટ પર લૉગ-ઇન કરી બૅન્ક સાથેની તમામ પ્રવૃત્તિઓ કરી શકે છે. આકૃતિ 4.5માં સ્ટેટ બૅન્ક ઑફ ઇન્ડિયાની વેબસાઇટનું હોમપેજ દર્શાવ્યું છે. આ માટે www.onlinesbi.com એ URLનો ઉપયોગ કરવામાં આવે છે.



આકૃતિ 4.5 : ઑનલાઇન બૅંકિંગની સુવિધા

આજે મોટા ભાગની બૅન્કો ઑનલાઇન બૅંકિંગ પૂરું પાડે છે; તેમાંની કેટલીક બૅન્કોની વેબસાઇટ નીચે આપેલ છે :

- www.centralbankofindia.co.in
- www.bankofbaroda.co.in
- www.iob.in
- www.pnbindia.in
- www.denabank.co.in

પરંપરાગત વ્યવસાય વિરુદ્ધ ઈ-કૉમર્સ (Traditional Commerce Vs. E-commerce)

ઇન્ટરનેટના વિકાસને કારણે ઇ-કૉમર્સનો ઉપયોગ વધવાથી પરંપરાગત વ્યવસાયોની સ્પર્ધામાં નોંધપાત્ર પરિવર્તન આવ્યું છે. પરંપરાગત વ્યવસાયમાં સંસ્થાએ કોઈ એક જ ઉદ્યોગ સાથે અને તે પણ સીમિત ભૌગોલિક વિસ્તારમાં જ સ્પર્ધા કરવાની રહે છે. ઘણા કિસ્સાઓમાં વ્યાવસાયિક પ્રક્રિયાઓમાં પરંપરાગત પ્રવૃત્તિઓનો ઉપયોગ ખૂબ જ અસરકારક હોય છે અને તેને ટેક્નોલૉજી દ્વારા વધુ સારો બનાવી શકાતો નથી. જે ઉત્પાદનોને ગ્રાહક સ્પર્શ કરીને, સૂંઘીને કે તપાસીને લેવા ઇચ્છતો હોય, તેનું ઇ-કૉમર્સની મદદથી વેચાણ કરવું મુશ્કેલ છે. ઉદાહરણ તરીકે, ખરીદદારો નાશવંત ખાદ્યપદાર્થો, કિંમતી ઘરેણાં અને ઉચ્ચ બનાવટનાં કપડાં ઑનલાઇન ખરીદતા અચકાશે, કારણકે આ વસ્તુઓને ખરીદતાં પહેલાં બરાબર તપાસી શકાશે નહીં. પરંપરાગત વ્યવસાયમાં છૂટક વ્યાપારીઓ તેમના વર્ષોના અનુભવ પરથી દુકાનની એવી રૂપરેખા અમલમાં મૂકે છે, જે ગ્રાહકને પોતાની વસ્તુ ખરીદવા માટે અનુકૂળતા પૂરી પાડે. ઉત્પાદનોની ગોઠવણ, દુકાનની રૂપરેખા અને ડિઝાઇનને જ વ્યાપાર કહે છે. વ્યાપારીઓમાં ગ્રાહકની જરૂરિયાતોને પારખી શકવાનું કૌશલ્ય હોય છે અને તેઓ જરૂરિયાત અનુસાર ઉત્પાદનો અને સેવાઓ શોધી કાઢી શકે છે. પરંતુ વ્યાપારની આ કળા ઇન્ટરનેટ પર અમલમાં મૂકવી મુશ્કેલ છે. પરંપરાગત વ્યાપારની થોડી સામાન્ય લાક્ષણિકતાઓ નીચે મુજબ છે :

- નિશ્ચિત સમયગાળા દરમિયાન અથવા વ્યવસાયના કલાકો દરમિયાન વ્યાપાર કરવામાં આવે છે.
- હરીફો સમક્ષ માહિતી વહેંચવી પડતી નથી.
- વેચાણ અધિકારી, વેચાણ વહીવટકાર અને તેવી અન્ય વ્યક્તિઓની નિમણૂંક.
- જગ્યાનું ભાડું, માલની ખરીદી, જાહેરાત, માલ-યાદી (inventory), માલ મોકલવાની વ્યવસ્થા વગેરે.

જોકે, આધુનિક યુગની ગતિશીલતાને જોતાં આ પરંપરાગત નિયમો તોડવા ઘણા જરૂરી બને છે. ઇન્ટરનેટનો ઉપયોગ કરી દુનિયાભરમાં વ્યવસાય કરી શકાય તે માટેના નવા માર્ગ અપનાવવા જરૂરી છે. ઇ-કૉમર્સની કેટલીક લાક્ષણિકતાઓ નીચે દર્શાવી છે :

- ઉત્પાદનની જાહેરાત ઇલેક્ટ્રૉનિક સંસાધનો દ્વારા થાય છે.
- ઉત્પાદનની સૂચિ અને ઉપલબ્ધ યોજનાઓ વિશે ગ્રાહકને જાણકારી આપી શકાય છે.
- ચુકવણીની ઇ-પેમેન્ટ પદ્ધતિઓનો ઉપયોગ કરી શકાય છે. તદુપરાંત માલ મેળવતી વખતે કરાતી ચુકવણીની પદ્ધતિનો વિકલ્પ પણ ઉપલબ્ધ હોય છે.
- ગ્રાહકને થોડા દિવસમાં માલ પહોંચાડવામાં આવે છે.
- વ્યવહારનો સરેરાશ ખર્ચ ઘટે છે.
- સંપૂર્ણ વ્યવહાર પૂરો કરવા માટેનો સમય ઘટે છે.

ભારતમાં ઈ-કૉમર્સ (E-commerce in India)

ભારતમાં ઇન્ટરનેટના વપરાશકાર ઉત્તરોત્તર વધતા રહ્યા છે. ઇ-કૉમર્સની વૃદ્ધિ પણ નોંધપાત્ર રીતે અનુભવાઈ રહી છે અને લોકોની વ્યવહાર કરવાની પદ્ધતિ સફળતાપૂર્વક બદલાઈ રહી છે. ભારતીય ઑનલાઇન ખરીદદારો માટે ઉચ્ચકક્ષાનાં સાધનોથી લઈને જૂનાં (second hand) પુસ્તકો સુધીની તમામ વસ્તુઓ હવે એક બટન ક્લિક કરવાથી મેળવી શકાય તે રીતે ઉપલબ્ધ છે. લાખો લોકો માટે તે નવી ક્ષિતિજો અને તક ખોલી આપે છે.

ઉત્પાદન અને સેવાની વિવિધતા ઇન્ટરનેટ પરની ઑનલાઇન ખરીદીને આકર્ષક અને અનુકૂળ બનાવે છે. ઇન્ટરનેટ ભારતના ગ્રામ્ય વિસ્તારમાં પણ પહોંચી ચૂક્યું હોવાથી ત્યાંના લોકોની જીવનપદ્ધતિ, વપરાશની ગુણવત્તા અને જથ્થો તથા વિચારધારામાં આમૂલ પરિવર્તન આવ્યું છે.

ભારતમાં ઇ-કૉમર્સની વૃદ્ધિમાં ઘણાં પરિબળો ભાગ ભજવે છે. તેમાંનાં કેટલાંક મુખ્ય પરિબળો નીચે આપ્યાં છે :

- ઇન્ટરનેટનું જોડાણ, બ્રૉડબેન્ડ અને ત્રીજી પેઢી (third generation - 3G)ની સેવાઓ, લેપટૉપ, સ્માર્ટફોન, ટેબ્લેટ અને ડોંગલ જેવી તક્નિકી સરળતામાં થયેલી વૃદ્ધિ.
- મોબાઇલ સાધનોના ઉપયોગમાં થયેલો વધારો.
- વધુ વ્યાપક ઉત્પાદનોના વિસ્તારની ઉપલબ્ધતા.
- પરંપરાગત ખરીદી માટે વ્યસ્ત જીવન, વાહન-વ્યવહારની ગીચતા અને સમયનો અભાવ.
- મધ્યસ્થીઓ દ્વારા ચલાવવામાં આવતા છૂટક વેચાણમાં થતા માલસંગ્રહ અને સ્થાયી મિલકતમાં ઘટાડો થવાથી પરંપરાગત પદ્ધતિ કરતા ઓછી કિંમતે વેચાણ.
- ઑનલાઇન વર્ગીકૃત સાઇટના ઉપયોગમાં થયેલા વધારાને કારણે વધુ ને વધુ ગ્રાહકો દ્વારા જૂના માલનું ખરીદ-વેચાણ.
- eBay, Flipkart, Snapdealને તેના જેવી અનેક સાઇટ સાથે ઑનલાઇન માર્કેટ-જગ્યાનો થયેલો વિકાસ.

અન્ય દેશોની તુલનામાં ભારતમાં ઇ-કૉમર્સનો પ્રવેશ અપેક્ષાકૃત મોડો થયો હોવા છતાં, તેમાં નવા પ્રવેશોની વિપુલ સંખ્યાને કારણે હવે તેનો અભૂતપૂર્વ ગતિથી વિકાસ થઈ રહ્યો છે. Flipkart, eBay India, Snapdeal, Amazon India, Myntra, Domino, PayTM, Jabong અને તેના જેવી અનેક ઑનલાઇન દુકાનો ભારતમાં પ્રચલિત બની રહી છે.

છૂટક વેપારીઓએ માલ ખરીદતી વખતે તત્કાલ ચુકવણીની સુવિધા (cash-on-delivery) પણ શરૂ કરી છે, તે ભારતમાં ચુકવણી માટેની સૌથી વધુ લોકપ્રિય પદ્ધતિ છે. લગભગ 80 % ભારતીય ઇ-કૉમર્સ વ્યવસાય માલ મેળવતી વખતે તત્કાલ ચુકવણીની સુવિધાનો ઉપયોગ કરે છે. જોકે, ઑનલાઇન છૂટક વેપારીની દૃષ્ટિએ જોતાં આ એક ખર્ચાળ દરખાસ્ત છે, કારણકે જ્યાં સુધી માલ મોકલવામાં ન આવે, ત્યાં સુધી તેમણે વેચાણ માટેનાં નાણાંની જોગવાઈ કરી રાખવી પડે છે. ગ્રાહક માટે તત્કાલ ચુકવણીની સુવિધાથી માલનો અસ્વીકાર કરવાનું સરળ બને છે.

ઉદ્યોગ-નિષ્ણાતો ભારતમાં ઇ-કૉમર્સ ઉદ્યોગના સંભવિત ભવિષ્ય માટે ઘણા આશાવાદી છે. તેમાં ઘણા પડકારોનો સામનો કરવાનો છે, પરંતુ સુનિયોજિત અને પૂર્વનિશ્ચિત યોજનાઓ દ્વારા ભારતમાં ઇ-કૉમર્સ સરળતાપૂર્વક વિકાસ સાધશે, તેમ માનવામાં આવે છે.

ઇ-કૉમર્સના ફાયદા (Advantages of E-commerce)

ઇન્ટરનેટના વિકાસને લીધે સંસ્થાઓ પોતાના વ્યવસાયના પ્રસાર માટે અનેક નવીન અને આકર્ષક યોજનાઓ શોધી રહી છે. આજે એવી સફળ સંસ્થા શોધવી મુશ્કેલ છે, જે પોતાની પ્રવૃત્તિઓ માટે કમ્પ્યૂટરનો ઉપયોગ કરતી ન હોય. ઇ-કૉમર્સ ગ્રાહકોને સસ્તા ભાવે માલની ઉપલબ્ધતા, વિશાળ પસંદગી અને સમયના બચાવ જેવા અનેક લાભ પૂરા પાડે છે. વધુ ઝડપી બ્રૉડબૅન્ડ સેવાઓ અને નવા વિનિયોગોની ઉપલબ્ધતાથી આવનારા વર્ષોમાં ઇ-કૉમર્સના વેચાણમાં વધારો થશે. ઇ-કૉમર્સના કેટલાક ફાયદા નીચે જણાવ્યા છે :

- **અવિરત સમય (24×7) માટેની વ્યાપારવ્યવસ્થા (Conduct Business 24×7)**

  ઇ-કૉમર્સની મદદથી કોઈ પણ સમયે અને સ્થળે વ્યવસાય કરી શકાય છે. વ્યાવસાયિક પ્રવૃત્તિઓને સમયનું કોઈ બંધન રહેતું નથી. દિવસમાં કોઈ પણ સમયે ગ્રાહક ઉત્પાદન ખરીદી શકે છે અને કોઈ પણ સમયે ઑર્ડર પણ સ્વીકારવામાં આવે છે.

- **ઓછો ખર્ચ (Lower Cost)**

  ઈન્ટરનેટના વ્યાવસાયિક ઉપયોગથી માર્કેટિંગ, વહેંચણી, ફોન, ટપાલ, છાપકામ અને તેના જેવી અનેક પ્રવૃત્તિઓમાં થતો ખર્ચ ઘટે છે. સ્થાનિક વ્યવસાયમાં કરાતા મૂડીરોકાણની તુલનામાં ઇ-કૉમર્સનું મૂડીરોકાણ અલ્પતમ હોય છે. ઈન્ટરનેટ દ્વારા કરવામાં આવતા વ્યવસાય કરતાં સ્થાનિક વ્યવસાય/દુકાન શરૂ કરવાનો ખર્ચ ઘણો વધુ આવે છે. ઉત્પાદનનાં વેચાણ માટે વિતરકો કે અન્ય મધ્યસ્થીઓની ગેરહાજરીથી ગ્રાહકોને ઓછા ખર્ચે ઉત્પાદનની વિશાળ શ્રેણી મેળવવાનો લાભ મળે છે.

- **સરહદ કે ભૌગોલિક મર્યાદાનો અભાવ (No Boundaries or Geographical Limitations)**

  સ્થાનિક દુકાનની સીમા નિશ્ચિત ક્ષેત્ર સુધી વિસ્તરેલી હોય છે. ઇ-કૉમર્સની મદદથી વ્યવસાયને તત્કાલ અસંખ્ય ગ્રાહકો સુધી પહોંચાડી શકાય છે. પરંપરાગત માર્કેટિંગની કોઈ પણ યોજના દ્વારા આ શક્ય નથી. નવા બજારની રચના સહિત સંભવિત ખરીદદારો સુધી સરળતાથી પહોંચી શકવાની ક્ષમતા તે ધરાવે છે. વ્યવસાયની વેબસાઇટ બનાવી તેને ઈન્ટરનેટ પર અપલોડ કરવાથી વૈશ્વિક વિસ્તાર સુધી પહોંચવું સરળ બને છે.

- **ઉન્નત અને વધુ સારી ગ્રાહકસેવા (Improved and Better Customer Service)**

  ગ્રાહક સાથે પ્રત્યક્ષ સંવાદ શક્ય હોવાને કારણે ઉત્પાદનની કિંમત, જથ્થા, વિશેષતાઓ અને અન્ય પ્રશ્નોનો ઉકેલ સરળતાથી મેળવી શકાય છે, જે વધુ સારી ગ્રાહક સેવા પૂરી પાડે છે. જે સંસ્થાઓ અન્ય સંસ્થાઓ સાથે મળીને વ્યવસાય કરે છે, તેમના માટે ઑનલાઇન ગ્રાહકસેવા ઉમેરવી એ એક સ્પર્ધાત્મક લાભ છે. ઑનલાઇન ગ્રાહકસેવાને ગ્રાહક વધુ પસંદ કરે છે. ઇ-કૉમર્સ વધુ સારી અને ત્વરિત ગ્રાહકસેવા પૂરી પાડે છે. ઈન્ટરનેટ પર માહિતીનો પ્રસાર ઘણી સરળતાથી થઈ શકે છે. વ્યાપારી તેના નવા ઉત્પાદનની જાણ ગ્રાહકને ઇ-મેઇલ દ્વારા કરી શકે છે અને ઉત્પાદન સંબંધી મુશ્કેલીઓનું નિવારણ પણ લાવી શકે છે.

- **જૂથકાર્ય (Teamwork)**

  ઇ-કૉમર્સની મદદથી એકાધિક સંસ્થાઓ સાથે મળી કાર્ય કરી શકે છે. આનું એક ઉદાહરણ ઇ-મેઇલ છે, જેની મદદથી લોકો માહિતીનો વિનિમય કરે છે. તેના દ્વારા વિતરકો, વ્યાપારીઓ, ધંધાકીય ભાગીદારો અને ગ્રાહકો સાથેના સંસ્થાના સંવાદની પદ્ધતિમાં સુધારો થયો છે. વધુ સંવાદ વધુ સારું પરિણામ આપે છે.

- **પરિવહનના સમય અને ખર્ચમાં ઘટાડો (Eliminate Travel Time and Cost)**

  ઇચ્છિત દુકાન સુધી પહોંચવા માટે ગ્રાહકે અહીં લાંબું અંતર કાપવાનું રહેતું નથી. ઇ-કૉમર્સની મદદથી આ દુકાનની મુલાકાત માઉસની કેટલીક ક્લિક કરવાથી લઈ શકાય છે.

- **ઝડપ (Speed)**

  પરંપરાગત પદ્ધતિઓની તુલનામાં ઇ-કૉમર્સ વ્યવસાય ઘણો ઝડપી છે. ઑનલાઇન વ્યવહારોને કારણે સંચારમાં થતો સમયનો વ્યય નાબૂદ થાય છે, જે વ્યવસાયની ઝડપ નોંધપાત્ર રીતે વધારે છે. વ્યાવસાયિક સંસ્થાઓ વિતરકને ખરીદીના ઑર્ડર ઑનલાઇન બનાવીને મોકલી શકે છે, જે સમયનો વ્યય અટકાવે છે. વિતરકો તત્કાલ ઑર્ડર મેળવી તેના પર કાર્ય શરૂ કરી શકે છે, આથી ઑર્ડર મોકલવાનો સમય પણ બચે છે. ઈન્ટરનેટ પર આપવામાં આપેલ માહિતીને અવારનવાર બદલી શકાય છે. આ સુવિધાની મદદથી સંસ્થાનાં ઉત્પાદનો કે સેવામાં થતા ફેરફાર વિશે ગ્રાહકને જાણકારી આપી શકાય છે.

ઇ-કૉમર્સ સમાજને નીચે જણાવેલ લાભ પણ પૂરા પાડે છે :

- ઘર, કાર્યાલય કે કોઈ પણ સ્થળેથી ખરીદી.
- ઉત્પાદનની ખરીદી માટે પરિવહન ઓછું થતું હોવાથી યાતાયાત અને પ્રદૂષણમાં ઘટાડો.
- આરોગ્યલક્ષી સેવાઓ.
- ડિસ્ટન્સલર્નિંગ અને અભ્યાસ.

ઈ-કૉમર્સની મર્યાદાઓ (Limitations of E-commerce)

ઈ-કૉમર્સના ફાયદાઓની યાદી બનાવી શકાય તેમ હોવા છતાં તેમાં કેટલીક મર્યાદાઓ અને ગેરફાયદા પણ રહેલાં છે. જોકે, સમયાંતરે ઘણી મર્યાદાઓ લુપ્ત થતી જાય છે. ઈ-કૉમર્સની કેટલીક મર્યાદાઓ નીચે દર્શાવી છે :

- **પરિવર્તનનો પ્રતિકાર (Resistance to Change)**

  વ્યાવસાયિક સંસ્થાઓને પરંપરાગત વ્યવસાયમાંથી ઈ-કૉમર્સ વ્યવસાય તરફ અભિમુખ થવું પડે છે, જેમાં તેમને ઘણો પ્રતિકાર સહન કરવો પડે છે. લોકોને પણ વ્યવસાયના કાગળરહિત અને પ્રત્યક્ષ ન હોય તેવા વ્યવહારોની આદત પાડવી પડે છે.

- **પ્રારંભિક ખર્ચ (Initial Cost)**

  ઈ-કૉમર્સની નવી તક્નિકો માટે નોંધપાત્ર પ્રારંભિક મૂડીરોકાણ જરૂરી છે, જે સંસ્થાની મૂળભૂત વ્યાવસાયિક પદ્ધતિઓ બદલી નાંખે છે. હાર્ડવેર અને સૉફ્ટવેરની તક્નિકોમાં મૂડીરોકાણ જરૂરી બને છે. ઈ-કૉમર્સની મદદથી કરવામાં આવતા વ્યવસાયને સમજવા માટે કર્મચારીઓને તાલીમની પણ જરૂર પડે છે.

- **સુરક્ષા (Security)**

  ઈ-કૉમર્સમાં મૂળભૂત રીતે ધ્યાન પર લેવા જેવી બાબત સુરક્ષાની છે. સંસ્થા કે વ્યક્તિની સંબંધિત માહિતીનું જ્યારે ઈન્ટરનેટ પર પ્રસારણ થતું હોય, ત્યારે અનધિકૃત ઉપયોગથી તેમનું રક્ષણ કરવું જરૂરી બને છે. ઈન્ટરનેટ વૈશ્વિક ઉપયોગ પૂરો પાડે છે. પરંતુ સંસ્થાઓએ દૂષિત (malicious) ઉપયોગ કે અકસ્માતે થતા દુરુપયોગ સામે પોતાની માહિતીને સુરક્ષા પૂરી પાડવી જોઈએ.

- **ગોપનીયતા (Privacy)**

  ગ્રાહકની ગોપનીયતા એ ઘણો ગંભીર મુદ્દો છે. દુરુપયોગ થવાના ભયને કારણે ગ્રાહક પોતાની અંગત માહિતી ઈન્ટરનેટ પર દર્શાવતા અચકાય છે. કેટલીક સંસ્થાઓ અન્ય માર્કેટિંગ સંસ્થાઓને તેમના ડેટાબેઝની માહિતી વેચે છે, જેનો દુરુપયોગ કરી ગ્રાહકોને બિનજરૂરી (spam) મેઈલ મોકલવામાં આવે છે. હેકર પણ ઈન્ટરનેટ પરથી માહિતીને આંતરીને તેનો દુરુપયોગ કરી શકે છે. ક્રેડિટકાર્ડની છેતરપિંડી ગ્રાહકને આર્થિક નુકસાન પહોંચાડે છે. આમ, હેકર, વાઈરસ, વિગતોનું સ્થાનાંતરણ અને વ્યવહારનાં જોખમો સામે ઈ-કૉમર્સને સુરક્ષા પૂરી પાડવી જરૂરી બને છે.

- **વિશ્વાસનો અભાવ (Lack of Trust)**

  કેટલીક વાર, ઉત્પાદન ન પહોંચાડવાની છેતરપિંડી, ઉત્પાદન અંગેની ભ્રામક રજૂઆત, ચુકવણીની સુરક્ષાનો અભાવ વગેરે કારણોસર ગ્રાહકોમાં ઈ-કૉમર્સ પ્રત્યે વિશ્વાસનો અભાવ જોવા મળે છે. તદુપરાંત ઑનલાઈન ખરીદીમાં ક્ષતિગ્રસ્ત માલને પરત કરવાનું પણ મુશ્કેલ છે. પરત કરેલ માલનો વહનખર્ચ કોણ ભોગવશે, વેપારી કિંમત પરત કરશે કે નહીં અને ઉત્પાદન તેના મૂળ સ્રોત સુધી પરત થશે કે નહીં જેવા પ્રશ્નો સામાન્ય રીતે અહીં ઉદ્ભવતા હોય છે.

- **ઉત્પાદનને પહોંચાડવાનો સમય (Time for Delivery of Products)**

  સ્થાનિક દુકાનમાં આપણે ઉત્પાદનની ખરીદી કરી તેને સાથે લઈ જઈ શકીએ છીએ. ઈ-કૉમર્સનો ઉપયોગ કરી આપણે એવાં ઉત્પાદનો ખરીદતાં હોઈએ છીએ કે સ્થાનિક રીતે ઉપલબ્ધ નથી હોતાં. એનો અર્થ એ થાય કે ઉત્પાદનને મોકલવામાં સમય લાગશે અને ખર્ચ થશે. દૂરના સ્થળેથી ઉત્પાદનનો ઑર્ડર આપવામાં આવ્યો હોય, તો તેને મોકલવાનો ખર્ચ વધારે ચૂકવવો પડે છે.

- ફળ, શાકભાજી અને અન્ય નાશવંત ઉત્પાદનો ઑનલાઈન ખરીદવા માટે પસંદ કરવામાં આવતાં નથી. ઑનલાઈન દુકાનો ઉત્પાદનને સ્પર્શવાની, ધારણ કરવાની કે પ્રત્યક્ષ અનુભવવાની સુવિધા આપી શકે નહીં. માટે વસ્ત્રો અને ઘરેલું રાચરચીલાં જેવાં ઉત્પાદનોનું ઑનલાઈન વેચાણ યુક્તિપૂર્વક કરવું પડે છે.

- ઈ-કૉમર્સમાં ચુકવણી માટે મોટે ભાગે ક્રેડિટકાર્ડનો ઉપયોગ કરવામાં આવે છે. ક્રેડિટકાર્ડ દ્વારા ઑનલાઈન તદ્દન નાના કે બહુ મોટા નાણાકીય વ્યવહારો પસંદ કરવામાં આવતા નથી. નાની સંસ્થાઓને પોતાનો ઑનલાઈન વ્યવસાય

સ્થાપિત કરવામાં સમયાંતરે કરવા પડતા સુધારાઓ (updates) મુશ્કેલી ઊભી કરે છે. ઇ-કૉમર્સ ઇન્ટરનેટ પર આધારિત છે અને ઇન્ટરનેટના જોડાણમાં થતો અવરોધ વ્યવસાય માટે જોખમી બની રહે છે.

ઇ-કૉમર્સની વ્યાવસાયિક પ્રતિકૃતિઓ (E-commerce Business Models)

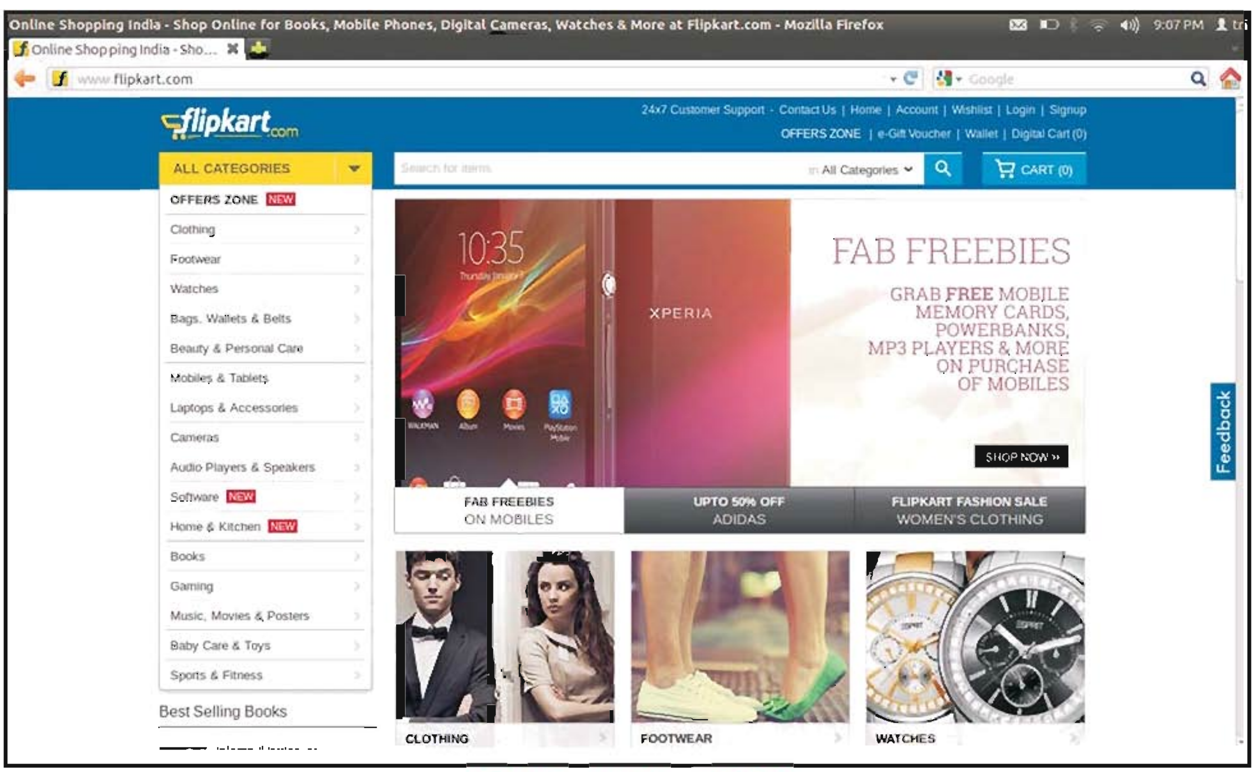
ઇ-કૉમર્સમાં સમાવિષ્ટ પક્ષ અને તેના દ્વારા પૂરી પાડવામાં આવતી પ્રવૃત્તિઓ તથા સેવાઓને આધારે ઇ-કૉમર્સની વ્યાવસાયિક પ્રતિકૃતિઓની વ્યાખ્યા કરવામાં આવે છે. ઇ-કૉમર્સમાં અનેક વ્યાવસાયિક પ્રતિકૃતિઓ સૂચિત કરવામાં આવી છે, પરંતુ નીચે આપવામાં આવેલી પ્રતિકૃતિઓ સૌથી વધુ પ્રચલિત છે. સામાન્ય રીતે આ માટેનું વર્ગીકરણ 'કોણ કોને માલ વેચે છે', તેના પર આધારિત હોય છે.

- વ્યવસાયીથી ગ્રાહક (Business to Consumer - B2C)
- વ્યાવસાયીથી વ્યવસાયી (Business to Business - B2B)
- ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)
- ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

હવે, દરેકની વિસ્તૃત સમજૂતી મેળવીએ.

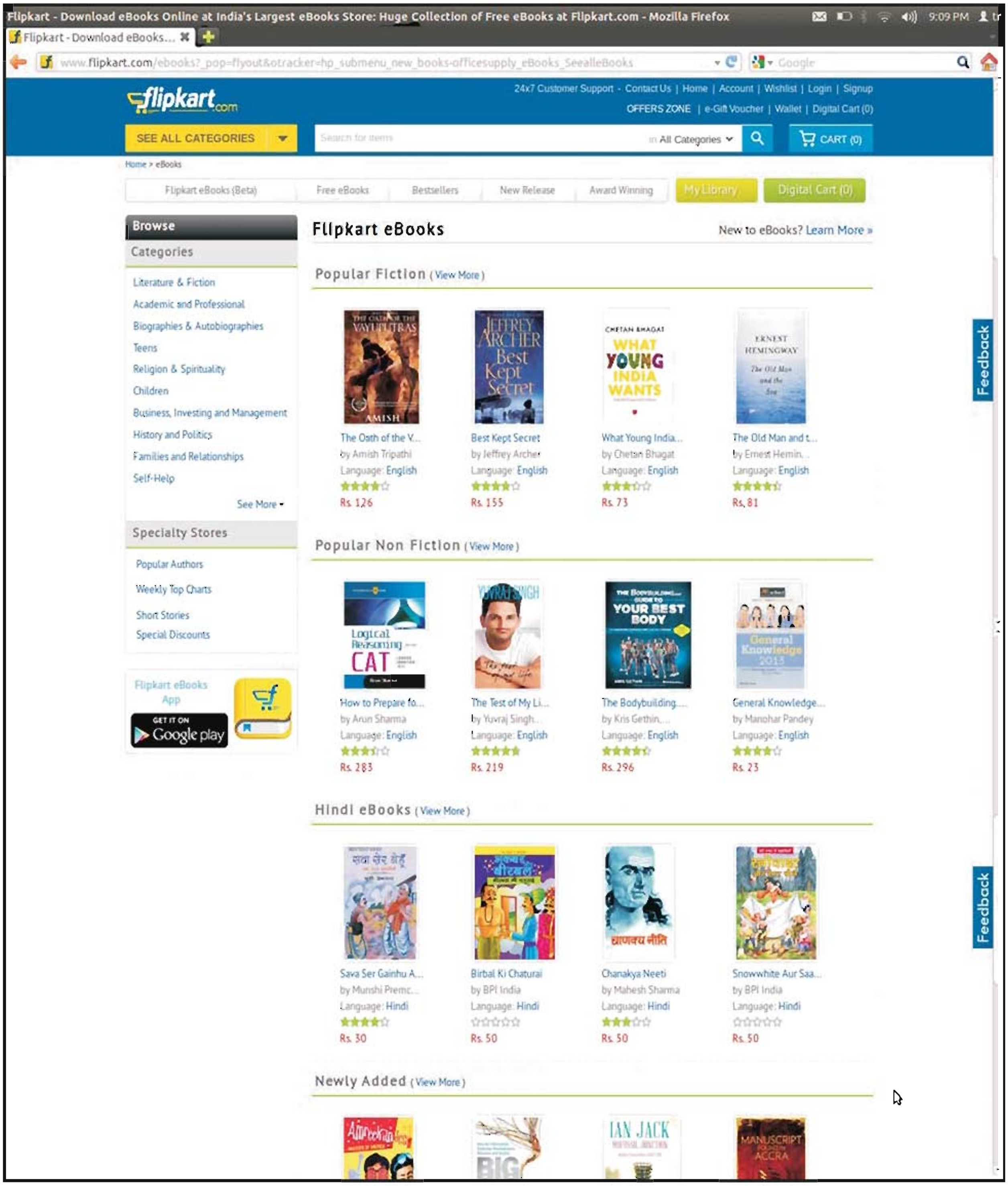
વ્યવસાયીથી ગ્રાહક (Business to Consumer)

ઇન્ટરનેટ દ્વારા વેબસાઇટના ઉપયોગથી જે વ્યાવસાયિકો કે સંસ્થાઓ ગ્રાહકોને તેમનાં ઉત્પાદનો કે સેવાઓનું વેચાણ કરે છે, તેમના માટે 'વ્યવસાયીથી ગ્રાહક' પ્રતિકૃતિનો સંદર્ભ આપી શકાય. ગ્રાહકો કોઈ પણ સ્થળેથી કોઈ પણ સમયે ઉત્પાદન કે સેવાઓ પસંદ કરી તેનો ઑર્ડર આપી શકે છે. વેચનાર પોતાના ઉત્પાદનનું વેચાણ મધ્યસ્થી વિના સીધું જ ખરીદદારને કરી શકે છે, જ્યાં ખરીદદાર એક સ્વતંત્ર ગ્રાહક છે. ઇન્ટરનેટ પર આ પ્રતિકૃતિ સૌથી વધુ જોવા મળે છે. ઉત્પાદન સરળતાથી અને ઝડપથી ખરીદી શકાતાં હોવાને કારણે આજકાલ તે ગ્રાહકોમાં વધુ પ્રચલિત છે. વસ્તુના છૂટક વેચાણ ઉપરાંત B2Cમાં ઑનલાઇન બૅંકિંગ, મકાનની લે-વેચ, પરિવહન-સુવિધાઓ અને તેના જેવી અન્ય અનેક સેવાઓનો સમાવેશ કરવામાં આવ્યો છે. amazon.com, rediff.com, fabmart.com, flipkart.com વગેરે B2C વેબસાઇટનાં કેટલાંક ઉદાહરણ છે. આકૃતિ 4.6માં Flipkart વેબસાઇટનું હોમપેજ દર્શાવ્યું છે. વિવિધ પ્રકારની વસ્તુ ખરીદવા માટે આ ઘણી પ્રચલિત વેબસાઇટ છે.



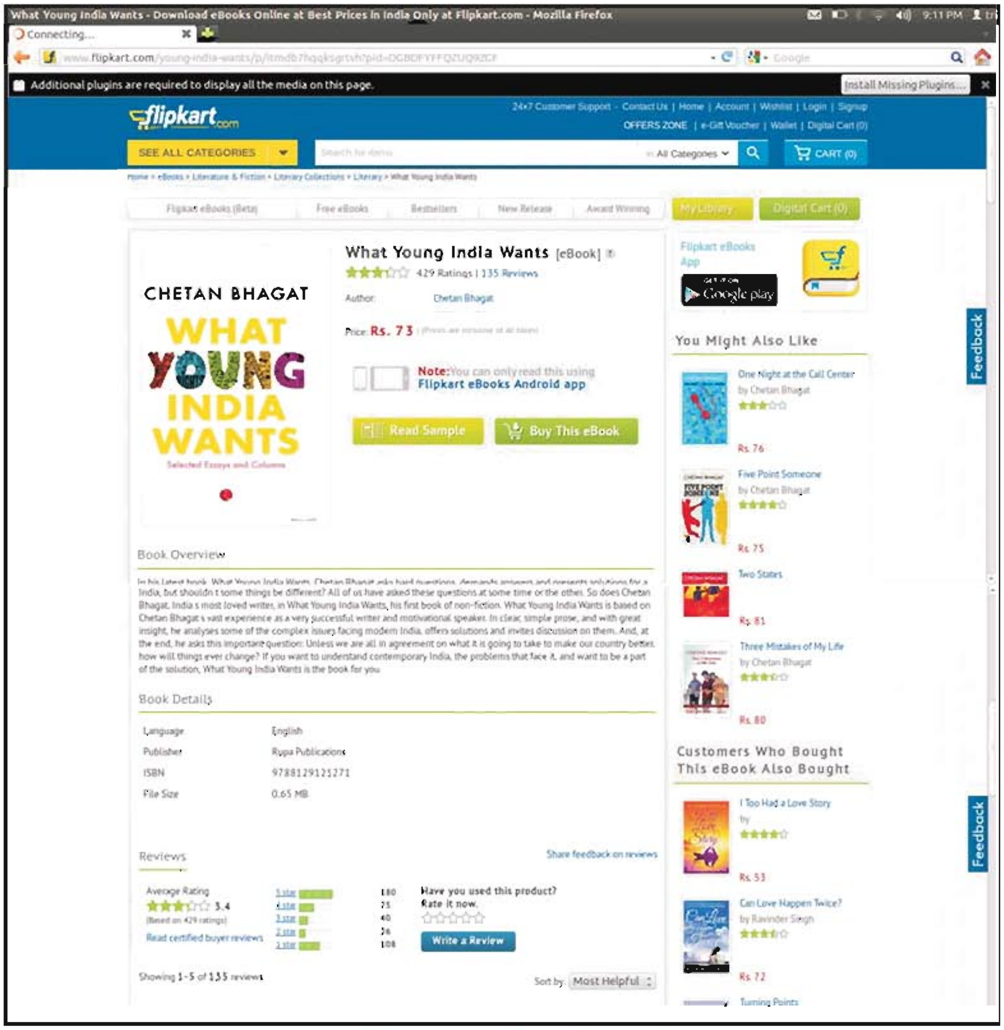
આકૃતિ 4.6 : Flipkartનું હોમપેજ

આકૃતિ 4.6માં દર્શાવ્યા મુજબ વિન્ડોની ડાબી બાજુ ઉત્પાદનોને વર્ગીકૃત સ્વરૂપે દર્શાવવામાં આવે છે. તમારી ઇચ્છા મુજબનો વર્ગ પસંદ કરો. આકૃતિ 4.7માં દર્શાવ્યા મુજબ અહીં eBooks વર્ગ પસંદ કરવામાં આવ્યો છે. તે તમામ ઇ-બુક્સને સૂચિ (category) દ્વારા પ્રદર્શિત કરે છે.



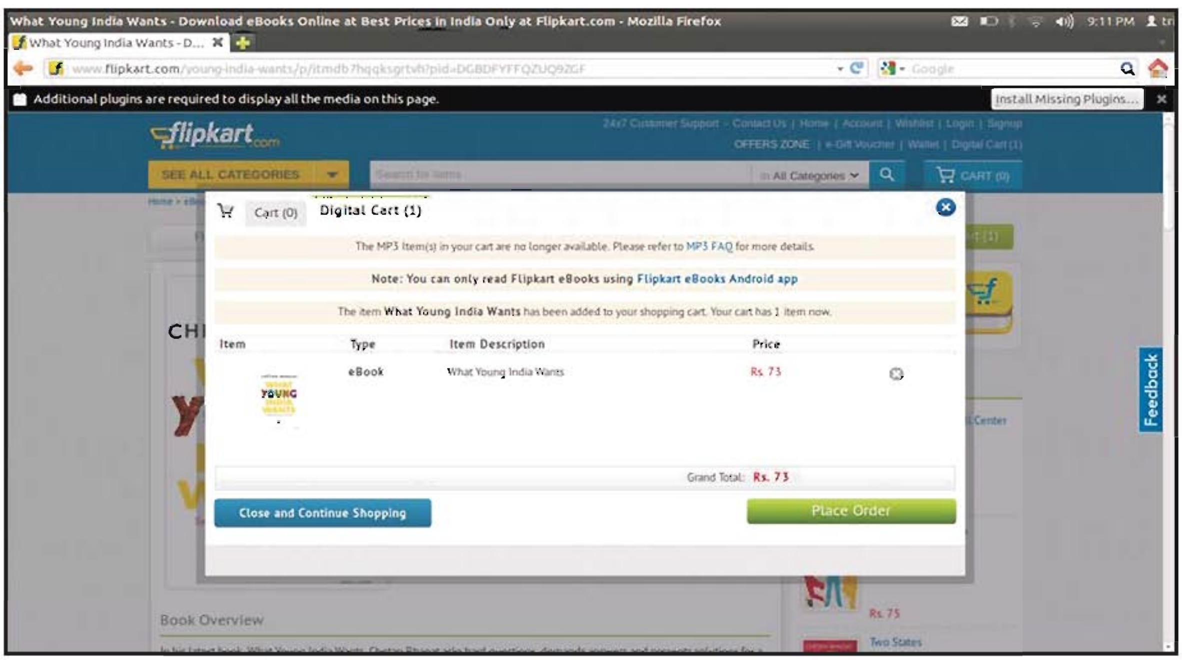
આકૃતિ 4.7 : Flipkart.comમાં eBooks માટેનો વર્ગ પસંદ કરવો

જ્યારે કોઈ પણ ઇ-બુક પર ક્લિક કરવામાં આવે છે, ત્યારે આકૃતિ 4.8માં દર્શાવ્યા મુજબ લેખકનું નામ, કિંમત, વાચકોના રેટિંગ, પુસ્તકના પ્રતિભાવ, પ્રકાશકનું નામ અને ISBN અંક અંગેની વિગતો રજૂ કરવામાં આવે છે. વિન્ડોના સૌથી ઉપરના ભાગમાં આપેલ સર્ચ-ટૂલનો ઉપયોગ કરી વિવિધ વર્ગોમાંથી નિશ્ચિત પુસ્તકને શોધી શકાય છે.



આકૃતિ 4.8 : ઈ-બુકની પસંદગી

"Buy this eBook" બટન પર ક્લિક કરવાથી તેને આકૃતિ 4.9માં દર્શાવ્યા મુજબ શૉપિંગકાર્ટમાં ઉમેરવામાં આવશે.



આકૃતિ 4.9 : પસંદ કરેલી ઈ-બુક દર્શાવતું શૉપિંગકાર્ટ

આવી જ રીતે, તમે અનેક વર્ગોમાંથી તમારી પસંદગીની ગમે તેટલી વસ્તુઓ પસંદ કરી શૉપિંગકાર્ટમાં ઉમેરી શકો છો, શૉપિંગકાર્ટમાં આપેલ ઉત્પાદન કાઢી નાંખી શકાય છે તથા તેનો જથ્થો પણ બદલી શકાય છે.

ઑર્ડરની પુષ્ટિ કરવા માટે આકૃતિ 4.9માં દર્શાવ્યા મુજબ 'Place Order' બટન પર ક્લિક કરો. આમ કરવાથી ગ્રાહકને અંતિમ સ્ક્રીન પર લઈ જવામાં આવશે અને જો લૉગ-ઇન કરેલું ન હોય, તો તે માટે પૂછવામાં આવશે. અહીં બિલિંગ માટેનું સરનામું, ઑર્ડરનો સારાંશ અને ચુકવણીનો વિકલ્પ પૂરો પાડવામાં આવશે. એકવાર આ તમામ પ્રક્રિયા પૂર્ણ થઈ જાય પછી ખરીદ-ઑર્ડર તૈયાર કરવામાં આવે છે અને ગ્રાહકે આપેલ સરનામાં પર ઇ-બુક મોકલી આપવામાં આવે છે. હાલમાં 'માલ મેળવ્યા બાદ ચુકવણી'નો વિકલ્પ પણ પૂરો પાડવામાં આવે છે, જેના દ્વારા ગ્રાહક ખરીદેલું ઉત્પાદન પોતાના સરનામા પર મેળવ્યા બાદ ચુકવણી કરે છે.

વ્યવસાયીથી વ્યવસાયી (Business to Business)

જુદા જુદા વ્યવસાયીઓ વચ્ચે થતી ઇ-કૉમર્સની પ્રવૃત્તિઓને 'વ્યવસાયીથી વ્યવસાયી' પ્રતિકૃતિનો સંદર્ભ આપવામાં આવે છે. B2Bમાં વેચનાર અને ખરીદનાર બન્ને વ્યવસાયી છે. તે પુરવઠાકાર (suppliers), વિતરક (distributors) કે અન્ય મધ્યસ્થી (agent) સાથે ઇ-સંબંધ (e-relationship)ની સ્થાપના કરે છે. સામેલ થયેલા વ્યવસાયીઓ વચ્ચે વધુ પારદર્શિતાને કારણે વધુ કાર્યક્ષમતા પ્રાપ્ત કરી શકાય છે. ઉદાહરણ તરીકે, સંગ્રાહક, વિતરક અને જથ્થાબંધ વેપારી સાથે ઉત્પાદક સોદો કરે છે. B2Bનો ઉપયોગ કરી ઉત્પાદક વિતરક સાથે માલના જથ્થામાં થતા ઘટાડા વિશે તત્કાલ સંવાદ કરી શકે છે તથા વિતરક તેનો ત્વરિત પ્રત્યુત્તર આપી શકે છે.

સંસ્થાઓ તેમની સામાન્ય વ્યાવસાયિક પ્રવૃત્તિઓ જેવી કે પુરવઠાકારનું વ્યવસ્થાપન, માલના જથ્થાની યાદીનું વ્યવસ્થાપન, ચુકવણીનું વ્યવસ્થાપન વગેરેની કાર્યક્ષમતામાં B2Bની મદદથી વધારો કરી શકે છે. તે ટેલિમાર્કેટિંગ (telemarketing)ના વ્યવસ્થાપન, પુરવઠા-સાંકળ (supply chain), માલની પ્રાપ્તિ, નિયમિત સમયે માલ પહોંચાડવો, ઑનલાઇન સેવા વગેરે માટે અસરકારક માધ્યમ છે. commodity.com અને tradeindia.com વેબસાઇટ B2Bનાં ઉદાહરણ છે.

ગ્રાહકથી ગ્રાહક (Consumer to Consumer)

ગ્રાહકો વચ્ચે થતી ઇ-કૉમર્સની પ્રવૃત્તિના વ્યવહારોને 'ગ્રાહકથી ગ્રાહક' પ્રતિકૃતિનો સંદર્ભ આપવામાં આવે છે. કોઈ પણ ત્રાહિત પક્ષની સંડોવણી વગર ઑનલાઇન હરાજી અને જાહેરાત દ્વારા તે ગ્રાહકોને પરસ્પર સોદા કરવાની સુવિધા પૂરી પાડે છે. C2C વેબસાઇટ પર ઇન્ટરનેટનો ઉપયોગકર્તા વેચનાર કે ખરીદનાર બની શકે છે. C2C પ્રતિકૃતિનું ઉત્તમ ઉદાહરણ હરાજીની સાઇટ છે. જો આપણે કોઈ ઉત્પાદન વેચવું હોય, તો તેને હરાજીની સાઇટ પર યાદી સ્વરૂપે મૂકી શકીએ છીએ અને અન્ય વ્યક્તિઓ તેની બોલી લગાવે છે. eBay.com, OLX.com અને Quikr.com એ C2C વેબસાઇટનાં ઉદાહરણ છે.

ગ્રાહકથી વ્યવસાયી (Consumer to Business)

જ્યારે ગ્રાહક દ્વારા ઉત્પાદન કે સેવાની કિંમત નક્કી કરવામાં આવે, ત્યારે ગ્રાહકથી વ્યવસાયી પ્રતિકૃતિ અવળી હરાજીને સમાવતી પદ્ધતિ છે. આ પ્રકારના ઇ-કૉમર્સમાં ગ્રાહકને પરવડી શકે તે રીતે અથવા તો નિશ્ચિત ઉત્પાદન કે સેવા માટે તેઓ ચૂકવવા ઇચ્છતા હોય તે કિંમતથી વ્યાપક શ્રેણીમાં ઉત્પાદન અને સેવાની પસંદગી મળે છે. સંસ્થાઓ ગ્રાહકને ઉત્પાદન કે સેવા આપવા માટે બોલી લગાવે છે. આ પદ્ધતિ ભાવતાલના સમયમાં ઘટાડો કરવામાં મદદરૂપ બને છે તથા ગ્રાહક અને સંસ્થા બંનેની લવચિકતા (flexibility) માં વધારો કરે છે. ખરીદીની સામાન્ય પ્રક્રિયાને ઉલટાવવા માટે C2B ઇન્ટરનેટનો ઉપયોગ કરે છે, જ્યાં ગ્રાહક દ્વારા ચુકવણી નક્કી કરવામાં આવે છે અને સંસ્થા નક્કી કરે છે કે તે સ્વીકારવું કે નહીં. bidstall.com અને JeetLe.in એ C2B પ્રકારની વેબસાઇટનાં ઉદાહરણ છે.

ઇ-કૉમર્સની વ્યવસાય પ્રતિકૃતિઓમાં કેટલીક અન્ય વિવિધતા પણ છે. જો આપણે સરકારને એક સ્વાયત્ત અસ્તિત્વ માની લઈએ, તો તે સંદર્ભે નીચે જણાવેલ પ્રતિકૃતિઓ પણ અસ્તિત્વ ધરાવે છે :

- સરકારથી વ્યવસાયી (Government to Business - G2B)
- સરકારથી નાગરિક (Government to Citizen - G2C)
- સરકારથી સરકાર (Government to Government - G2G)

સરકારથી વ્યવસાયી (Government to Business)

સરકારી વેબસાઇટના વિશાળ નેટવર્કનો ઉપયોગ કરી સરકાર દ્વારા વ્યવસાયી સંસ્થાઓને પૂરી પાડવામાં આવતી સેવાઓ અને માહિતીને સરકારથી વ્યવસાયી (G2B)ના સંદર્ભે જોવામાં આવે છે. વ્યવસાયી આ વેબસાઇટ પરથી સંસ્થાઓ સંબંધિત માહિતી જેવી કે, વ્યવસાયનીતિ, વ્યવસાય શરૂ કરવાની મંજૂરી, વ્યવસાયના સુયોજન (setup) માટેની જરૂરિયાતો અને અન્ય વિવરણો વગેરે મેળવી શકે છે. અહીં સરકારી કાર્યાલયોને લગતાં વિવિધ ફૉર્મ ભરીને રજૂ કરી શકાય છે. ઉદાહરણ તરીકે, ભારતીય સરકારના કરવેરા વિભાગની વેબસાઇટ www.incometaxindia.gov.in છે, જેમાં કરવેરાને લગતા તમામ નિયમો, વિવિધ ફૉર્મ અને કરવેરાનાં ઑનલાઇન રિટર્ન ભરવા માટેની સુવિધા આપવામાં આવી છે. આ વ્યવસાયી પ્રતિકૃતિ ઇ-ગવર્નન્સની એક પહેલ છે.

સરકારથી નાગરિક (Government to Citizen)

સરકારથી નાગરિક (G2C) પણ ઇ-ગવર્નન્સનો એક ભાગ છે. આ વ્યવસાયી પ્રતિકૃતિનો હેતુ સ્વતંત્ર નાગરિકને સારી અને અસરકારક સેવાઓ પૂરી પાડવાનો છે. આ વેબસાઇટ જુદાં-જુદાં સરકારી ખાતાંઓ, વિવિધ કલ્યાણકારી યોજનાઓ, તથા નાગરિકોને ઉપયોગી અનેક પ્રકારનાં અરજીપત્રકો વગેરેની માહિતી પૂરી પાડે છે. આ ઉદ્દેશ્યની પૂર્તિ માટે ગુજરાત સરકારે Gujarat State Wide Area Network (GSWAN) નામનું પોતાનું નેટવર્ક વિકસિત કર્યું છે, જે www.gswan.gov.in પર ઑનલાઇન ઉપલબ્ધ છે. આકૃતિ 4.10 GSWANનું હોમપેજ દર્શાવે છે.

આકૃતિ 4.10 : GSWAN વેબસાઇટનું હોમપેજ

સરકારથી સરકાર (Government to Government)

એક સરકારી સંસ્થા, એજન્સી કે વિભાગ અન્ય સરકારી સંસ્થા, એજન્સી કે વિભાગ સાથે વાણિજ્યરહિત (non-commercial) સંદેશાવ્યવહાર કરે, તો તેને સરકારથી સરકાર (G2G) તરીકે ઓળખવામાં આવે છે. IT ખર્ચાને ઘટાડવા, પ્રક્રિયાઓને સુનિયોજિત બનાવવા અને કાર્યાલયોને વધુ અસરકારક બનાવવા આ માહિતીનું વિવરણ મદદરૂપ બને છે.

ઉપર વિવેચિત તમામ ઇ-કૉમર્સ પ્રતિકૃતિઓમાં B2C અને B2B વિપુલ પ્રમાણમાં ઉપયોગમાં લેવામાં આવતી પ્રતિકૃતિઓ છે. આ બંને પ્રતિકૃતિમાં મુખ્ય તફાવત 'ગ્રાહક' અંગે છે. B2B પ્રતિકૃતિમાં ગ્રાહક એક સંસ્થા છે, જ્યારે B2C પ્રતિકૃતિમાં ગ્રાહક એક સ્વતંત્ર વ્યક્તિ છે.

સારાંશ

આ પ્રકરણમાં આપણે ઇ-કૉમર્સ, તેના ફાયદા અને ગેરફાયદા, ઇ-કૉમર્સના વિનિયોગો અને તેની વ્યવસાય પ્રતિકૃતિઓ વિશે ચર્ચા કરી. વ્યાવસાયિક પ્રવૃત્તિઓ માટે કરવામાં આવતા ઇન્ટરનેટના ઉપયોગને ઇ-કૉમર્સ તરીકે વ્યાખ્યાયિત કરવામાં આવે છે. ઇ-કૉમર્સનો ઉપયોગ માર્કેટિંગ અને વેચાણ, ઉત્પાદનની ખરીદી, ઑનલાઇન વર્તમાનપત્ર, માહિતી સેવાઓ, સહાયસેવાઓ, નેટબૅંકિંગ અને તેને સમકક્ષ અન્ય અનેક પ્રવૃત્તિઓ માટે કરવામાં આવે છે. ઇ-કૉમર્સની વ્યવસાય-પ્રતિકૃતિનું વર્ગીકરણ તેમાં સમાવિષ્ટ પક્ષો અને તેના દ્વારા પૂરી પાડવામાં આવતી વ્યાવસાયિક પ્રવૃત્તિઓ કે સેવાના પ્રકારના આધારે કરવામાં આવે છે. આપણે વ્યવસાયીથી ગ્રાહક (B2C), વ્યવસાયીથી વ્યવસાયી (B2B), ગ્રાહકથી ગ્રાહક (C2C), ગ્રાહકથી વ્યવસાયી (C2B) તેમજ સરકારથી વ્યવસાયી (G2B) જેવી પ્રતિકૃતિઓનો અભ્યાસ કર્યો.

સ્વાધ્યાય

1. ઇ-કૉમર્સને વ્યાખ્યાયિત કરો. ઇ-કૉમર્સના વિનિયોગની યાદી બનાવો.
2. ઑનલાઇન હરાજી એટલે શું ?
3. ઇલેક્ટ્રૉનિક વર્તમાનપત્ર એટલે શું ? આ પ્રકારની વેબસાઇટની વિશેષતાઓની યાદી બનાવો.
4. પુસ્તકોની ઑનલાઇન દુકાનોની યાદી બનાવો અને કોઈ પણ એકની વિશેષતાઓ વિશે ચર્ચા કરો.
5. નેટબૅંકિંગ એટલે શું ? ઑનલાઇન બૅંકિંગ માટેની કેટલીક વેબસાઇટનાં નામ આપો.
6. પરંપરાગત વ્યવસાય અને ઇ-કૉમર્સ વચ્ચેનો તફાવત સ્પષ્ટ કરો.
7. ઇ-કૉમર્સના ફાયદા જણાવો.
8. નીચે આપેલ ઇ-કૉમર્સની વ્યવસાય પ્રતિકૃતિઓ વિશે માહિતી આપો. દરેકનાં ઉદાહરણ પણ આપો :
   (1) વ્યવસાયીથી ગ્રાહક (Business to Consumer - B2C)
   (2) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)
   (3) ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)
   (4) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)
   (5) સરકારથી વ્યવસાયી (Government to Business - G2B)
9. **આપેલ વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :**
   (1) નીચેનાંમાંથી પુસ્તકોની ઑનલાઇન દુકાનનું ઉદાહરણ કયું છે ?
   (a) Amazon (b) irctc (c) Gmail (d) yahoo

(2) ઇન્ટરનેટ પર ડિજિટલ સ્વરૂપે ઉપલબ્ધ વર્તમાનપત્રને શું કહે છે ?

(a) I-newspaper (b) Internet-newspaper

(c) www-newspaper (d) E-newspaper

(3) કઈ પ્રક્રિયા દ્વારા ગ્રાહકને ઉત્પાદનની કિંમત માટે બોલી લગાવીને ખરીદ કે વેચાણની સુવિધા પૂરી પાડવામાં આવે છે ?

(a) માર્કેટિંગ (b) હરાજી (c) પુસ્તકોની દુકાન (d) નોંધણી

(4) ઇન્ટરનેટ પર કરવામાં આવતા બૅન્કના વ્યવહારોને નીચેનામાંથી કઈ પ્રક્રિયા તરીકે ઓળખવામાં આવે છે ?

(a) હરાજી (b) બોલી (c) નેટબૅંકિંગ (d) www-બૅંકિંગ

(5) નીચેનામાંથી કઈ વિશેષતા પરંપરાગત વ્યવસાયની છે ?

(a) નિશ્ચિત સમયગાળા માટે અથવા તો વ્યવસાયના કલાકોમાં પ્રક્રિયા કરવામાં આવે.

(b) ઇન્ટરનેટ પર જાહેરાત કરવામાં આવે.

(c) ચુકવણી મેળવવા માટે ઇ-ચુકવણીની પદ્ધતિનો ઉપયોગ કરવામાં આવે.

(d) ગ્રાહકો ઉત્પાદનો અને યોજનાઓ શોધી શકે.

(6) ઇ-કૉમર્સમાં નીચેનામાંથી કઈ વિશેષતા જોવા મળે છે ?

(a) નિશ્ચિત સમયગાળા માટે અથવા તો વ્યવસાયના કલાકોમાં પ્રક્રિયા કરવામાં આવે.

(b) સ્પર્ધકો સાથે કોઈ માહિતી વહેંચવામાં ન આવે.

(c) સ્થળને ભાડે લેવું પડે અથવા ખરીદવું પડે.

(d) ઉત્પાદનની જાહેરાત ઇન્ટરનેટ દ્વારા કરવામાં આવે.

(7) ઇ-કૉમર્સ દ્વારા નીચેનામાંથી કયો ફાયદો થતો નથી ?

(a) ઓછી કિંમત (b) વ્યવસાયનો 24×7 સમય

(c) સુરક્ષા (d) ભૌગોલિક મર્યાદાનો અભાવ

(8) નીચેનામાંથી ઇ-કૉમર્સનો ગેરફાયદો કયો છે ?

(a) ગોપનીયતા (b) ગ્રાહકસેવામાં સુધારો

(c) ઝડપ (d) વ્યવસાયનો 24×7 સમય

(9) ઉત્પાદન કે સેવા પૂરી પાડતા વ્યાવસાયિકો કે સંસ્થાઓ ઇન્ટરનેટની વેબસાઇટ પર કઈ વ્યવસાયી પ્રતિકૃતિનો ઉપયોગ કરે છે ?

(a) વ્યવસાયીથી ગ્રાહક (Business to Consumer - B2C)

(b) વ્યાવસાયીથી વ્યવસાયી (Business to Business - B2B)

(c) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

(d) સરકારથી વ્યવસાયી (Government to Business - G2B)

(10) જુદા-જુદા વ્યવસાયી ભાગીદારો વચ્ચે કરવામાં આવતી પ્રવૃત્તિઓને ઇ-કૉમર્સની કઈ પ્રતિકૃતિ કહેવામાં આવે છે ?

(a) સરકારથી વ્યવસાયી (Government to Business - G2B)

(b) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

(c) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)

(d) વ્યવસાયીથી ગ્રાહક (Business to Consumer - B2C)

(11) નીચેનાંમાંથી કયું ઉદાહરણ C2C પ્રતિકૃતિનું છે ?

(a) હરાજી માટેની સાઇટ (b) ઇ-વર્તમાનપત્ર

(c) ઑનલાઇન ખરીદી (d) માહિતીસેવા

(12) ગ્રાહકો વચ્ચે કરવામાં આવ્યા હોય, તેવા વ્યવહારોને સમાવતી પ્રવૃત્તિને ઇ-કૉમર્સની કઈ વ્યવસાય પ્રતિકૃતિ કહે છે ?

(a) સરકારથી વ્યવસાયી (Government to Business - G2B)

(b) ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)

(c) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)

(d) વ્યવસાયીથી ગ્રાહક (Business to Consumer - B2C)

(13) નીચેનામાંથી કઈ ઇ-કૉમર્સ પ્રતિકૃતિમાં અવળી હરાજીનો સમાવેશ કરવામાં આવે છે, કે જેમાં ગ્રાહક દ્વારા ઉત્પાદન કે સેવાની કિંમત નક્કી કરવામાં આવે છે ?

(a) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

(b) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)

(c) ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)

(d) સરકારથી વ્યવસાયી (Government to Business - G2B)

(14) નીચેનામાંથી કઈ ઇ-કૉમર્સની પ્રતિકૃતિ ઇ-ગવર્નન્સનો એક ભાગ છે ?

(a) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)

(b) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

(c) ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)

(d) સરકારથી વ્યવસાયી (Government to Business - G2B)

(15) નીચેનામાંથી કઈ ઇ-કૉમર્સ પ્રતિકૃતિ દ્વારા એક સરકારી એજન્સી, સંસ્થા કે વિભાગ અન્ય સરકારી એજન્સી, સંસ્થા કે વિભાગ સાથે વાણિજ્યરહિત સંદેશાવ્યવહાર કરે છે ?

(a) વ્યવસાયીથી વ્યવસાયી (Business to Business - B2B)

(b) ગ્રાહકથી વ્યવસાયી (Consumer to Business - C2B)

(c) સરકારથી સરકાર (Government to Government - G2G)

(d) ગ્રાહકથી ગ્રાહક (Consumer to Consumer - C2C)

એમ-કૉમર્સનો પરિચય 5

મોબાઇલ કૉમર્સ (Mobile Commerce)ને એમ-કૉમર્સ (M-commerce) તરીકે ઓળખવામાં આવે છે. તે મોબાઇલ ફોન, પર્સનલ ડિજિટલ આસિસ્ટન્ટ (Personal Digital Assistant - PDAs), સ્માર્ટફોન, ટૅબ્લેટ, પામટૉપ કે તેના જેવા કોઈ પણ મોબાઇલ સાધન દ્વારા ઇન્ટરનેટની મદદથી માલ કે સેવાના ખરીદ કે વેચાણને રજૂ કરે છે.

એમ-કૉમર્સ ઉપયોગકર્તાને લવચિકતા (flexibility) અને સાર્વત્રિકતા (ubiquity)ના લાભ પૂરા પાડે છે. મોબાઇલ ફોનના ઉપયોગથી ગ્રાહક કમ્પ્યૂટરના ટર્મિનલ પાસે કે દુકાનમાં પ્રત્યક્ષ હાજર ન હોવા છતાં વ્યાવસાયિક વ્યવહાર કરી શકે છે. આને લગતાં ઉપકરણો ઉપયોગકર્તા જ્યાં જાય ત્યાં તેની પાસે જ હોય છે, જે કોઈ પણ સ્થળેથી ઇન્ટરનેટના ઉપયોગને શક્ય બનાવે છે. પરિવહન વખતે તે વાસ્તવિક સમયના વ્યવહારની સુવિધા આપે છે. સ્માર્ટફોન અને ટૅબ્લેટની લોકપ્રિયતા દિન-પ્રતિદિન થતી વૃદ્ધિને કારણે વધુ ને વધુ ઉપયોગકર્તાઓ એમ-કૉમર્સ તરફ વળે છે. એમ-કૉમર્સનાં કેટલાંક ઉદાહરણ આ મુજબ છે :

- હવાઈ ટિકિટની ખરીદી
- ચલચિત્રની ટિકિટની ખરીદી
- રેસ્ટોરન્ટનું બુકિંગ અને આરક્ષણ
- હોટલનું બુકિંગ અને આરક્ષણ
- શૅરબજારનું પૃથક્કરણ

બૅન્ક અને તેના જેવી અન્ય વાણિજ્યિક સંસ્થાઓ પોતાના વ્યવસાયને સુરક્ષિત રાખવા એમ-કૉમર્સનો વધુ ને વધુ ઉપયોગ કરી રહી છે. તેઓ પોતાના ગ્રાહકને મોબાઇલ ફોન દ્વારા ખાતાની સિલક જાણવા, શૅરબજારની વધઘટ જોવા અને તેમાં ખરીદ-વેચાણ કરવા માટેની સુવિધા આપે છે. આ સેવાને મોબાઇલ-બૅંકિંગ (Mobile Banking) અથવા એમ-બૅંકિંગ (M-Banking) કહે છે. મોબાઇલ ઉપકરણો દ્વારા પૂરી પાડવામાં આવતી શૅરબજારને લગતી સેવાઓ પણ હવે પ્રચલિત બની રહી છે અને તેને મોબાઇલ બ્રોકરેજ (Mobile Brokerage) તરીકે ઓળખવામાં આવે છે. મોબાઇલ સંબંધિત સેવાઓની માંગણી સાથે સમાચાર-માહિતી, રમતગમત, મનોરંજન, ખરીદી અને આરક્ષણ અંગેની માહિતી જેવાં ક્ષેત્રો પણ વિસ્તાર પામ્યાં છે.

એમ-કૉમર્સના લાભ (Benefits of M-Commerce)

મોબાઇલના ઉપયોગમાં ઉત્તરોત્તર વૃદ્ધિ થઈ રહી છે અને તે હવે માત્ર પ્રત્યાયન-વ્યવહાર એટલે કે પરસ્પર વાત કરવા પૂરતો મર્યાદિત રહ્યો નથી. મોબાઇલ સંસ્થાઓ નવી સુવિધાઓ ધરાવતા સ્માર્ટફોન સાથે આવી રહી છે, જેના દ્વારા ગ્રાહકને સરળતા, લવચિકતા અને સુરક્ષા એકસાથે પૂરી પાડવામાં આવે છે. સહજ ઉપલબ્ધતા અને ગતિશીલતાને કારણે એમ-કૉમર્સ આજકાલ વધુ પ્રચલિત થઈ રહ્યું છે. વેબવિકાસ કરતી સંસ્થાઓ પણ મોબાઇલ ઉપકરણો પર વેબસાઇટ યોગ્ય રીતે દર્શાવી શકાય તે માટે સક્રિયતા દાખવી રહ્યા છે.

એમ-કૉમર્સ વાયરલેસ નેટવર્ક અને ઇન્ટરનેટનું એવું સંચાલન છે, જેને હસ્તગત ઉપકરણ દ્વારા સતત પાસે રાખી શકાય છે. ઇન્ટરનેટ અને ઇ-કૉમર્સના ફાયદા એમ-કૉમર્સ દ્વારા પણ ઉપલબ્ધ થાય છે. એમ-કૉમર્સના કેટલાક લાભ નીચેની યાદીમાં દર્શાવ્યા છે :

- એમ-કૉમર્સ ઉપયોગકર્તાને અનુકૂલન પૂરું પાડે છે. મોબાઇલ ઉપકરણ પરની કેટલીક ક્લિક દ્વારા ઉપયોગકર્તા પરિવહનના સમય દરમિયાન પણ ખરીદી, બૅંકિંગ અને મીડિયા ફાઇલ ડાઉનલોડ કરવા જેવી પ્રવૃત્તિઓ કરી શકે છે.

- મોબાઇલ સાધન દ્વારા ઉપયોગકર્તાનો પરોક્ષ રીતે કોઈ પણ સમયે ને સ્થાને સંપર્ક કરી શકાય છે.
- વ્યવહારખર્ચમાં ઘટાડો થાય છે.
- વિક્રેતાને ઑર્ડર આપવામાં ઓછો સમય લાગે છે. તદુપરાંત ઉપયોગકર્તા પાસે આ માટે PC કે લેપટૉપ હોવું જરૂરી નથી.
- વ્યાવસાયિક પ્રક્રિયાઓ સુવ્યવસ્થિત બને છે.
- વૈશ્વિક સંપર્ક શક્ય બને છે.
- વ્યવસાય નિરંતર એટલે કે 24*7 સમયે ચલાવી શકાય છે.
- કોઈ પણ મોબાઇલ ઉપકરણ દ્વારા માહિતીનો ઉપયોગ કરી શકવાની લવચિકતા પૂરી પાડે છે.
- અંગત કમ્પ્યૂટરના ઉપયોગની જેમ જ મોબાઇલ સાધનના ઉપયોગ દ્વારા પણ ચુકવણી કરી શકાય છે.
- સમયાધારિત જટિલ (time critical) અને સંકટકાલીન માહિતી મોકલવા ઉપયોગી છે.
- હસ્તગત ઉપકરણનું ભૌતિક સ્થાન સરળતાથી શોધી શકાય છે. સ્થાન આધારિત વિનિયોગોના ઉદ્ભવથી ઉપયોગકર્તા સુસંગત માહિતી મેળવી શકે છે. આ પ્રકરણમાં આગળ આપણે સ્થાન આધારિત વિનિયોગો વિશે ચર્ચા કરીશું.
- મોબાઇલ ઉપકરણ પર વિવિધ ચેતવણીઓ (Customized alerts) સરળતાથી મેળવી શકાય છે.
- વર્તમાન યુગમાં તત્કાલ જોડાણ અને વધુ ઝડપી 3G સેવાઓની ઉપલબ્ધિએ એમ-કૉમર્સને વધુ પ્રચલિત બનાવ્યું છે.
- ઉપયોગકર્તા સુધી માહિતી સમયસર પહોંચાડી શકાય છે. હવાઈ કે ટ્રેનની અનુસૂચિ, વિલંબ કે રદ કર્યાની માહિતી ઉપયોગકર્તાને તેના મોબાઇલ ઉપકરણ પર વાસ્તવિક સમયે પૂરી પાડી શકાય છે.

એમ-કૉમર્સની મર્યાદાઓ (Limitations of M-commerce)

એમ-કૉમર્સના લાભની યાદી લાંબી હોવા છતાં તેની કેટલીક મર્યાદાઓ પણ છે, જે નીચે દર્શાવેલી છે :

- હાલમાં ઉપલબ્ધ હસ્તગત (handheld) સાધનો સ્ક્રીનનું મર્યાદિત કદ ધરાવે છે. વળી, તે ફાઇલના પ્રકાર અને ડેટાના સ્થાનાંતરમાં પણ મર્યાદિત છે. ઘણીવાર વીડિયો દર્શાવવા પણ મુશ્કેલ બને છે.
- અંગત કમ્પ્યૂટરની સરખામણીમાં ઉપયોગકર્તા માટેનો ઇન્ટરફેસ ઓછો અનુકૂળ હોય છે.
- મોબાઇલ ઉપકરણો પાસે કમ્પ્યૂટિંગ ક્ષમતા, મેમરી અને સંગ્રહક્ષમતા મર્યાદિત હોય છે.
- એમ-કૉમર્સ વાયરલેસ નેટવર્ક પર ચાલે છે, જે વાયરયુક્ત નેટવર્કની તુલનામાં ઓછું સુરક્ષિત છે.
- તે મર્યાદિત બેન્ડવિડ્થ (ગતિ) ધરાવે છે.
- મોબાઇલ અને વાયરલેસ બ્રૉડબેન્ડ માળખાંને સ્થાપિત કરવાનો ખર્ચ વધુ હોય છે.

એમ-કૉમર્સના વિનિયોગ (Applications of M-Commerce)

એમ-કૉમર્સના વિનિયોગ ઝડપથી વૃદ્ધિ પામી રહ્યા છે. તેના ઉપયોગ અહીં ચર્ચવામાં વિસ્તારો પૂરતાં મર્યાદિત નથી. નિકટના ભવિષ્યમાં વધુ ને વધુ વિનિયોગોનો વિકાસ થવાનો છે. એમ-કૉમર્સના કેટલાક વિનિયોગ વિશે ચર્ચા કરીએ.

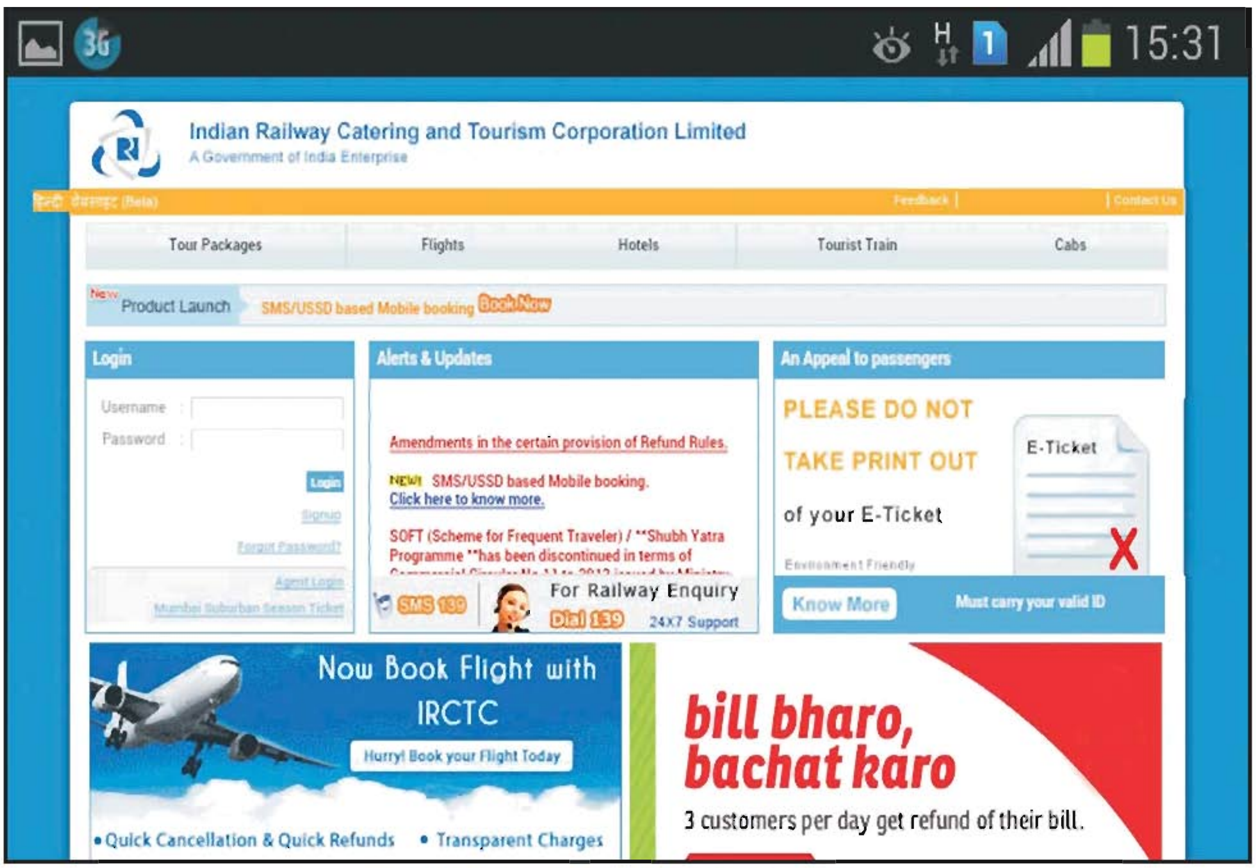
મોબાઈલ માર્કેટિંગ અને વિજ્ઞાપન (Mobile Marketing and Advertising)

હવે સંસ્થાઓ માર્કેટિંગથી વિજ્ઞાપન સુધીની સેવાઓના વિસ્તાર માટે એમ-કૉમર્સનો ઉપયોગ કરે છે. સંસ્થાઓ દ્વારા વિશાળ સંખ્યામાં પ્રેક્ષકો સુધી પહોંચવા માટે મોબાઇલ વિજ્ઞાપનનો માર્ગ સૌથી વધુ પ્રચલિત બન્યો છે. મોટા ભાગના પોર્ટલ માટે ઇન્ટરનેટ પર વિજ્ઞાપન એ આવકનો એક મોટો સ્રોત બની રહ્યો છે. ઘણા છૂટક વેપારીઓ સ્થાન આધારિત મોબાઇલ વિજ્ઞાપનનો વિકલ્પ પૂરો પાડે છે, જેના દ્વારા ગ્રાહકો સુધી પહોંચી વેચાણ વધારી શકાય છે. આમ, મોબાઇલ ઉપકરણ પર મૂકવામાં આવેલ વિજ્ઞાપન અંગત જરૂરિયાત અને ચોકસાઈ સ્થાન પર આધારિત હોઈ શકે છે. ઉપયોગકર્તાના વર્તમાન સ્થાન પાસે આપવામાં આવતા વિવિધ વળતર અને યોજનાઓ અંગે તેને વાકેફ કરી શકાય છે.

મોબાઈલ પર ટિકિટ (Mobile Ticketing)

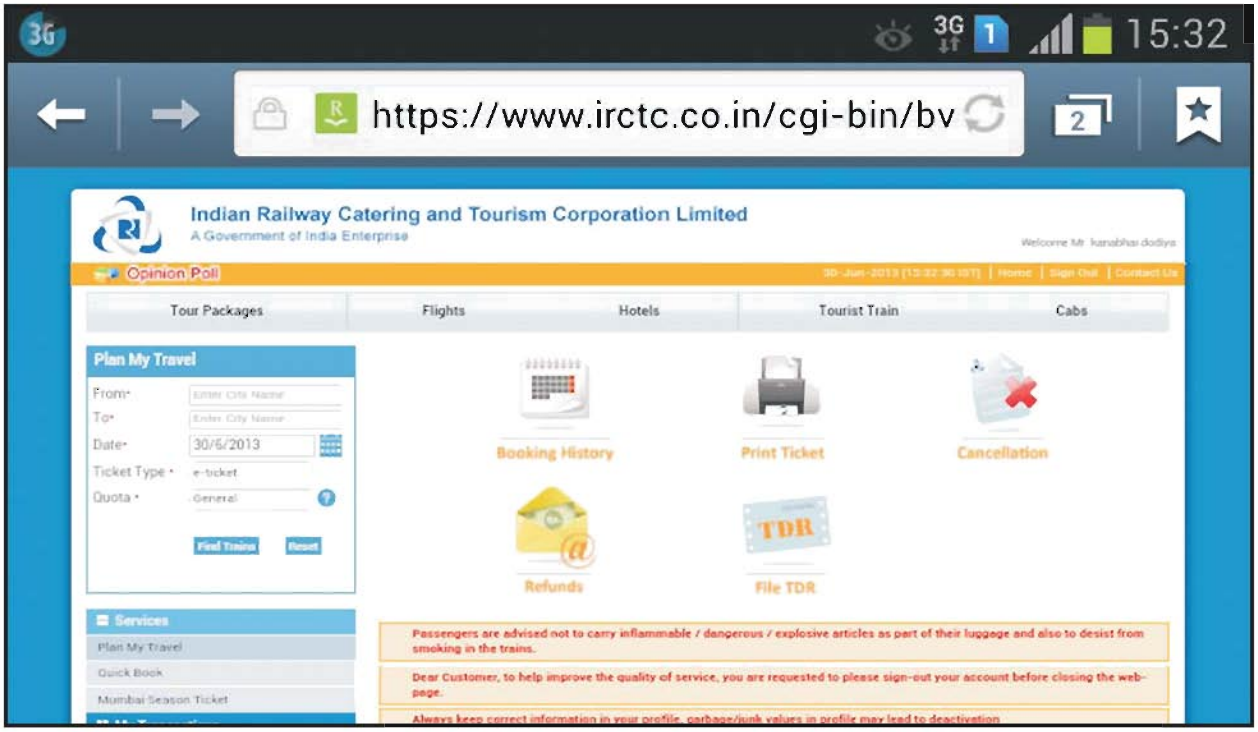
ઉપયોગકર્તા મોબાઇલનો ઉપયોગ કરી સરળતાથી હવાઈ, રેલવે કે ચલચિત્રની ટિકિટ ખરીદી શકે છે. ઉપયોગકર્તાના મોબાઇલ ઉપકરણ પર આ ટિકિટ મોકલવામાં આવે છે. ઉપયોગકર્તા જરૂરી સ્થાન પર પોતાના મોબાઇલ ઉપકરણમાં આ ટિકિટને રજૂ કરી શકે છે. વિનિયોગનો ઉપયોગ કરીને અથવા હવાઈ યાત્રાના એજન્ટનો પોર્ટલનો ઉપયોગ કરીને મોબાઇલ ફોન દ્વારા ટિકિટ સરળતાથી રદ પણ કરાવી શકાય છે. તે અવરજવર કે પાર્કિંગના કષ્ટમાંથી મુક્તિ અપાવે છે, કારણકે ઉપયોગકર્તાએ ટિકિટ ખરીદવા જે-તે સ્થળે જવાની જરૂર રહેતી નથી.

આકૃતિ 5.1 Indian Railway Catering And Tourism Corporation limited (IRCTC)નું લોગ-ઈન પાનું દર્શાવે છે.



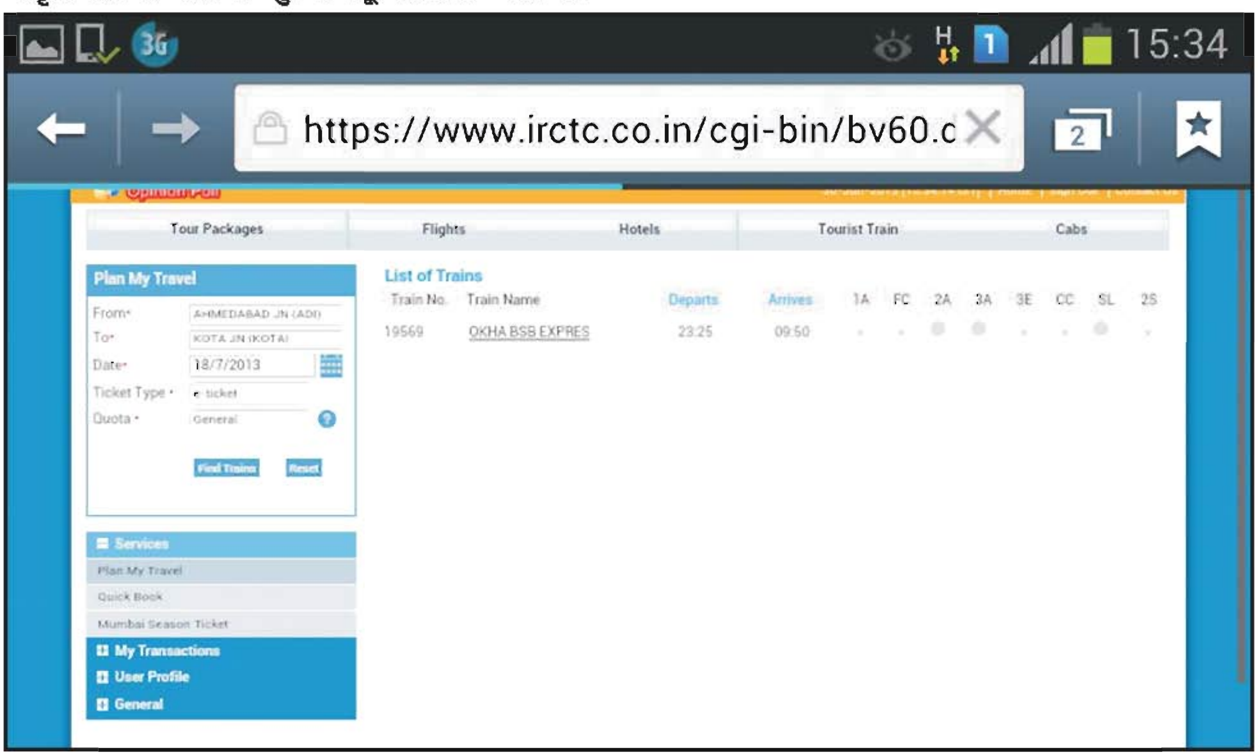
આકૃતિ 5.1 : IRCTCનું લોગ-ઈન પાનું

આપણે મોબાઇલ સાધનનો ઉપયોગ કરી ઑનલાઇન ટિકિટની નોંધણી કરવાનો પ્રયત્ન કરીએ. યૂઝરનેમ અને પાસવર્ડ પૂરાં પાડ્યા બાદ (નોંધ : ઉપયોગકર્તા પાસે પહેલેથી ખાતું છે) ઉપયોગકર્તા આકૃતિ 5.2માં દર્શાવ્યા મુજબ મૂળ સ્થાન (source) અને ગંતવ્ય (destination)નાં નામ, મુસાફરીની તારીખ, ટિકિટનો પ્રકાર અને નિયત હિસ્સા (quota) અંગેની તેની યોજના દાખલ કરી શકે છે.



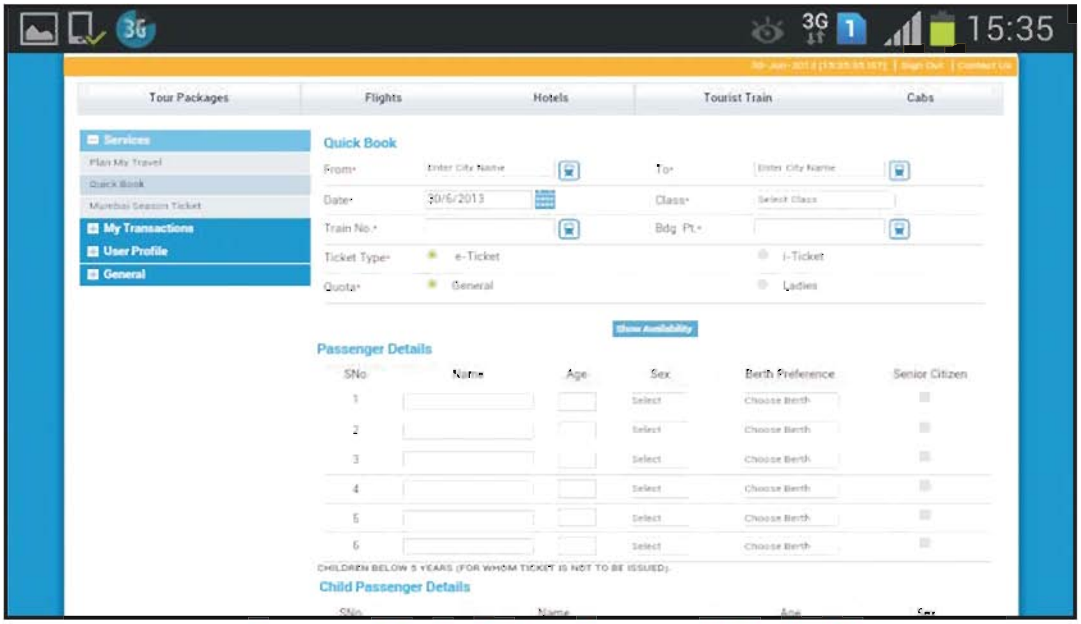
આકૃતિ 5.2 : મુસાફરીની વિગતો

જ્યારે ઉપયોગકર્તા Find Trains બટન પર ક્લિક કરે, ત્યારે તેની જરૂરિયાત મુજબની તારીખે ઉપલબ્ધ ટ્રેનની યાદી આકૃતિ 5.3માં દર્શાવ્યા મુજબ રજૂ કરવામાં આવે છે.



આકૃતિ 5.3 : ઉપલબ્ધ ટ્રેનની યાદી સાથે મુસાફરીની વિગતો

ઉપયોગકર્તા ટ્રેનમાં નામ પર ક્લિક કરી વધુ વિગતો મેળવી શકે છે. આકૃતિ 5.3માં આપણને માત્ર એક ટ્રેનનું નામ જોવા મળી શકે છે, પરંતુ એથી વધુ ટ્રેનની યાદી મેળવવી પણ શક્ય છે. ત્યાર પછી, આકૃતિ 5.4માં દર્શાવ્યા મુજબ ટિકિટની નોંધણી દરમિયાન મુસાફર પોતાની વિગતો પૂરી પાડી શકે છે.

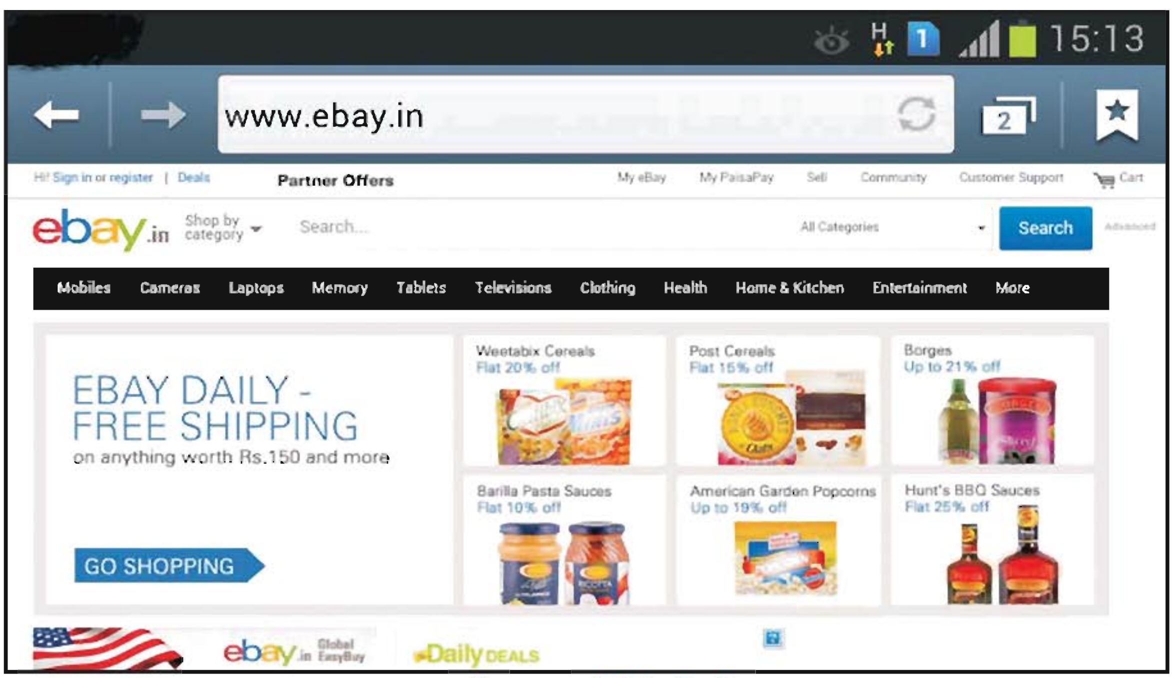


આકૃતિ 5.4 : ટિકિટની નોંધણી માટે મુસાફરની વિગતો ઉમેરવી

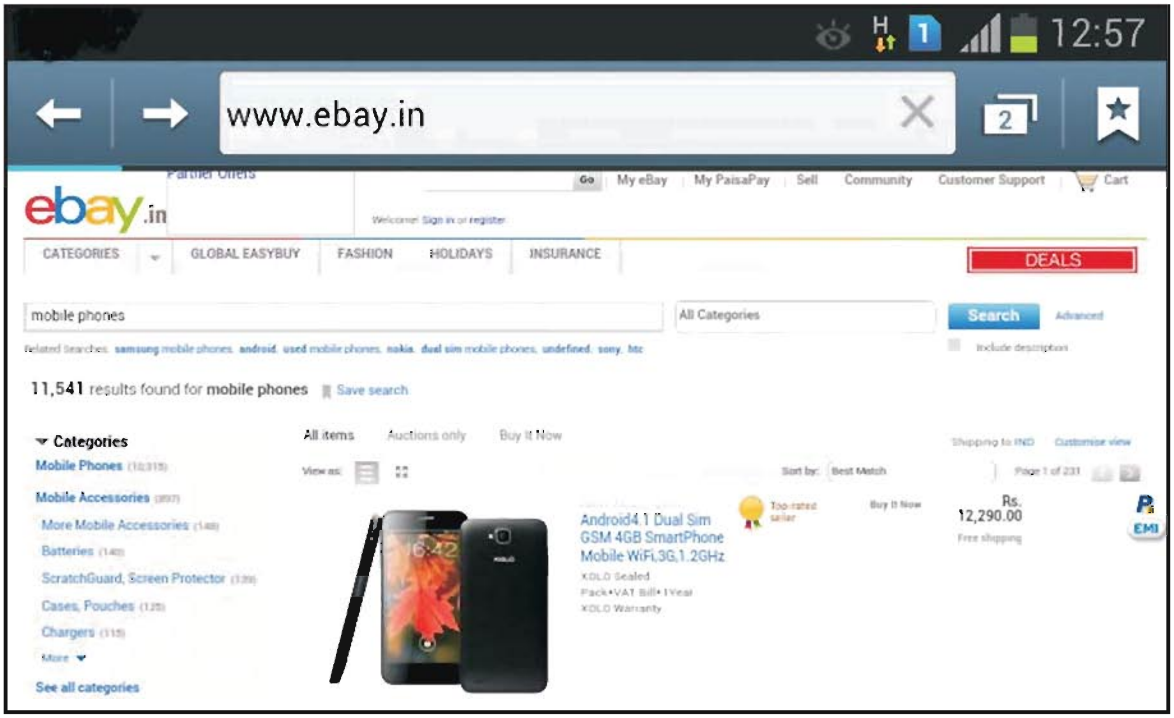
નોંધણી થયા બાદ ઉપયોગકર્તાને IRCTC તરફથી ઇ-ટિકિટનો સંદેશ મોકલવામાં આવે છે, જે મુસાફરી દરમિયાન રજૂ કરી શકાય છે. તમે જોઈ શકો છો કે, કમ્પ્યૂટર સાથે જોડાયા વિના કે રેલવે-સ્ટેશને પ્રત્યક્ષ હાજરી આપ્યા વિના ઉપયોગકર્તા કોઈ પણ સ્થળે, કોઈ પણ સમયે મોબાઇલ ઉપકરણની મદદથી આ સેવાનો ઉપયોગ કરી શકે છે.

મોબાઇલ હરાજી (Mobile Auctions)

હાલના સમયમાં હરાજીની સાઇટ ઘણી પ્રચલિત થઈ રહી છે. મોબાઇલ ઉપકરણ હરાજીની આવી સાઇટનો સંપર્ક સાધવામાં મદદરૂપ બને છે. ઉપયોગકર્તા હરતાં-ફરતાં પણ આ સાઇટનો ઉપયોગ કરી શકે છે, બોલી લગાવી શકે છે, બોલી પર સતત દેખરેખ રાખી શકે છે અને આ પ્રક્રિયામાં યોગ્ય રીતે સમયસર સક્રિય રહી શકે છે. હરાજી માટેની અનેક વેબસાઇટોએ વાયરલેસ નેટવર્ક દ્વારા મોબાઇલ ઉપકરણનો ઉપયોગ કરી શકાય તે પ્રકારના ગેટ-વે અને ઇન્ટરફેસની રચના કરી છે. મોબાઇલ ઉપકરણ પર હરાજી માટેની ઇ-બે સાઇટનું દૃશ્ય આકૃતિ 5.5 અને 5.6માં દર્શાવ્યું છે. ઉપયોગકર્તા મોબાઇલ ઉપકરણ વડે અહીં ઉત્પાદન માટે બોલી લગાવી શકે છે.



આકૃતિ 5.5 : ઇ-બેનું હોમપેજ



આકૃતિ 5.6 : ઉત્પાદન માટે બોલી લગાવવી

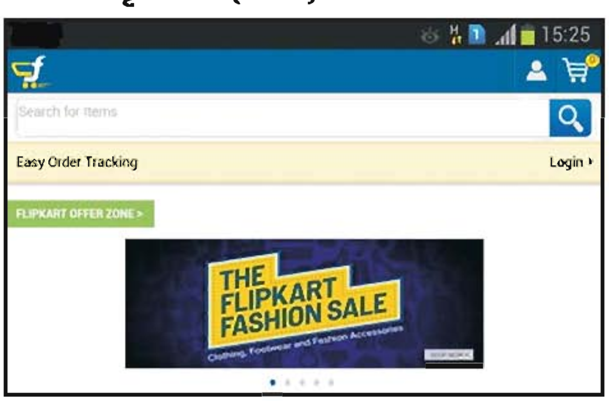
મોબાઈલ દ્વારા મનોરંજન (Mobile Entertainment)

મોબાઇલ સાધનોનો વ્યાપક ઉપયોગ ઑડિયો સાંભળવા, વીડિયો જોવા અને રમતો રમવા માટે કરવામાં આવે છે. મોબાઇલના ઉપયોગકર્તાઓ મનોરંજનની ઑનલાઇન લાઇબ્રેરીમાં સભ્ય થઈ શકે છે, જ્યાં તેઓ ગીતો, વીડિયો કે રમતોને શોધી પોતાના સંગ્રહમાં સરળતાથી ડાઉનલોડ કરી અનુકૂળતા મુજબ તેનો ઉપયોગ કરી શકે છે. વિપુલ સંખ્યાના મોબાઇલ ધારકોને ડાઉનલોડ આધારિત ચુકવણી, કાર્યક્રમ આધારિત ચુકવણી કે લવાજમ આધારિત સેવાઓ પૂરી પાડી શકાય છે. અને તેમણે આ સેવાઓ માટે ચુકવણી કરવા તૈયાર રહેવું પડે છે.

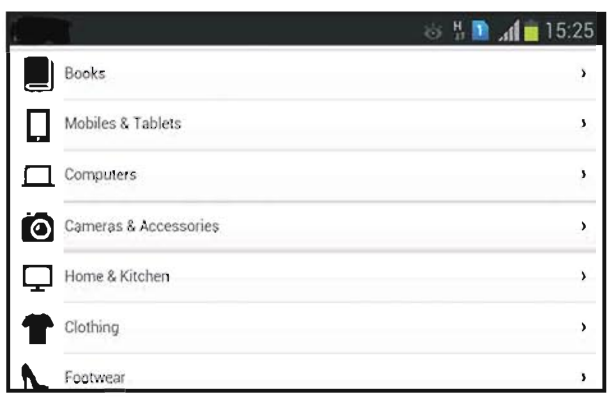
મોબાઈલ દ્વારા ખરીદી (Mobile Purchase)

મોબાઇલ દ્વારા ખરીદીની પ્રક્રિયા ગ્રાહકને કોઈ પણ સમયે ને સ્થળે ઑનલાઇન ખરીદીની સુવિધા પૂરી પાડે છે. ગ્રાહકો ઉત્પાદનની જાણકારી મેળવી ચુકવણીની સુરક્ષિત પદ્ધતિનો ઉપયોગ કરી ખરીદીનો ઑર્ડર આપી શકે છે. છૂટક વેપારી ગ્રાહકને લેખિત સૂચિને બદલે તેને જોઈતાં ઉત્પાદનોની યાદી સીધી જ તેના મોબાઇલ પર મોકલી શકે છે. આ જ રીતે ગ્રાહક છૂટક વેપારીની ઇ-કૉમર્સ સાઇટની મોબાઇલ આવૃત્તિની મુલાકાત લઈ શકે છે. છૂટક વેપારીઓ ગ્રાહકનો સંપર્ક સાધી તેમને સ્થાનિક દુકાનો પર મળતા વળતર વિશેની જાણકારી પૂરી પાડી શકે છે.

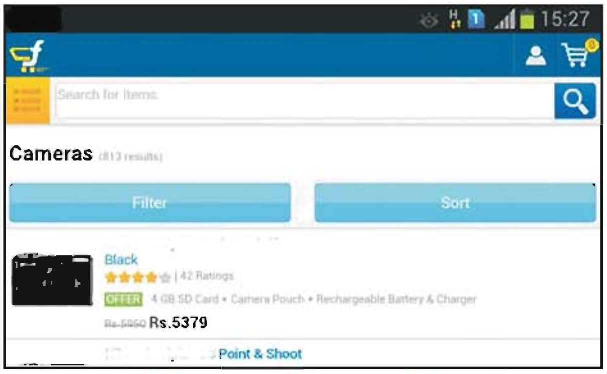
આકૃતિ 5.7(a)માં મોબાઇલ ઉપકરણ પર જોવા મળતું ફ્લિપકાર્ટનું હોમપેજ રજૂ કરવામાં આવ્યું છે. મોબાઇલ દ્વારા થતી ખરીદી સંદર્ભે ઉત્પાદનના વિભાગની પસંદગી, ઉત્પાદનની પસંદગી, ઉત્પાદન અંગેની માહિતી, ઑર્ડર-પ્રક્રિયા અને વિગતો આકૃતિ 5.7 (bથી f)માં દર્શાવવામાં આવેલ છે.



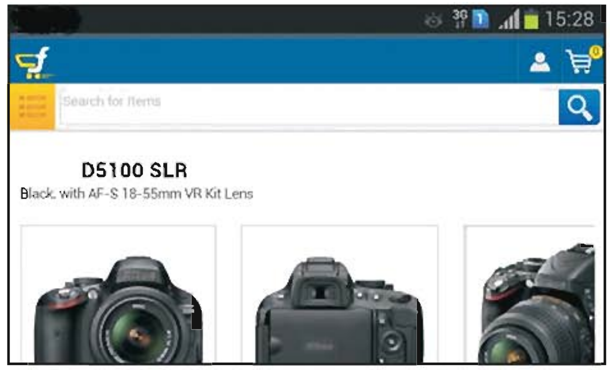
આકૃતિ 5.7(a) : ફ્લિપકાર્ટનું હોમપેજ



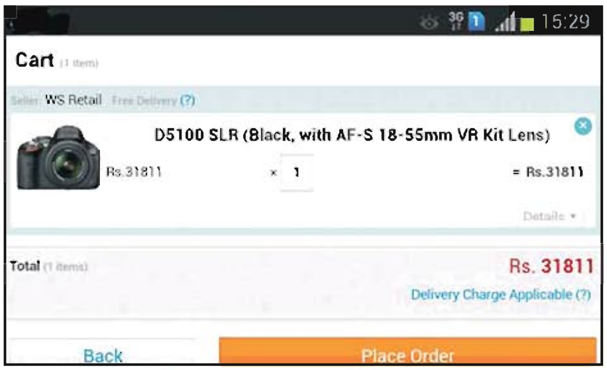
આકૃતિ 5.7(b) : વિભાગની પસંદગી



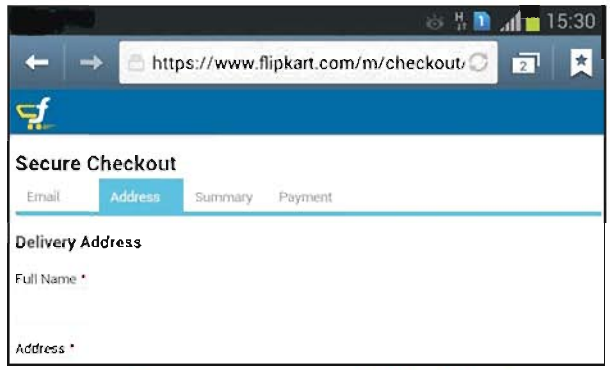
આકૃતિ 5.7(c) : વિભાગના ઉત્પાદનો



આકૃતિ 5.7(d) : ઉત્પાદનની પસંદગી



આકૃતિ 5.7(e) : ઑર્ડર-પ્રક્રિયા



આકૃતિ 5.7(f) : ગ્રાહક સંબંધિત વિગતો

મોબાઈલ દ્વારા નાણાકીય સેવાઓ (Mobile Financial Services)

આજકાલ, ઘણી પ્રતિષ્ઠિત બૅન્કો અને નાણાકીય સંસ્થાઓ એમ-કૉમર્સનો ઉપયોગ કરે છે. તેઓ તેમના ગ્રાહકોને મોબાઈલ ફોન કે અન્ય મોબાઈલ સાધનો દ્વારા તેમનાં ખાતાંની જાણકારી, શેરનું ખરીદ-વેચાણ, દેવાં-માફી વગેરે જેવી સુવિધાઓ પૂરી પાડે છે. યુનિયન બૅન્ક દ્વારા પૂરી પાડવામાં આવતી મોબાઈલ-સેવા આકૃતિ 5.8માં દર્શાવી છે. ઍન્ડ્રોઇડ માર્કેટમાંથી ગ્રાહક "umobile" નામની ઍપ્લિકેશન ડાઉનલોડ કરી શકે છે. નોંધણી થઈ ગયા બાદ, આકૃતિ 5.9માં દર્શાવ્યા મુજબ ગ્રાહક તેના મોબાઈલ સાધન પર બૅન્ક દ્વારા પૂરી પાડવામાં આવેલી સેવાઓ મેળવી શકે છે. મુખ્ય મેનુમાં વિવિધ વિકલ્પો આકૃતિ 5.10માં જોઈ શકાય છે. ગ્રાહક તેની ખાતાકીય સિલક તપાસી શકે છે, અન્ય બૅન્કનાં ખાતાંમાં રકમનું હસ્તાંતરણ કરી શકે છે, ચૅકબુક મેળવવા માટે વિનંતી કરી શકે છે અને આ સિવાય પણ અન્ય પ્રકારની અનેક સેવાઓ પોતાના મોબાઈલ ઉપકરણ દ્વારા મેળવી શકે છે.

આકૃતિ 5.8 : મોબાઈલ દ્વારા નાણાકીય સેવાઓ

આકૃતિ 5.9 : પાસવર્ડની સહાયથી લૉગ-ઈન

આકૃતિ 5.10 : વિવિધ સેવાઓની યાદી

મોબાઈલ દ્વારા માહિતીસેવા (Mobile Information Services)

અંગત કમ્પ્યૂટરની જેમ જ મોબાઇલધારકોને વિપુલ પ્રમાણમાં માહિતીને લગતી સેવાઓ પૂરી પાડી શકાય છે. આમાં સમાવિષ્ટ છે :

- સમાચારસેવા
- શેરબજારની વિગતો
- ખેલકૂદ-સમાચાર
- નાણાકીય નોંધ
- યાતાયાતની માહિતી

સ્થળ અને શોધસેવા (Location and Search Service)

મોબાઇલ કૉમર્સના ઉપયોગ સમયે ઉપયોગકર્તાના મોબાઇલ ફોનનું સ્થાન એક અગત્યની માહિતી બની રહે છે. ઉદાહરણ તરીકે, મર્યાદિત કિંમતમાં અને ચોક્કસ વિશિષ્ટતાઓ સાથે ઉપલબ્ધ સ્માર્ટફોનને ખરીદવા માંગતા ઉપયોગકર્તાને તેની નજીકની દુકાનનું સ્થાન જાણવામાં રસ હશે કે જ્યાંથી તે પોતાની અપેક્ષા મુજબનું ઉત્પાદન મેળવી શકે. સ્થાન અને શોધની સેવા ઉપયોગકર્તાને તેના વર્તમાન સ્થાનથી શહેરની નિકટવર્તી દુકાનોની માહિતી મળે એ જરૂરી છે. ઉપયોગકર્તાના મોબાઇલના સ્થાન અંગેની જાણકારી વ્યાપારીને સ્થાનિક નકશા, સ્થાનિક રજૂઆતો, સ્થાનિક હવામાન, વ્યક્તિ અંગેની ખબર અને દેખરેખ જેવી સ્થળ આધારિત સેવાઓ પૂરી પાડવાની સુવિધા આપે છે. મોબાઇલ સાધનોની મદદથી કોઈ નિશ્ચિત જગ્યા, સિનેમાગૃહ, ભોજનાલય, દવાખાનું કે અન્ય સુવિધાઓ માટેની માહિતી મેળવી શકાય છે. L-commerce નામથી ઓળખાતી સ્થળ આધારિત શોધ વિશે અભ્યાસ કરીએ.

એલ-કૉમર્સ (L-Commerce)

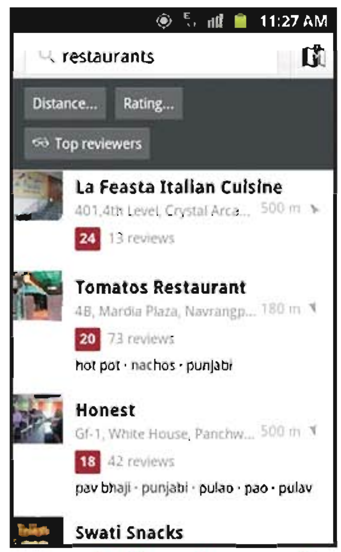
PDA, સેલ્યુલર ફોન અને પૉકેટ પીસી જેવા વધુ ને વધુ વાયરલેસ સાધનોએ એમ-કૉમર્સની વૃદ્ધિ માટે નોંધપાત્ર તકો ઊભી કરી છે. ગ્રાહક કે વિક્રેતાની ઉત્પાદન તથા સેવા સંબંધી બધી જ આવશ્યકતાઓની પૂર્તિ માટે મોબાઇલ કૉમર્સ સક્ષમ સાબિત થયું છે. આમ છતાં, કેટલીક વિશિષ્ટ સ્થિતિઓમાં ગ્રાહક અને વિક્રેતાઓનું નિશ્ચિત સ્થાન વ્યવહાર માટે મહત્ત્વનું હોય છે. આજકાલ સ્થાન આધારિત ઘણા વિનિયોગો અને સેવાઓ આપવામાં આવે છે. આ વિનિયોગો સેવા કે ઉત્પાદનને પહોંચાડવા માટે ઉપયોગકર્તાના સ્થાનની દેખરેખ રાખે છે. વ્યાવસાયિક હેતુ માટે સ્થાનની માહિતી પૂરી પાડતી તક્નિકના ઉપયોગને એલ-કૉમર્સ તરીકે ઓળખવામાં આવે છે.

મોબાઇલ ઉપકરણના ભૌતિક સ્થાનને આધારે તક્નિક વિનિયોગના ઉપર્યુક્ત ક્ષેત્રનું નિર્ધારણ કરે છે. અને તેના દ્વારા ઉપયોગકર્તા સ્થળવિશેષને લૉગ-ઇન કરી શકે છે, અન્ય વ્યક્તિનું સ્થાન જાણી શકે છે અને બૅન્ક કે ભોજનાલય જેવાં સ્થળ શોધી શકે છે. આ તક્નિક GPS, સેલ્યુલર અને Wi-Fi સ્રોત તરફથી મળતા સિગ્નલ પર કાર્ય કરે છે. મોબાઇલ ઉપકરણોની સ્થિતિ નક્કી કરવા માટેની સૌથી વધુ ચોક્કસ પદ્ધતિ ગ્લોબલ પોઝિશનિંગ સિસ્ટમ (Global Positioning

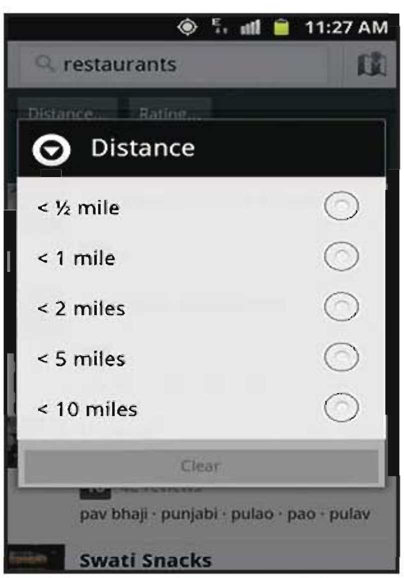
System - GPS) છે. તે સમગ્ર વિશ્વમાં વ્યાપ્ત સેટેલાઇટ ટ્રેકિંગ પદ્ધતિ પર આધારિત છે, જ્યાં પૃથ્વીની આસપાસ પરિભ્રમણ કરતા ત્રણ ઉપગ્રહોના સમૂહ દ્વારા GPS સિગ્નલ દ્વારા ઉત્પન્ન કરેલાં સિગ્નલના છેદબિંદુનો ઉપયોગ કરવામાં આવે છે. તે 500 મીટરના ઘેરાવામાં ઉપકરણનું સ્થાન નક્કી કરે છે. આને ત્રિકોણીય (Triangulation) તરીકે ઓળખવામાં આવે છે. GPS સિગ્નલ અલ્પ કે અવરુદ્ધ હોય, તો મોબાઇલ ઉપકરણ સેલ ટાવર અને Wi-Fi હોટ સ્પોટ તરફથી મળતા સિગ્નલનો ઉપયોગ કરી શકે છે. આ સિગ્નલ સ્વયં સ્થાનને પ્રસારિત કરી શકતા નથી, પરંતુ સ્માર્ટફોન સંસ્થાઓ પોતાના ડેટાબેઝમાં સંગૃહીત સ્રોતનો ઉપયોગ કરી આ સ્થાનની જાણકારી આપે છે.

GPS દ્વારા મોબાઇલ સાધનનો ઉપયોગ કરી ઉપયોગકર્તાના સ્થાનની જાણકારી મેળવવા માટેનો વિનિયોગ આકૃતિ 5.11માં દર્શાવ્યો છે. (**નોંધ :** આ માટે સાધન પર GPSનું સમર્થન હોવું જરૂરી છે.) આકૃતિ 5.11માં મોબાઇલ સાધન પર દર્શાવવામાં આવેલું વિજ્ઞાપન પણ સ્થાનઆધારિત છે. આકૃતિમાં ઉપયોગકર્તાનું વર્તમાન મોબાઇલ સ્થાન સી.જી.રોડ, અમદાવાદ છે. જો ઉપયોગકર્તા તેના સ્થાનથી નજીક આવેલા ભોજનાલય શોધવા ઇચ્છતો હોય, તો તે આકૃતિ 5.12માં દર્શાવ્યા મુજબનો વિકલ્પ પસંદ કરશે. આમ કરવાથી તેના સ્થાનની નજીક આવેલ ભોજનાલયની યાદી દર્શાવવામાં આવશે. ઉપયોગકર્તા આ શોધ માટે નિશ્ચિત અંતરની સ્પષ્ટતા પણ કરી શકે છે, જે આકૃતિ 5.13માં દર્શાવ્યું છે. આમ કરવાથી આકૃતિ 5.14માં દર્શાવ્યા મુજબ શોધના પરિણામને વધુ ફિલ્ટર કરીને રજૂ કરી શકાય છે.

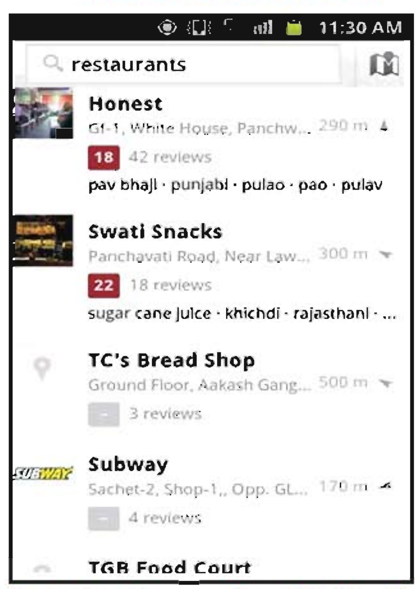
આકૃતિ 5.11 : સ્થાન આધારિત વિનિયોગ



આકૃતિ 5.12 : ઉપયોગકર્તાના સ્થાનને અનુરૂપ નજીકના ભોજનાલયની શોધ



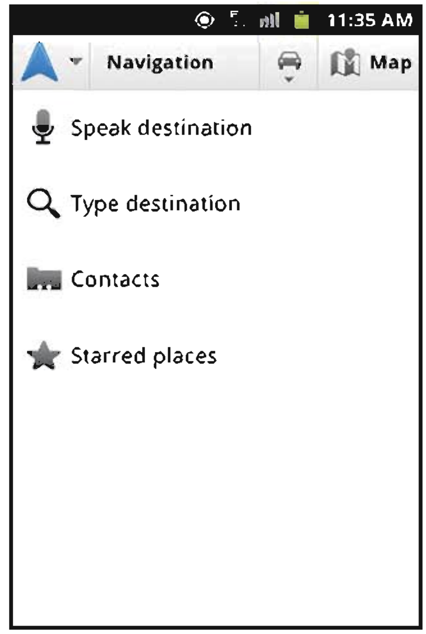
આકૃતિ 5.13 : શોધપરિણામને ચાળવું



આકૃતિ 5.14 : ફિલ્ટર કર્યા પછીનું શોધપરિણામ

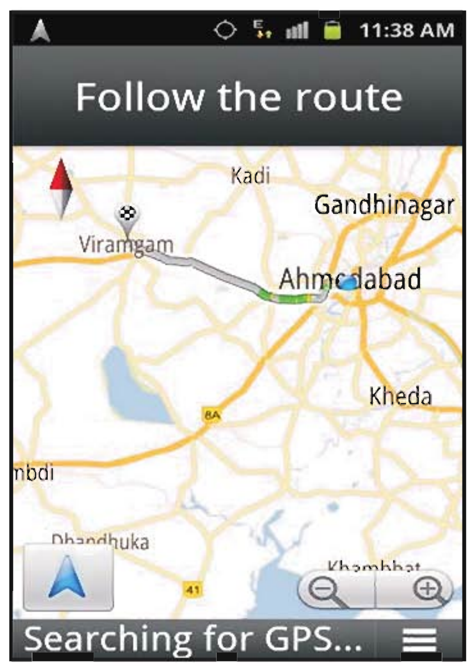
હવે, સ્થાન આધારિત સેવાઓના એક અન્ય ઉદાહરણની ચર્ચા કરીએ. જેમાં આપણે ગંતવ્યસ્થાન સુધીનો માર્ગ શોધવા માટે નકશાનો ઉપયોગ કરી શકીએ છીએ. ઉદાહરણ તરીકે, ઉપયોગકર્તાને ગંતવ્યસ્થાન સુધી પહોંચવું છે પરંતુ તેનો માર્ગ તે જાણતો નથી. આકૃતિ 5.15માં દર્શાવ્યા મુજબ નકશાના ઉપયોગ દ્વારા સ્થાન આધારિત સેવાઓ ઉપયોગકર્તાનું વર્તમાન સ્થાન શોધી શકે છે, જેને ભૂરા રંગના નિર્દેશક દ્વારા દર્શાવવામાં આવે છે. હવે આકૃતિ 5.16માં દર્શાવ્યા મુજબ આપણે ગંતવ્યસ્થાનનું નામ ટાઇપ કરીને કે બોલીને તેનો નિર્દેશ કરી શકીએ છીએ.

આકૃતિ 5.15 : ઉપયોગકર્તાનું સ્થાન શોધવા માટે નકશાનો ઉપયોગ



આકૃતિ 5.16 : ગંતવ્યસ્થાન ઉમેરવા માટેના વિકલ્પો

આકૃતિ 5.17 : ગંતવ્યસ્થાન નક્કી કરવું



આકૃતિ 5.18 : ગંતવ્યસ્થાન સુધીનો માર્ગ

આકૃતિ 5.17માં દર્શાવ્યા મુજબ ગંતવ્યસ્થાન તરીકે 'વીરમગામ' પસંદ કરવામાં આવ્યું છે. આકૃતિ 5.18માં આપેલ નકશો સ્રોતસ્થાનથી ગંતવ્યસ્થાન સુધીનો માર્ગ દર્શાવે છે.

આકૃતિ 5.19 : ગંતવ્યસ્થાન સુધી પહોંચવા માટેનું અંતર અને સંભવિત સમય

આકૃતિ 5.20 : વાહનચાલકને માર્ગદર્શન

આકૃતિ 5.19માં ગંતવ્યસ્થાને પહોંચવા માટેનું અંતર અને અંદાજિત સમય પણ દર્શાવવામાં આવ્યા છે. આકૃતિ 5.20માં દર્શાવ્યા મુજબ ગંતવ્યસ્થાન સુધી પહોંચવા માટેની મદદ પણ મેળવી શકાય છે સ્થાન આધારિત સેવાઓ ગ્રાહકને જરૂરિયાત અનુસાર વધુ ઝડપી અને નિશ્ચિત સેવાઓ માટે અનુકૂલન અને અવસર પૂરાં પાડે છે. સ્થાન આધારિત કેટલીક સેવાઓનાં ઉદાહરણ નીચે દર્શાવ્યાં છે :

- માહિતી અથવા નિર્દેશન-સેવાઓ (Information or directory services) : ક્રિયાશીલ 'યલોપેજ' ઉપયોગકર્તાને આપોઆપ નજીકના ભોજનાલય, પાર્કિંગ-સુવિધા, યાતાયાતની માહિતી વગેરેથી માહિતગાર કરે છે.
- અનુસંધાન-સેવા (Tracking services) : અકસ્માતની શોધખોળ, મિત્રના ભૌગોલિક સ્થાનની શોધ, ખોવાયેલી મોટરકારની શોધ, વાલી દ્વારા પોતાનાં બાળકોની શોધ વગેરે.
- સંકટકાળ-સેવાઓ (Emergency services) : સંકટ સમયે આરોગ્યલક્ષી અને ઍમ્બ્યુલન્સ સેવા, શોધ અને બચાવ-કામગીરી, માર્ગ અકસ્માતોમાં સહાય, પોલીસ, સંરક્ષણ અને અગ્નિશામક સહાય.
- વિજ્ઞાપન દ્વારા પ્રોત્સાહન (Advertising promotion) : લક્ષ્યવેધી વિજ્ઞાપનો, પ્રોત્સાહક સંદેશાઓ, દુકાનમાં ગ્રાહકની ઓળખાણ.
- નકશા (Mapping) : વિશિષ્ટ ભૌગોલિક સ્થાનના નકશાઓની રચના.
- નેવિગેશન (Navigation) : એક સ્થળથી અન્ય સ્થળ સુધીનું દિશાસૂચન

ઈ-કૉમર્સ અને એમ-કૉમર્સમાં સુરક્ષાની સમસ્યાઓ (Security Issues in E-commerce & M-commerce)

ઇન્ટરનેટ એવું સાર્વજનિક નેટવર્ક છે, જેમાં હજારોની સંખ્યામાં અંગત કમ્પ્યૂટર નેટવર્ક પરસ્પર સંલગ્ન હોય છે. આ અંગત નેટવર્ક ક્યારેક સાર્વજનિક નેટવર્ક પર સુરક્ષારહિત બની જતું હોય છે. વ્યવસાય માટે ઇન્ટરનેટ સારી તકો પૂરી પાડે છે, પણ અનુકૂળતાઓ સાથે હંમેશાં નવાં જોખમો આવતાં હોય છે. ઇન્ટરનેટ પર સ્થાનાંતરિત થતી મહત્ત્વની વિગતો કે માહિતીનો દુરુપયોગ થઈ શકે તેની ઉચાપત થઈ શકે કે તેને વિકૃત કે નષ્ટ કરી દેવામાં પણ આવે. ઉદાહરણ તરીકે, ઈ-કૉમર્સ વેબસાઇટ દ્વારા ઑનલાઇન ખરીદી કરતી વખતે ગ્રાહકે ક્રેડિટકાર્ડનો નંબર અને અંગત માહિતી પૂરી પાડવી પડે છે. આ માહિતી વ્યાપારીના સર્વર પર મોકલવામાં આવે છે. વ્યાપારીનું સર્વર આ માહિતી પેમેન્ટ ગેટ-વે દ્વારા કાર્ડ આપનાર બૅન્કને ખરાઈ માટે મોકલી આપે છે. આ તમામ વ્યવહારો ઇન્ટરનેટ જેવાં સાર્વજનિક નેટવર્ક પર કરવામાં આવે છે. આ પ્રક્રિયા દરમિયાન અનધિકૃત ઉપયોગકર્તા ક્રેડિટકાર્ડનો નંબર જાણી લઈને ભવિષ્યમાં તેનો

દુરુપયોગ કરી શકે છે. તદુપરાંત, ઑર્ડરની માહિતીને અધવચ્ચે બદલી નાંખવામાં આવે તેવી પણ સંભાવના છે. ગ્રાહકે 10 વસ્તુઓનો ઑર્ડર આપ્યો હોય, પરંતુ કોઈક કારણોસર વિક્રેતાને 100 વસ્તુઓનો ઑર્ડર મળે, તો તે 100 વસ્તુઓની કિંમત ચૂકવવા માંગણી કરશે. કોઈક ઘૂસણખોર દુનિયાના કોઈ પણ ખૂણેથી પોતાના કમ્પ્યૂટર દ્વારા માહિતી ચોરી લઈ શકે છે અને તેમાં ફેરફાર પણ કરી શકે છે. તે પોતાની અંગત ઓળખ છુપાવી નવા પ્રોગ્રામની રચના દ્વારા દૂરસ્થ કમ્પ્યૂટર પર ભંગાણ સર્જી શકે છે અને અતિખરાબ સ્થિતિમાં કમ્પ્યૂટરને સ્થગિત પણ કરી શકે છે.

ઇ-કૉમર્સ/એમ-કૉમર્સની સાઇટ પર ગ્રાહકની અંગત માહિતી, તેમની બૅન્ક અંગેની માહિતી અને તેના જેવી અનેક મહત્ત્વપૂર્ણ માહિતીઓ મૂકવામાં આવે છે, માટે સંસ્થાઓએ આવી તમામ પ્રકારની છેતરપિંડીથી સચેત રહેવું જરૂરી છે. ઇ-કૉમર્સમાં ચૂકવણી માટે ઑનલાઇન બૅંકિંગ, ઇલેક્ટ્રૉનિક વ્યવહાર, ક્રેડિટકાર્ડ અને ડેબિટકાર્ડનો ઉપયોગ થતો હોવાથી, ઇ-કૉમર્સ/એમ-કૉમર્સ વેબસાઇટમાં સુરક્ષા અંગેની વધુ સમસ્યા જોવા મળે છે. અન્ય વેબસાઇટ કરતા આ પ્રકારની વેબસાઇટ છેતરપિંડીનો વધુ ભોગ બને છે. આમ, ઇન્ટરનેટ પર વિગતોની સુરક્ષાનું વિશેષ મહત્ત્વ છે. ઇ-કૉમર્સ/એમ-કૉમર્સની સુરક્ષા નીચે દર્શાવેલ ચાર અગત્યના મુદ્દા અનુસાર હોવી જરૂરી છે :

- **ગુપ્તતા (Confidentiality)**

  અનધિકૃત ઉપયોગકર્તા વાંચી ન શકે તે માટે અહીં માહિતીની ગુપ્તતાનો નિર્દેશ કરવામાં આવ્યો છે, જે સાંકેતીકરણ (cryptography) દ્વારા પ્રાપ્ત કરી શકાય છે. તેમાં તમામ સંદેશાઓ સાંકેતિક લખાણમાં પ્રસારિત કરવામાં આવે છે. અને યોગ્ય કી (key)નો ઉપયોગ કરી સંદેશને મૂળ રૂપમાં ફેરવ્યા પછી માત્ર પ્રાપ્તકર્તા જ તે વાંચી શકે છે. આ પ્રક્રિયા અંગત વિગતોને બાહ્ય આક્રમણથી સુરક્ષિત રાખે છે અને ખાતરી આપે છે કે સંદેશ ગંતવ્યસ્થાને પહોંચતા સુધીમાં જાહેર થયેલ નથી કે અન્ય કોઈને તેની જાણ થયેલ નથી. ક્રેડિટકાર્ડના નંબર જેવી ખાનગી વિગતોની સુરક્ષામાં આ પ્રક્રિયા મદદરૂપ બને છે.

- **અખંડિતતા (Integrity)**

  તે ખાતરી આપે છે કે પ્રસારણ દરમિયાન માહિતી આકસ્મિક રીતે કે દ્વેષપૂર્વક બદલવામાં આવી નથી કે તેની સાથે કોઈ ફેરફાર કરવામાં આવ્યો નથી. પ્રાપ્તકર્તા એ જ સંદેશ મેળવે છે, જે પ્રેષક દ્વારા મોકલવામાં આવ્યો હતો. જો પ્રસારણ દરમિયાન સંદેશ બદલાયો હોય, તો તે શોધી શકાવો જોઈએ. આમ કરવાથી ઑર્ડરના જથ્થામાં થતા ફેરફારની ક્ષતિ દૂર કરી શકાય છે, જે પછીથી ઉદ્‌ભવતી ચુકવણીની ક્ષતિને પણ રોકે છે.

- **અધિકૃતતા (Authorization)**

  તે ખાતરી આપે છે કે માત્ર અધિકૃત ઉપયોગકર્તાને સિસ્ટમના ઉપયોગની અનુમતિ આપવામાં આવી છે. અધિકૃતતા મેળવવા માટે લૉગ-ઇન અને પાસવર્ડ એક માર્ગ છે.

- **અસ્વીકાર (Non-repudiation)**

  સંદેશ મોકલનાર વ્યક્તિ સંદેશ મોકલ્યાનો ઇનકાર કરી શકે નહીં, તેની અહીં ખાતરી આપવામાં આવે છે. મોકલનારે ખરેખર સંદેશો મોકલ્યો હોય, તો તેના અસ્વીકારને અટકાવે છે. ઉદાહરણ તરીકે, જો ગ્રાહક ખરીદ-ઑર્ડર મોકલ્યા છતાં કોઈ કારણસર તેનો ઇનકાર કરે તો સાબિત થઈ શકે છે કે ગ્રાહકે તે સંદેશ મોકલ્યો છે. સામાન્ય રીતે ડિજિટલ સહી (digital signatures) અથવા વિશ્વસનીય ત્રાહિત પક્ષ (Trusted Third Party - TTP) દ્વારા તેને પ્રમાણિત કરવામાં આવે છે.

ઇન્ટરનેટ સુરક્ષાનાં ભયસ્થાનો (Internet Security Threats)

ઇન્ટરનેટ પર સામાન્ય રીતે જે ધમકીઓનો સામનો કરવો પડે છે, તે નીચે મુજબ છે :

- **દૂષિત કોડ (Malicious code)**

  આપણા કમ્પ્યૂટર કે સિસ્ટમને નુકસાન પહોંચાડતા કોડને દૂષિત કોડ તરીકે ઓળખવામાં આવે છે. દૂષિત કોડ જાતે જ સક્રિય બને છે અથવા ઉપયોગકર્તા કોઈ સ્થાને ક્લિક કરે અથવા તો ઇ-મેઇલનું જોડાણ (attachment) ખોલે, ત્યારે કોઈ વાઇરસની જેમ તે કાર્ય કરે છે અને માહિતી ચોરી લે છે અથવા તો ફાઇલને નષ્ટ કરી વધુ નુકસાન પણ પહોંચાડી શકે છે.

- **સ્નિફિંગ (Sniffing)**

  સ્નિફર એવો પ્રોગ્રામ છે જે ઇન્ટરનેટનો ઉપયોગ કરી કમ્પ્યૂટર કે રાઉટર દ્વારા પ્રેષકથી પ્રાપ્તકર્તા સુધી પ્રસારિત થતી માહિતીની નોંધ કરે છે. સ્નિફર-પ્રોગ્રામનો ઉપયોગ ટેલીફોન વાયરને જોડીને વાર્તાલાપની નોંધ કરવા સમાન છે. સ્નિફર-પ્રોગ્રામ ઇ-મેઇલ સંદેશ, ઉપયોગકર્તાનું લોગ-ઇન, પાસવર્ડ અને ક્રેડિટકાર્ડ નંબર વાંચી શકે છે.

- **સેવાના અસ્વીકાર સ્વરૂપે આક્રમણ (Denial of Service Attack)**

  આ એક એવું આક્રમણ છે જેના દ્વારા મશીન કે નેટવર્ક બંધ થઈ જાય છે અને તેના ઉપયોગકર્તાઓ માટે તે નિરૂપયોગી બની જાય છે. હુમલાખોર ઉપયોગકર્તાના કમ્પ્યૂટર અને તેના નેટવર્કજોડાણને કે સાઇટને લક્ષ્ય બનાવી ઉપયોગકર્તાને ઇ-મેઇલ, વેબસાઇટ અને અસરગ્રસ્ત કમ્પ્યૂટર પર રહેલી બૅંકિંગ કે અન્ય સેવાઓ પૂરી પાડતાં ઑનલાઇન ખાતાંઓનો ઉપયોગ કરતાં અટકાવે છે. ઉપયોગકર્તાને હજારો સંદેશ મોકલવામાં આવે છે, જે નેટવર્ક પર યાતાયાતની મુશ્કેલીઓ ઊભી કરે છે.

- **સાયબર જંગાલિયાત (Cyber Vandalism)**

  સાયબર જંગાલિયાત એ હયાત વેબસાઇટ પર કરવામાં આવતી ઇલેક્ટ્રૉનિક ભાંગફોડિયા પ્રવૃત્તિ છે. આક્રમણકાર વેબસાઇટની મૂળભૂત વિગતોને પોતાની વિગતો સાથે બદલી નાંખે છે. તે અખંડિતતા ઉલ્લંઘનનું ઉદાહરણ છે. તે મિલકતનો નાશ અથવા કોઈની તસવીર પર કરવામાં આવતા દૂષિત ફેરફારોનું ઇલેક્ટ્રૉનિક સ્વરૂપ છે. આજકાલ સાયબર જંગલિયાતના અનેક કિસ્સા બને છે, જેમાં વ્યાવસાયિક વિગતોને અરુચિકર સામગ્રી સાથે બદલવામાં આવે છે.

- **છેતરપિંડી (Spoofing)**

  છેતરપિંડી અથવા છદ્મવેશીપણું (masquerading) એટલે અન્ય વ્યક્તિ હોવાનો સ્વાંગ ભરવો અથવા અન્યની વેબસાઇટનો પોતાની અધિકૃત વેબસાઇટ ગણાવવી. આ તકનિકમાં હુમલાખોર ત્રાહિત વ્યક્તિ કે વેબસાઇટની ઓળખ ધારણ કરી વ્યવહાર કરે છે. ઉદાહરણ તરીકે, હુમલાખોર www.gswan.co.in નામની નકલી વેબસાઇટ બનાવી શકે છે અને તેના IP સરનામાને મૂળ વેબસાઇટના IP સરનામા સાથે બદલી નાખે છે. આમ, મૂળ સાઇટની મુલાકાત લેવા ઇચ્છતા તમામ ઉપયોગકર્તાઓ નકલી વેબસાઇટ તરફ જાય છે.

સુરક્ષાના ઉપાયો (Security Measures)

સુરક્ષા સામેની જુદી-જુદી ધમકીઓને પહોંચી વળવા ઘણા ઉપાયો યોજવામાં આવે છે. તેમાંના કેટલાકની ચર્ચા કરીએ.

- **ઍન્ટિવાઇરસ સૉફ્ટવેર (Antivirus Software)**

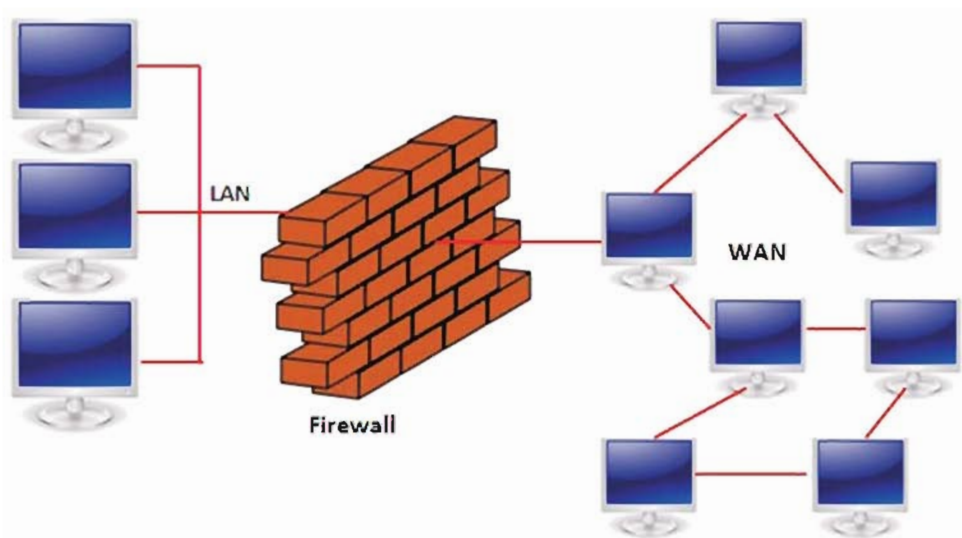
  ઍન્ટિવાઇરસ સૉફ્ટવેર એ એવો કમ્પ્યૂટર-પ્રોગ્રામ છે, જે અસરગ્રસ્ત સિસ્ટમમાં આવેલ વાઇરસ, વૉર્મ અને ટ્રોજન હોર્સ જેવા દૂષિત કોડને શોધે છે, અટકાવે છે અને દૂર કરવા જરૂરી પગલાં લે છે. કમ્પ્યૂટરની સુરક્ષા માટે સારું ઍન્ટિવાઇરસ સૉફ્ટવેર જરૂરી છે. ઍન્ટિવાઇરસ સૉફ્ટવેર વગરની સિસ્ટમ ઇન્ટરનેટ પર બહુ અલ્પ સમયમાં દૂષિત કોડનું સરળ લક્ષ્ય બની જાય છે. દૂષિત કોડને કારણે તેને અનેક પ્રકારની મુશ્કેલીઓ કે નુકસાનમાંથી પસાર થવું પડે છે. દૂષિત કોડનો આ ચેપ વિચિત્ર ધ્વનિ કે પોપ-અપ વિન્ડો ઉત્પન્ન કરવી અને અન્ય ત્રાસજન્ય કાર્યો કરવા જેવા સામાન્ય પણ હોઈ શકે છે. તે ફાઇલોને નષ્ટ કરી શકે છે, સિસ્ટમને ધીમી પાડી દે છે અથવા હાર્ડવેરને નુકસાન કરી આખી કમ્પ્યૂટર સિસ્ટમનો નાશ પણ કરી શકે છે. એક વાર વાઇરસથી સિસ્ટમ અસરગ્રસ્ત બને પછી સિસ્ટમ સાથે જોડાયેલા અન્ય પ્રોગ્રામ અને ફાઇલોમાં પણ તે પ્રસરે છે. વાઇરસ માત્ર પોતાની પ્રતિકૃતિ ઉત્પન્ન કરવાનું કાર્ય કરતા નથી, પરંતુ ઉપયોગકર્તા દ્વારા સંપર્કસૂચિમાં રહેલી અન્ય વ્યક્તિઓની વિગતોને પણ તે પ્રભાવિત

કરે છે. સિસ્ટમ પર હુમલો થવાનો સૌથી સરળ માર્ગ ઇ-મેઇલ સાથે મોકલવામાં આવતા ઈ-મેઇલનાં દૂષિત જોડાણો (attachments)નો છે. આ જોડાણો ઇ-મેઇલ સાથે જોડી શકાય તેવા ચિત્રો, વીડિયો, ધ્વનિ કે અન્ય પ્રકારનાં હોઈ શકે છે. ઇન્ટરનેટ પરથી કરવામાં આવતા ડાઉનલોડ દ્વારા પણ વાઇરસનો ચેપ ફેલાઈ શકે છે.

સિસ્ટમને થતું નુકસાન રોકવા માટે ઍન્ટિવાઇરસ સૉફ્ટવેર ડાઉનલોડ થયેલ વિગતો સહિત આખી કમ્પ્યૂટર સિસ્ટમને વાઇરસની હાજરી શોધવા માટે તપાસે છે. આજકાલ, ઇન્ટરનેટના ઉપયોગ માટે મોબાઇલ ઉપકરણો પણ પ્રચલિત થવાને કારણે તેના માટે પણ ઍન્ટિવાઇરસ સૉફ્ટવેર ઉપલબ્ધ છે. બજારમાં અનેક પ્રકારનાં ઍન્ટિવાઇરસ સૉફ્ટવેર ઉપલબ્ધ છે. ઍન્ટિવાઇરસ સૉફ્ટવેર કમ્પ્યૂટરમાં સ્થાપિત કરવું અને તેને નિયમિત રીતે અદ્યતન (update) બનાવતા રહેવું અત્યંત જરૂરી છે.

- **ફાયરવૉલ (Firewall)**

પોતાની વેબસાઇટ ધરાવતી સંસ્થાઓએ સંસ્થાની અંદર અને બહાર બંનેની નેટવર્ક-સેવાઓ પર નિયંત્રણ રાખવું પડે છે. સંસ્થાના નેટવર્ક અને બહારની દુનિયા વચ્ચે સામાન્ય રીતે ઉપયોગમાં લેવાતી નેટવર્કની સુરક્ષા-સીમાને ફાયરવૉલ કહે છે. આકૃતિ 5.21માં દર્શાવ્યા મુજબ ફાયરવૉલ એ સંસ્થાના સ્થાનિક નેટવર્ક અને બહારની દુનિયા વચ્ચેના યાતાયાત પર નિરીક્ષણ અને નિયંત્રણ રાખવા માટે મૂકવામાં આવેલું એક સાધન (કમ્પ્યૂટર અથવા રાઉટર) છે. અનધિકૃત વ્યક્તિઓને સંસ્થાની ઇ-કૉમર્સ વેબસાઇટના માળખાથી દૂર રાખવા એ ફાયરવૉલનો પ્રાથમિક હેતુ છે. તે ખાતરી આપે છે કે સંસ્થાની અતિ મહત્ત્વની વિગતો સલામત રીતે સચવાયેલી છે અને અનધિકૃત વ્યક્તિ તેનો દુરુપયોગ કરી શકશે નહિ.



આકૃતિ 5.21 : સ્થાનિક અને સાર્વજનિક નેટવર્ક વચ્ચે આપેલી ફાયરવૉલ

ફાયરવૉલ સ્થાનિક નેટવર્કને નીચેનાં જોખમો સામે સુરક્ષા આપે છે :

- ઇ-મેઇલ સેવા જે કેટલીક વાર મુશ્કેલી ઊભી કરે છે.
- સ્થાનિક નેટવર્કમાં અનપેક્ષિત ચિત્રો, વીડિયો જેવી વિગતોનો પ્રવેશ રોકવો.
- સ્થાનિક નેટવર્કનો અનધિકૃત વ્યક્તિ દ્વારા ઉપયોગ અટકાવવો.
- સંસ્થાના નેટવર્કમાંથી બહાર જતી અનધિકૃત વિગતો કે માહિતી રોકવી.
- બહારની દુનિયામાંથી સ્થાનિક નેટવર્કમાં થતા યાતાયાતને અટકાવવો.
- કોઈ પણ પ્રકારના નેટવર્ક આક્રમણ સામે સુરક્ષા આપવી.

• ડિજિટલ સર્ટિફિકેટ (Digital Certificate)

ઈલેક્ટ્રોનિક વ્યવહારમાં આપણી ઓળખાણ સાબિત કરવા માટે ડિજિટલ સર્ટિફિકેટ કે ડિજિટલ IDનો ઉપયોગ કરવામાં આવે છે. જેમ વાસ્તવિક દુનિયામાં આપણી ઓળખ પૂરી પાડવા ડ્રાઇવિંગ લાઇસન્સ કે પાસપોર્ટની જરૂર હોય છે. તેવી જ રીતે ડિજિટલ સર્ટિફિકેટની મદદથી વ્યાવસાયિક સંસ્થાઓ ઑનલાઇન સેવાઓ અને મિત્રોને ખાતરી કરાવી શકાય છે કે આપણી પાસેથી તેમણે મેળવેલી માહિતી અધિકૃત છે. ધારકની ઓળખ પ્રસ્થાપિત કરવા માટે વિશ્વસનીય ત્રાહિત પક્ષ (trusted third party) દ્વારા ડિજિટલ સર્ટિફિકેટ આપવામાં આવે છે. સર્ટિફિકેટ આપનાર ત્રાહિત પક્ષને સર્ટિફિકેશન ઑથોરિટી (Certification Authority - CA) કહે છે. ડિજિટલ સર્ટિફિકેટમાં ધારકનું નામ, ક્રમ, સમાપ્તિ તારીખ, સંદેશને સાંકેતિક બનાવવા માટે ડિજિટલ સર્ટિફિકેટ ધારકની પબ્લિક કી, ધારકની સહી તથા સર્ટિફિકેટ આપનાર અધિકૃત પક્ષની સહી હોય છે, જેના દ્વારા સર્ટિફિકેટની સત્યતા સાબિત થઈ શકે છે.

• સાંકેતીકરણ (Cryptography)

માહિતીને અવાચ્ય રૂપમાં ફેરવી સુરક્ષિત રાખવાની કળાને સાંકેતીકરણ (Cryptography) કહે છે. સાંકેતીકરણની પ્રક્રિયામાં સામાન્ય લખાણ કે સાદા લખાણ (plain text)ને સાંકેતિક અલ્ગોરિધમના ઉપયોગ દ્વારા ખાનગી કે ગુપ્ત લખાણ (cipher text)માં રૂપાંતરિત કરી અવાચ્ય બનાવવામાં આવે છે. સંદેશના સાંકેતીકરણ કે બિનસાંકેતીકરણ માટે ગુપ્ત કી (secret key)નો ઉપયોગ કરવામાં આવે છે. તે લખાણને અદૃશ્ય બનાવતું નથી, પરંતુ લખાણને અન્ય એવા લખાણમાં પરિવર્તિત કરે છે, જે અર્થપૂર્ણ હોતું નથી. તેનો ઉદ્દેશ ઇન્ટરનેટ પર સંદેશના અધિકૃત પ્રાપ્તકર્તા સિવાયની અન્ય વ્યક્તિઓની માહિતીને અવાચ્ય રાખી ગુપ્તતાની જાળવણીનો છે. ઇન્ટરનેટ કે નેટવર્ક પર પ્રસારિત કરતાં પહેલાં સંદેશનું સાંકેતીકરણ (encryption) કરવામાં આવે છે. સાંકેતીકરણ દ્વારા પરિવર્તિત થયેલો ગુપ્ત સંદેશ પ્રાપ્તકર્તાને મળે, ત્યારે તેને મૂળ સ્વરૂપમાં લાવવો જરૂરી છે. આ માટે સાંકેતીકરણ (encryption)થી વિરુદ્ધ એવી બિનસાંકેતીકરણ (decryption)ની પ્રક્રિયાનો અમલ કરવામાં આવે છે. આ પ્રક્રિયા અન્વયે ગુપ્ત લખાણને પુનઃ સાદા લખાણમાં રૂપાંતરિત કરવામાં આવે છે. સાંકેતીકરણ માટેના અનેક અલ્ગોરિધમ બજારમાં ઉપલબ્ધ છે.

છેલ્લાં કેટલાંક વર્ષોમાં એવા અનેક કિસ્સા બન્યા છે, જેમાં પ્રસારિત થતી વિગતોને આંતરવામાં આવી હોય. ઇન્ટરનેટ, ઇ-કૉમર્સ, મોબાઇલ ટેલિફોન, બ્લ્યુટૂથ સાધનો, બૅન્કનાં ઑટોમેટિક ટેલર મશીન (Automatic Teller Machines - ATM) જેવા નેટવર્કમાં પ્રસારણ દરમિયાન વિગતોની સુરક્ષા માટે સાંકેતીકરણનો ઉપયોગ કરવામાં આવે છે. ધારો કે, મિત્રને "HOW ARE YOU?" સંદેશ મોકલવાનો છે. સંદેશને સુરક્ષિત બનાવવા માટે સાંકેતિક ભાષામાં રૂપાંતરિત કરવો પડશે. સાંકેતીકરણનો ઉપયોગ કરી ગુપ્ત સંદેશ મિત્રને મોકલવામાં આવશે. આ ઉદાહરણમાં સાંકેતીકરણની પ્રક્રિયા દ્વારા દરેક અક્ષરને તેના પછીના અક્ષર સાથે બદલીને રજૂ કરવામાં આવ્યો છે. આનો અર્થ એ થયો કે, "A" એ "B" બનશે, "B" એ "C" બનશે અને તેવી રીતે તમામ અક્ષરોને બદલવામાં આવશે. તમે મિત્રને એ પણ જાણ કરી હશે કે આ કોડને સાદા લખાણમાં ફેરવવા માટે "1થી ખસેડો" (shift by 1). આકૃતિ 5.22માં દર્શાવ્યા મુજબ સંદેશને ગુપ્ત ભાષામાં ફેરવવામાં આવશે.

H	O	W	A	R	E	Y	O	U	← Plain text
I	P	X	B	S	F	Z	P	V	← Cipher text

આકૃતિ 5.22 : કીની મદદથી સાંકેતીકરણ

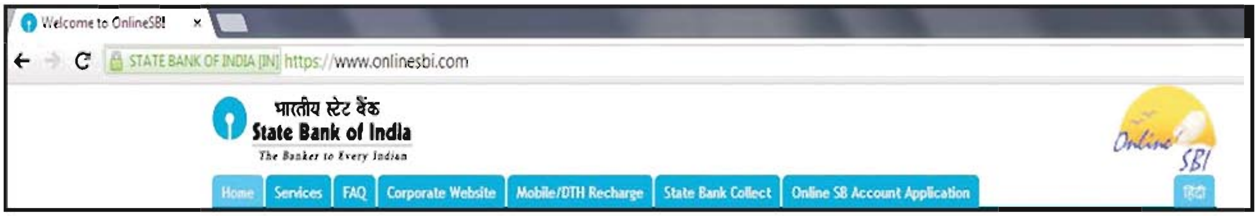
જ્યારે તમારો મિત્ર સંદેશ મેળવશે ત્યારે તેનું બિનસાંકેતીકરણ કરશે. અન્ય કોઈ પણ વ્યક્તિ પ્રસારણ દરમિયાન સંદેશના નિરર્થક અક્ષરો જ જોઈ શકશે. અહીં રાખવામાં આવેલી કી સરળ છે, પરંતુ સામાન્ય રીતે તે ઘણી લાંબી હોય છે.

ઇ-કૉમર્સની સુરક્ષા માટે વિવિધ પ્રોટોકોલ પણ ઉપયોગમાં લેવામાં આવે છે. તેમાંના એક SSL પ્રોટોકોલને સમજીએ.

• સિક્યોર સૉકેટ લેયર (Secure Socket Layer - SSL)

આજકાલ દરેક ઉપયોગકર્તા જુદી-જુદી જાતની અનેક વિગતો ઇ-મેઇલ દ્વારા ક્રેડિટકાર્ડ માટે મોકલતો હોય છે. ઉપયોગકર્તા

ઇચ્છે છે કે સાર્વજનિક નેટવર્ક પર પ્રસારિત થતી વખતે આ વિગતો સુરક્ષિત રહે. ઇન્ટરનેટ પરના વેબ-વ્યવહારોની સુરક્ષા માટે SSL પ્રોટોકોલનો ઉપયોગ કરવામાં આવે છે. તેની રચના નેટસ્કેપ દ્વારા કરવામાં આવી હતી. ઇ-કૉમર્સના વ્યવહાર કરી માહિતીનું સુરક્ષિત પ્રસારણ કરવામાં આવે છે. તે ડિજિટલ સર્ટિફિકેટ દ્વારા વ્યાપારી કે દુકાનદારની અધિકૃતતા પ્રમાણિત કરે છે, જેથી ગ્રાહકને માન્ય માલિક સાથેના વ્યવહારની ખાતરી થાય છે. સાઇટ સુરક્ષિત છે કે નહિ તે તેમાં આવેલા લોગો દ્વારા જાણી શકાય છે. જો સાઇટને વેરિસાઇન (VeriSign) દ્વારા સુરક્ષિત કરવામાં આવી હોય તો સાઇટની લોગ ઇન સ્ક્રિન પર વેરિસાઇનનો લોગો જોવા મળે છે. તેના પર ક્લિક કરવાથી માલિક અંગેની માહિતી તથા સર્ટિફિકેટની માન્યતા જાણી શકાય છે. આ દર્શાવે છે કે સાઇટ સાથેનો વ્યવહાર સુરક્ષિત છે, વેબસાઇટનો માલિક માન્ય છે તથા તે સાઇટ વેરિસાઇન જેવી સત્તા દ્વારા પ્રમાણિત છે. સુરક્ષાનો અન્ય નિર્દેશ એ છે કે સુરક્ષિત જોડાણ ધરાવતી સાઇટનું સરનામું આકૃતિ 5.23માં દર્શાવ્યા મુજબ http:// ને બદલે https:// થી શરૂ થાય છે.



આકૃતિ 5.23 : https://નું ઉદાહરણ

ઇ-કૉમર્સ અને એમ-કૉમર્સમાં કાયદાકીય પ્રશ્નો (Legal Issues in E-commerce/M-commerce) :

વ્યવસાય માટે, વૈશ્વિક બજારના સંપર્ક માટે અને ઑનલાઇન ખરીદીની સુવિધા માટેનો અવસર ઇ-કૉમર્સ અને એમ-કૉમર્સ પૂરો પાડે છે. તે વ્યાવસાયિક પ્રક્રિયામાં વૃદ્ધિની તકો પૂરી પાડે છે. જોકે, નવા કોઈ વ્યવસાયની જેમ ઇ-કૉમર્સ અને એમ-કૉમર્સમાં પણ કેટલાંક પ્રશ્નો અને જોખમો રહેલાં છે. બંનેમાં ઘણા કાયદાકીય પડકાર પણ રહેલા છે, કારણ કે રાષ્ટ્રીય સીમાને ધ્યાનમાં રાખ્યા વગર તે વૈશ્વિક ઇન્ટરનેટ પર પ્રવૃત્ત બને છે. દરેક દેશના પોતાના કાયદા અને નિયમો હોય છે. અહીં બૌદ્ધિક મિલકતોના અધિકાર (intellectual property rights), સ્વાધિકાર (copyrights), અને ગુપ્તતા (privacy) તથા કેટલીક વાર પક્ષો વચ્ચેની તકરારને લગતા પડકાર જોવા મળે છે. આ પ્રશ્નોના નિરાકરણ માટે કાયદાકીય માળખું જરૂરી છે. ઘણા દેશોએ ઇલેક્ટ્રોનિક કૉમર્સ માટે તેમના કાયદાકીય માળખાની સ્થાપના કરી છે. ભારત સરકારે પણ IT કાયદા હેઠળ IT laws (Information technology laws)ની સ્થાપના કરી છે.

કાયદાકીય નીતિ અને નિયમોના પાલનથી ગ્રાહકને વિશ્વાસ બેસે છે કે તેની અંગત વિગતો ખાનગી રહેશે અને તેનો દુરુપયોગ થશે નહીં. જો તેનો દુરુપયોગ કરવામાં આવ્યો હોય, તો તે માટે જવાબદાર પક્ષોને સજા પણ કરી શકાશે. આ પ્રકારની સુરક્ષા ઇ-કૉમર્સ અને એમ-કૉમર્સ માટે અનિવાર્ય છે, કારણકે તેના દ્વારા ગ્રાહક ક્રેડિટકાર્ડની વિગતો જેવી સંવેદનશીલ માહિતીઓ પૂરી પાડે છે. ઑનલાઇન વ્યવસાય કરતી સંસ્થાઓને તેમના લોગો કે કૉપીરાઇટ જેવી ડિજિટલ માહિતીના દુરુપયોગ સામે કાયદાકીય સમર્થન પૂરું પાડવામાં આવે છે. આ કાયદા અન્વયે બે પક્ષો વચ્ચે થયેલ તકરારનું પણ નિરાકરણ લાવી શકાય છે.

બૌદ્ધિક સંપત્તિની સુરક્ષા (Securing Intellectual Property)

બૌદ્ધિક સંપત્તિમાં પુસ્તકો, સૉફ્ટવેર, સંગીત, વીડિયો, કૉપીરાઇટ, ટ્રેડમાર્ક અને વેબપેજનો સમાવેશ કરવામાં આવે છે. તેને સંબંધિત કેટલાક મુદ્દાઓની ચર્ચા કરીએ.

- **સ્વાધિકાર (Copyright)**

  સર્જકના મૂળભૂત કાર્ય પર તેની અનુમતિ વિના અન્ય વ્યક્તિ અધિકાર ધરાવે કે પોતાના ઉપયોગમાં લે તેની સામે રક્ષણ આપતા અધિકારને સ્વાધિકાર (Copyright) કહે છે. પુસ્તકો, સૉફ્ટવેર-પ્રોગ્રામ અને લખાણને તે લાગુ પાડવામાં આવે છે. સ્વાધિકારનો કાયદો બૌદ્ધિક સંપત્તિને અનેક રીતે સુરક્ષા પૂરી પાડે છે. સ્વાધિકાર સંપત્તિ મુક્તપણે ઉપયોગમાં લઈ શકાતી નથી. ઇ-કૉમર્સ કે એમ-કૉમર્સમાં બૌદ્ધિક સંપત્તિની સુરક્ષાનું કાર્ય ઘણું કઠિન છે. ઉદાહરણ તરીકે, તમે

કોઈ સૉફ્ટવેર ખરીદ્યું હોય, તો તમને તેનો ઉપયોગ કરવાનો અધિકાર છે, પરંતુ તેના વિતરણનો નહીં. વિતરણનો અધિકાર માત્ર સ્વાધિકારધારક (copyright holder) પાસે હોય છે. મોટા ભાગનાં વેબપેજ સ્વાધિકારની સુરક્ષા ધરાવતાં હોય છે. વેબપેજ પરથી વિગતોની નકલ કરવી એ પણ સ્વાધિકારના કાયદાનો ભંગ છે.

- **ટ્રેડમાર્ક (Trademark)**

  આ એક નિર્ધારિત લૉગો, ચિહ્ન, શબ્દ, નિશાની, શબ્દસમૂહ કે ચિત્ર છે, જેનો ઉપયોગ કોઈ વ્યક્તિગત કે સંસ્થા દ્વારા કોઈ ઉત્પાદન કે સેવાને બજારના અન્ય ઉત્પાદન કે સેવાથી અલગ પાડવા માટે કરવામાં આવે છે. ટ્રેડમાર્કને TM, SM અને ® નિશાની દ્વારા દર્શાવવામાં આવે છે.

- **ડોમેઈન નામની તકરાર (Domain Name Disputes)**

  ડોમેઇન નામ માટેની સ્પર્ધા એ એક અન્ય કાયદાકીય મુદ્દો છે. પહેલાંના સમયમાં ડોમેઇન નામ વહેલા તે પહેલાનાં ધોરણે આપવામાં આવતાં હતાં. પણ લોકો દ્વારા ઉપયોગમાં લેવાતાં ન હોય તેવાં નામની નોંધણી કરાવી લેવામાં આવતી. પછીથી આ ડોમેઇન નામ કોઈક સંસ્થાને ઘણી ઊંચી કિંમતે વેચી દેવામાં આવતું. આને સાયબર સ્ક્વૉટિંગ (cyber squatting) તરીકે ઓળખવામાં આવે છે. બીજી સમસ્યા નામ બદલવા માટેની છે. જેમાં કોઈ પ્રચલિત ડોમેઇન નામની જોડણીમાં ઇરાદાપૂર્વક સામાન્ય ફેરફાર કરી નોંધણી કરાવવામાં આવે છે. જેણે સામાન્ય રીતે URL ટાઇપ કરવામાં ટાઇપિંગની ભૂલ કરી હોય તેવા ગ્રાહક ગેરમાર્ગે દોરાઈ શકે છે.

બૌદ્ધિક સંપત્તિની સુરક્ષા (Protecting Intellectual Property)

બૌદ્ધિક સંપત્તિની સુરક્ષા માટે ઘણી નવી સુધારેલી પદ્ધતિઓનો સતત વિકાસ થઈ રહ્યો છે. તેમાંની કેટલીક પદ્ધતિઓની ચર્ચા અહીં કરવામાં આવી છે.

સ્ટેગ્નોગ્રાફી (Steganography)

એક માહિતીમાં અન્ય માહિતી સંતાડવાની ક્રિયાને સ્ટેગ્નોગ્રાફી (Steganography) કહે છે. જો ફાઇલમાં આવેલી માહિતીને સુરક્ષિત બનાવવામાં ન આવે, તો તેને કોઈ પણ દૂષિત હેતુ માટે ઉપયોગમાં લઈ શકાય છે. તેનું કાર્ય કમ્પ્યૂટરમાં રહેલાં પણ ઉપયોગમાં લેવાયાં ન હોય તેવા ચિત્રો, ધ્વનિ કે લખાણ ધરાવતી ફાઇલને અદૃશ્ય માહિતી સાથે બદલવાનું છે. આ અદૃશ્ય માહિતી સાદું લખાણ, ગુપ્ત લખાણ કે ચિત્ર પણ હોઈ શકે છે. સ્ટેગ્નોગ્રાફી માટે વિશિષ્ટ સૉફ્ટવેર જરૂરી છે અને ઇન્ટરનેટ પરથી સરળતાથી ડાઉનલોડ થઈ શકે તેવી ફ્રી-વેર આવૃત્તિઓ પણ ઉપલબ્ધ છે.

ડિજિટલ ઓળખચિહ્ન (Digital Watermarking)

વૉટરમાર્ક એ ડિજિટલ ચિત્ર, ધ્વનિ કે વીડિયોમાં ઉમેરવામાં આવતો ડિજિટલ કોડ છે જે ફાઇલના સ્વાધિકાર (copyright)ની ઓળખ આપી શકે છે. તે માહિતીનો સંપૂર્ણપણે અદૃશ્ય સ્વરૂપે સંગ્રહ કરવાની સુવિધા આપે છે. શરૂઆતના સમયમાં કલાકારો ચિત્રની ઉપર પીંછી દ્વારા સર્જનાત્મક નિશાની કરી સ્વાધિકારનો હક દાવો કરતા. પરંતુ ડિજિટલની દુનિયામાં કલાકારો ચિત્રની અંદર તેમનું નામ છુપાવી ઓળખચિહ્ન (watermark) બનાવી શકે છે. આમ, સર્જકની ઓળખાણ માટે વૉટરમાર્ક અદૃશ્ય રીતે મદદરૂપ બને છે.

આ અભિગમને ડિજિટલ ઑડિયો અને વીડિયો જેવાં અન્ય માધ્યમોમાં પણ ક્રિયાન્વિત કરી શકાય છે. હાલમાં ઇન્ટરનેટ પર ડિજિટલ ઑડિયોનું MP3 સ્વરૂપમાં અનધિકૃત વિતરણ એક મોટી સમસ્યા છે. ડિજિટલ ઓળખચિહ્ન દ્વારા ઑડિયોના વિતરણને નિયંત્રિત બનાવી શકાય છે. અને અસરકારક રીતે સ્વાધિકાર સુરક્ષા પૂરી પાડી શકાય છે. ડેટાસુરક્ષાના ક્ષેત્રમાં ઓળખચિહ્નનો ઉપયોગ પ્રમાણપત્ર અને અધિકૃતતા માટે પણ કરી શકાય છે. ઉદાહરણ તરીકે, વ્યક્તિના ફોટો-ઓળખપત્ર પર 123456 સંખ્યા લખીને અને તેને ડિજિટલ વૉટરમાર્ક ઓળખચિહ્ન રૂપે અદૃશ્ય રાખીને ફોટો ઓળખને સુરક્ષિત બનાવી શકાય છે. આમ, તસવીરમાં થયેલ ફેરફારને સરળતાથી શોધી શકાય છે.

ડિજિટલ ઓળખચિહ્ન દ્વારા દસ્તાવેજમાં માહિતીને લિંક પણ આપી શકાય છે. ઉદાહરણ તરીકે, સામાન્ય રીતે પાસપોર્ટધારક વ્યક્તિનું નામ સ્પષ્ટ શબ્દોમાં લખાયેલું હોય છે. પરંતુ પાસપોર્ટની તસવીર સાથેના નામને ડિજિટલ ઓળખચિહ્નની મદદથી અદૃશ્ય બનાવી શકાય છે. જો કોઈએ પાસપોર્ટમાં ચેડાં કરી ફોટો બદલી નાખ્યો હોય, તો તે જાણવા માટે પાસપોર્ટને

સ્કેન કરી ફોટા સાથેના અદૃશ્ય નામની ચકાસણી પણ કરી શકાય છે. GIMP જેવા ફોટો-એડિટર પ્રોગ્રામની મદદથી ચિત્રમાં દૃશ્યમાન વૉટરમાર્ક ઉમેરી શકાય છે.

ઇ-કૉમર્સ/એમ-કૉમર્સમાં ચુકવણી (Payment in E-commerce/M-commerce)

ઇ-કૉમર્સ અને એમ-કૉમર્સમાં ચુકવણી એ સૌથી મહત્ત્વનું પાસું છે. ચુકવણીની પરંપરાગત પદ્ધતિઓમાં રોકડ, ચૅક કે ક્રૅડિટકાર્ડ વડે થતી ચુકવણીઓનો સમાવેશ થાય છે. આજકાલ ઑનલાઇન કરવામાં આવતા વ્યવસાયોમાં ચુકવણીની ઇલેક્ટ્રૉનિક પદ્ધતિ વધુ મહત્ત્વની સાબિત થઈ રહી છે. કારણકે સંસ્થાઓ ગ્રાહકોને ઝડપી અને ઓછા ખર્ચે સેવા આપવાના જુદા-જુદા માર્ગ શોધતી હોય છે. ગ્રાહક અને વિક્રેતા વચ્ચે થયેલા ઑનલાઇન નાણાકીય વિનિમયને ઇલેક્ટ્રૉનિક ચુકવણી કહે છે. આજકાલ બજારમાં ઇલેક્ટ્રૉનિક ચુકવણીના અનેક વિકલ્પ ઉપલબ્ધ છે. ઇલેક્ટ્રૉનિક ચુકવણી માટે નીચે જણાવેલ પ્રકાર ઉપયોગમાં લેવાય છે :

- **ચુકવણી માટેનાં કાર્ડ (Payment Cards)**

ચુકવણી માટેનાં કાર્ડમાં ક્રૅડિટકાર્ડ, ડેબિટ કાર્ડ અને સ્માર્ટકાર્ડનો સમાવેશ કરી શકાય. સામાન્ય રીતે ગ્રાહક દ્વારા ખરીદી માટે ઉપયોગમાં લેવાતા તમામ પ્રકારનાં પ્લાસ્ટિકકાર્ડ માટે 'પેમેન્ટકાર્ડ' (payment card) પદનો ઉપયોગ કરવામાં આવે છે.

ક્રૅડિટકાર્ડ (Credit Card)

ઇન્ટરનેટ પર ચુકવણી માટેની આ સૌથી વધુ પ્રચલિત અને વ્યાપક પ્રમાણમાં સ્વીકૃત પદ્ધતિ છે. ઇશ્યૂઇંગ બૅન્ક (issuing bank) નામે ઓળખાતી બૅન્ક દ્વારા ગ્રાહકને ક્રૅડિટકાર્ડ આપવામાં આવે છે. ઇશ્યૂઇંગ બૅન્ક ક્રૅડિટકાર્ડના વ્યવસાયમાં સ્થાપિત અને નામાંકિત નાણાકીય સંસ્થાઓનાં ક્રૅડિટકાર્ડ પૂરાં પાડે છે. MasterCard® અથવા Visa® તેનાં ઉદાહરણ છે. ગ્રાહકને ધિરાણક્ષમતા અને આવકના સ્તર પર આધારિત ધિરાણ પૂરું પાડવામાં આવે છે. બિલિંગ સમયગાળા દરમિયાન ગ્રાહક તે ધિરાણની મર્યાદામાં રહી ઇશ્યૂઇંગ બૅન્ક સાથે ખર્ચ કે ચુકવણી કરી શકે છે.

ક્રૅડિટકાર્ડ બૅન્કનાં ખાતાં સાથે સંલગ્ન હોવાથી ગ્રાહક જ્યારે તેનો ઉપયોગ ઑનલાઇન ચુકવણી માટે કરે ત્યારે વિક્રેતા પોતાના માલની કિંમત ઇશ્યૂઇંગ બૅન્ક પાસેથી વસૂલ કરે છે અને બૅન્ક ગ્રાહકના હવે પછીના સ્ટેટમેન્ટમાં તે કિંમત બાદ કરીને બતાવે છે. ગ્રાહક બૅન્કને ચૂકવણી કરે છે. વેબસાઇટ દ્વારા ક્રૅડિટકાર્ડથી ચુકવણી સ્વીકારવા માટે વ્યાપારી એક્વાયરિંગ બૅન્ક (acquiring bank)ને નામે ઓળખાતી બૅન્કમાં વ્યાવસાયિક ખાતું ખોલાવે છે, જેથી તેને ઑનલાઇન અધિકૃતતા અને ચુકવણીની સેવા મળે છે. કાર્ડની સક્રિયતા, ખરીદી માટે અપાતા ધિરાણની મર્યાદા અને બિલિંગની માહિતી જેવી વિગતોની ચકાસણીને ગ્રાહકની અધિકૃતતા (Authorization) કહેવામાં આવે છે. વેપારીઓ દ્વારા ક્રૅડિટકાર્ડનો વ્યાપક પ્રમાણમાં ઉપયોગ કરવામાં આવી રહ્યો છે. અને તે વિક્રેતા તથા ગ્રાહકને વિશ્વસનીયતા પૂરી પાડે છે. આકૃતિ 5.24માં કેટલાંક ક્રૅડિટકાર્ડ દર્શાવ્યાં છે.

આકૃતિ 5.24 : ક્રૅડિટકાર્ડનાં ઉદાહરણ

ઈન્ટરનેટ પર કરવામાં આવતા ક્રેડિટકાર્ડના વ્યવહારમાં ગ્રાહક વિક્રેતાની વેબસાઇટની મુલાકાત લે છે અને ખરીદી માટે માલ પસંદ કરે છે. ત્યાર બાદ ક્રેડિટકાર્ડને સંબંધિત તમામ માહિતી ઉમેરી આ માહિતી ઇલેક્ટ્રોનિક પદ્ધતિથી વ્યાપારીને મોકલી આપવામાં આવે છે. આ વ્યવહારમાં ચાર પક્ષ સમાવિષ્ટ થાય છે :

- ક્રેડિટકાર્ડધારક ગ્રાહક
- ક્રેડિટકાર્ડ સ્વીકારતો વિક્રેતા
- ઈશ્યૂઈંગ બૅન્ક (Issuing Bank) : જે ગ્રાહકને ક્રેડિટકાર્ડ આપે છે અને વિક્રેતાને ચુકવણીની ખાતરી આપે છે. બૅન્ક ગ્રાહક પાસેથી ચુકવણીની રકમ પ્રાપ્ત કરે છે.
- એક્વાયરિંગ બૅન્ક (Acquiring banks) : વિક્રેતા સાથે ખાતું સ્થાપિત કરતી નાણાકીય સંસ્થા, જે ક્રેડિટકાર્ડની યથાર્થતા ચકાસે છે અને ગ્રાહકની ધિરાણમર્યાદાને આધારે વેચાણને અધિકૃત કરે છે.

ઑનલાઇન ચુકવણીમાં અન્ય બે પક્ષ ભાગ ભજવે છે : પેમેન્ટ ગેટ-વે (payment gateways) અને પેમેન્ટ પ્રોસેસર (payment processor). પે-પાલ (PayPal) જેવા ત્રાહિત પક્ષો દ્વારા પૂરી પાડવામાં આવતી સેવાને પેમેન્ટ ગેટ-વે તરીકે ઓળખવામાં આવે છે. તે અધિકૃતતા સ્થાપિત કરી ચુકવણી માટેના તમામ પક્ષોના નેટવર્કનું સુરક્ષિત સંકલન કરે છે. પ્રોસેસર્સ એવાં ડેટાસેન્ટર છે, જે ક્રેડિટકાર્ડના વ્યવહાર પૂરા કરે છે તથા વિક્રેતાને ભંડોળની પતાવટ કરે છે. પેમેન્ટ ગેટ-વે દ્વારા વ્યાપારીની ઇ-કૉમર્સ વેબસાઇટ સાથે જોડાયેલી હોય છે.

ઈન્ટરનેટ પર ક્રેડિટકાર્ડ દ્વારા થતી ઑનલાઇન ચુકવણી બે ભાગમાં વહેંચાયેલી છે : અધિકૃતતા (Authorization) અને વ્યવહારકરાર (Settlement).

અધિકૃતતાની પ્રક્રિયા દરમિયાન નીચેનાં પગલાં લેવામાં આવે છે :

- ચકાસણી (checkout) સમયે ગ્રાહક ક્રૅડિટકાર્ડની માહિતી સાથે વ્યવહારની તમામ વિગતો (જેવી કે, વસ્તુની વિગત, ખરીદ તારીખ વગેરે) ઇ-કૉમર્સ વેબસાઇટને પૂરી પાડે છે, જે પેમેન્ટ ગેટ-વેને મોકલી આપવામાં આવે છે.
- પેમેન્ટ ગેટ-વે આ માહિતી પ્રોસેસરને મોકલે છે, જે માહિતીની ચકાસણી માટે ઇશ્યૂઈંગ બૅન્કનો સંપર્ક કરે છે.
- ઈશ્યૂઈંગ બૅન્ક ચકાસણી બાદ તેનું પરિણામ (વ્યવહારનો સ્વીકાર કે અસ્વીકાર) પેમેન્ટ ગેટ-વેને મોકલી આપે છે.
- જો વિક્રેતાને વ્યવહારને સ્વીકારે તો વ્યવહાર-કરાર (settlement) બને છે અને તે સમયે વ્યવહારની રકમ ગ્રાહકના ખાતામાંથી વિક્રેતાના ખાતામાં હસ્તાંતરિત કરવામાં આવે છે.

કરાર કે ચુકવણીની પ્રક્રિયા દરમિયાન નીચેનાં પગલાં લેવામાં આવે છે :

- વિક્રેતા તમામ વિગતો સાથેની વ્યવહારની વિનંતી પેમેન્ટ ગેટ-વેને મોકલે છે. જે ત્યાર પછી પ્રોસેસરને મોકલવામાં આવે છે.
- પ્રોસેસર ચુકવણીની વિગતો ગ્રાહકની ઇશ્યૂઈંગ બૅન્કને મોકલી આપે છે. તે ચુકવણીની વિગતો એક્વાયરિંગ બૅન્કને પણ મોકલે છે, જ્યાં વિક્રેતાનું ખાતું છે.
- એક્વાયરિંગ બૅન્ક વિક્રેતાના ખાતામાં રકમ જમા કરે છે. ઇશ્યૂઈંગ બૅન્ક તમામ ખર્ચ સાથેનું બિલ ગ્રાહકને મોકલે છે, જે તેણે સૂચિત સમયમર્યાદામાં ચૂકવવાનું રહે છે.

ઑનલાઇન વ્યવહારોને સુરક્ષિત બનાવવા માટે મોટા ભાગની ક્રૅડિટકાર્ડ સંસ્થાઓ સિક્યોર ઇલેક્ટ્રૉનિક ટ્રાન્સફર (Secure Electronic Transfer - SET)નો ઉપયોગ કરે છે.

ક્રેડિટકાર્ડના ફાયદા નીચે મુજબ છે :

- ગ્રાહકને મોટી રોકડ રકમ લઈને ફરવું પડતું નથી. ગ્રાહક માલ અને સેવા માટે ઑનલાઇન અને ઑફલાઇન બંને રીતે ચુકવણી કરી શકે છે.
- બૅન્કના સ્ટેટમેન્ટની મદદથી ગ્રાહકની ખરીદીની નોંધ રાખી શકાય છે.
- બૅન્કના ખાતામાં રોકડ ઉપલબ્ધ ન હોય તે છતાં ગ્રાહકને માલ ખરીદવાની સુવિધા મળે છે. મહિનાના અંતે તેઓ રોકડ ચૂકવીને ખાતું સરભર કરે છે.

ક્રેડિટકાર્ડની મર્યાદાઓ નીચે મુજબ છે :

- બહુ નાની કે બહુ મોટી ચુકવણીઓ માટે તે અયોગ્ય છે. સુરક્ષાની દૃષ્ટિએ આ કાર્ડની એક મર્યાદા છે. ખૂબ જ મોટી રકમના વ્યવહારો માટે તેનો ઉપયોગ શક્ય નથી.
- ગ્રાહક ક્રેડિટકાર્ડનો ઉપયોગ કરી પોતાની ક્ષમતાથી વધુ ખર્ચ કરવાની માનસિકતા ધરાવી શકે છે.
- ક્રેડિટકાર્ડ ખોવાઈ કે ચોરાઈ જવાની શક્યતા રહે છે.

ડેબિટકાર્ડ (Debit Cards)

ડેબિટકાર્ડ દેખાવમાં ક્રેડિટકાર્ડ જેવું લાગે છે, પરંતુ તેનું કાર્ય અલગ છે. ગ્રાહકના બૅન્કખાતામાંથી સીધા જ વ્યાપારીને રકમનું હસ્તાંતરણ કરી શકાય તે પ્રકારનું આ કાર્ડ છે. ગ્રાહકના ખાતામાંથી રકમની તત્કાલ કપાત કરવામાં આવે છે. ડેબિટકાર્ડ ગ્રાહકને ખરીદીની મર્યાદામાં રાખે છે અને તેની ક્ષમતાથી વધુ ખરીદી કરવાની પરવાનગી આપતું નથી. ખરીદી માટે ડેબિટકાર્ડનો ઉપયોગ કરતી વખતે ગ્રાહકે હંમેશાં પોતાનાં ખાતાંની સિલકનું ધ્યાન રાખવું જરૂરી છે. આકૃતિ 5.25માં ડેબિટકાર્ડનું ઉદાહરણ આપવામાં આવ્યું છે.

આકૃતિ 5.25 : ડેબિટકાર્ડનું ઉદાહરણ

સ્માર્ટકાર્ડ (Smart Cards)

સ્માર્ટકાર્ડ ક્રેડિટકાર્ડ જેવું જ દેખાય છે, પરંતુ સપાટી પર માઇક્રોચિપ જડેલી હોવાથી તેનું કાર્ય અલગ પડે છે. ખાતાંઓની માહિતી, આરોગ્ય સંબંધિત વિમાની માહિતી, અંગત કી અને ઉપયોગકર્તાની અન્ય અંગત માહિતીનો સ્માર્ટકાર્ડમાં સંગ્રહ કરવામાં આવે છે. સામાન્ય કાર્ડ કરતા તે સો ગણી વધુ માહિતીનો સંગ્રહ કરી શકે છે. સ્માર્ટકાર્ડમાં સંગૃહીત માહિતી સાંકેતિક સ્વરૂપમાં હોવાથી તે ક્રેડિટ કે ડેબિટકાર્ડથી વધુ સુરક્ષિત છે. ગ્રાહકો કાર્ડમાં રોકડ પણ ઉમેરી શકે છે અને ત્યાર પછી છૂટક વ્યાપારીની વેબસાઇટ કે દુકાન પર માલની ખરીદી તેની ચુકવણી માટે તેનો ઉપયોગ પણ કરી શકે છે. છૂટક વ્યાપારીની દુકાન પર કાર્ડરીડર ઉપલબ્ધ હોય છે તથા તેને PC સાથે પણ જોડી શકાય છે. આ અનુકૂળતા સ્માર્ટકાર્ડને ઘણું ફાયદાકારક બનાવે છે. ડિજિટલ રોકડનો સંગ્રહ કરવા, દર્દીનો તબીબી અહેવાલ મેળવવા જેવા ઘણા હેતુઓ માટે તેનો ઉપયોગ કરી શકાય છે. યુ.એસ., યુરોપ, જાપાન અને એશિયાના કેટલાક ભાગમાં સ્માર્ટકાર્ડ ઘણાં પ્રચલિત છે.

સ્માર્ટકાર્ડમાં રહેલી માહિતી જોવા માટે તથા નવી માહિતી ઉમેરવા માટે કાર્ડરીડર જરૂરી છે. આ એક એવું નાનું ઉપકરણ છે જેમાં કાર્ડ દાખલ કરવામાં આવે છે. ઉદાહરણ તરીકે, તમે તમારા ફેમિલી ડૉક્ટર પાસે જાઓ છો અને તમારી બીમારીનું પુનઃનિરીક્ષણ કરી દવાઓ સૂચવવા તેમને સ્માર્ટકાર્ડ આપો છો. તમારા વિગતો તમારા સ્માર્ટકાર્ડમાં દાખલ કરી દેવાય છે. એ પછી દવાની દુકાનમાં જઈ ફાર્માસિસ્ટને તમે તમારું સ્માર્ટકાર્ડ આપો છો. તેમાંથી તે ડૉક્ટરની સૂચનાઓ વાંચી તમને નિયત દવાઓ આપી શકે છે. સ્માર્ટકાર્ડનો ઉપયોગ કરી ચુકવણી પણ કરી શકાય છે.

ચાર્જકાર્ડ (Charge Cards)

ચાર્જકાર્ડ પણ એક પ્રકારની ચુકવણીની પ્રક્રિયા છે, જેમાં ગ્રાહક કાર્ડ દ્વારા વિક્રેતાને ચુકવણી કરી શકે છે. ક્રેડિટકાર્ડની તુલનામાં ચાર્જકાર્ડમાં ધિરાણની કોઈ મર્યાદા હોતી નથી. કાર્ડ આપનાર સંસ્થાને ગ્રાહકે બિલિંગ સમયગાળાને અંતે પૂરેપૂરી રકમની ચુકવણી કરવાની હોય છે. જો આમ ન બને તો તેણે લેઇટ ફી ભરવી પડે છે.

- **નેટબૅંકિંગ (Net-Banking)**

નેટબૅંકિંગ અથવા ઑનલાઇન બૅંકિંગનો વિકલ્પ આજકાલ ઘણો પ્રચલિત થઈ રહ્યો છે. તેમાં કોઈ પ્રકારના કાર્ડનો સમાવેશ કરવામાં આવતો નથી. ઇન્ટરનેટ બૅંકિંગ સહિતનું બેન્કખાતું ધરાવતો ગ્રાહક તેનો ઉપયોગ કરી શકે છે. ગ્રાહક આ ખાતાના ઉપયોગ દ્વારા ઑનલાઇન ખરીદી તેમજ ચૂકવણી કરી શકે તે માટે બૅન્ક દ્વારા તેને નેટબૅંકિંગ પાસવર્ડ પૂરો પાડવામાં આવે છે. ઇ-કૉમર્સ/એમ-કૉમર્સની અનેક વેબસાઇટ નેટબૅંકિંગ દ્વારા ચુકવણીની સુવિધા પૂરી પાડે છે. વેબસાઇટ પર કાર્ડની વિગતો રજૂ કરવાને બદલે જે બૅન્ક દ્વારા ચુકવણી કરવાની હોય તેનું નામ પૂછવામાં આવે છે. આ વેબસાઇટ પર તમે ચુકવણીની પ્રક્રિયામાં આગળ વધો છો ત્યારે બૅન્કનાં નામની પસંદગી પૂછવામાં આવે છે. બૅન્કની પસંદગી કર્યા બાદ સ્ક્રીન પર તે બૅન્કની વેબસાઇટ જોવા મળે છે. જ્યાં ખાતા-નંબર અને નેટબૅંકિંગ પાસવર્ડ દ્વારા લોગ ઇન કરવાની જરૂર પડે છે. હવે તમે વ્યવહારની પ્રક્રિયા દ્વારા તમારા ખાતામાંથી વિક્રેતાનાં ખાતાંમાં રકમનું હસ્તાંતરણ કરી શકો છો.

ભારતીય રેલવે તેની વેબસાઇટ www.irctc.co.in પર ઑનલાઇન ટિકિટની નોંધણીની સુવિધા પૂરી પાડે છે. આ માટે ઉપયોગકર્તા સૌથી પહેલાં સાઇટ પર પોતાની નોંધણી (registration) કરાવી મુસાફરોની માહિતી સહિત મુસાફરીની વિગતો આપશે. એ પછી ચુકવણી માટે નેટબૅંકિંગના વિકલ્પ દ્વારા બૅન્કની પસંદગી કરી શકાય છે. અહીં તમારી બેન્કની લોગ-ઇન સ્ક્રીન દર્શાવવામાં આવશે. અને ભારતીય રેલવેને ચુકવણી કરી ટિકિટ નોંધણીની પ્રક્રિયા પૂર્ણ કરી શકાશે. સફળતાપૂર્વક ચૂકવણી કર્યા બાદ ઉપયોગકર્તા નોંધાયેલ મોબાઇલ નંબર પર irctc પરથી સંદેશ મેળવશે, જે મુસાફરી દરમિયાન ટિકિટની અવેજીમાં દર્શાવી શકાશે અથવા ટિકિટની ઇ-કૉપી (E-copy) પણ મુદ્રિત કરી શકાશે. નેટબૅંકિંગની ચુકવણી પદ્ધતિ ક્રેડિટકાર્ડની તુલનામાં વધુ સુરક્ષિત માનવામાં આવે છે અને ભારતમાં લગભગ દરેક વ્યાપારી સંસ્થાઓ આ વિકલ્પ પૂરો પાડે છે.

- **ઇલેક્ટ્રૉનિક ફંડ-ટ્રાન્સફર (Electronic Fund Transfer)**

એક બેન્કના ખાતામાંથી અન્ય બેન્કના ખાતામાં રકમ સ્થાનાંતરિત કરવાના કાર્યને ઇલેક્ટ્રૉનિક ફંડ-ટ્રાન્સફર તરીકે ઓળખવામાં આવે છે. કાગળના રૂપમાં ચેક કે પ્રત્યક્ષ ઉઘરાણી કરતાં તે સુરક્ષિત, નિશ્ચિત, અસરકારક અને ઓછું ખર્ચાળ છે. દુનિયાની બેન્કોના પારસ્પરિક વ્યવહાર, ATMના ઉપયોગથી ટ્યૂશન ફીની ચુકવણી, કર્મચારીઓના પગાર તેમનાં ખાતાંમાં જમા કરવા, બેન્કખાતાની માસિક કપાત વગેરે EFTનાં ઉદાહરણ છે. ઑનલાઇન ચુકવણી માટે EFTની અસરકારકતામાં વધારો, સરળ નામાપદ્ધતિ અને વધુ સુરક્ષા EFTના લાભ છે. જોકે, ઇન્ટરનેટ દ્વારા બિલ મોકલતી અને મેળવતી સંસ્થાઓની સંખ્યા હજી ઓછી છે.

- **ઈ-વૉલેટ (E-wallet)**

જ્યારે તમે વેબ પરથી ખરીદી કરો છો, ત્યારે તમારું નામ, માલ મોકલવાનું સરનામું, બિલ મોકલવા માટેનું સરનામું, ક્રેડિટકાર્ડની માહિતી સહિતનું એક ફૉર્મ ભરવાનું રહે છે. એકાદવાર આ કાર્ય યોગ્ય છે, પરંતુ દરેક સમયે ખરીદી કરતી વખતે આ કાર્ય કષ્ટસાધ્ય બની રહે છે. આ માટે, કેટલાક વ્યાપારીઓ ગ્રાહકે એક વાર ભરેલા ફૉર્મની વિગતોના પુનઃઉપયોગ માટે તેને પોતાના સર્વર પર સંગ્રહ કરે છે. આ વ્યાપારીઓ ગ્રાહકને ઇ-વૉલેટ સેવાઓ પૂરી પાડે છે, જેના દ્વારા ખરીદી કે ટિકિટ-નોંધણીનું કાર્ય વધુ સરળ બને છે. આજે ઘણી બૅન્કો, કરિયાણાની ઑનલાઇન દુકાનો, ટેલિફોનસેવાઓ વગેરે ઇ-વૉલેટ સેવાઓ પૂરી પાડે છે. વ્યાપારીઓને સુરક્ષિત રીતે ઑનલાઇન ચુકવણી માટેનાં ઇલેક્ટ્રૉનિક કાર્ડને ઇ-વૉલેટ કહે છે. તે ક્રેડિટકાર્ડ કે ડેબિટકાર્ડની જેમજ કાર્ય કરે છે. ઇ-વૉલેટ દ્વારા ચુકવણી કરતી વખતે ગ્રાહકે ક્રેડિટ કે ડેબિટકાર્ડ નંબર પૂરો પાડવો જરૂરી નથી. આથી, ક્રેડિટ/ડેબિટ કાર્ડની ગોપનીયતા નષ્ટ થવાનું જોખમ રહેતું નથી.

ઉદાહરણ તરીકે, ઑનલાઇન નોંધણી સરળ બનાવવા માટે IRCTCએ ગ્રાહકો માટે ઇ-વૉલેટ યોજના શરૂ કરી છે. ખાતું ધરાવનાર ગ્રાહક IRCTCમાં અગાઉથી પૈસા જમા કરાવી શકે છે, જેનો ઉપયોગ ભવિષ્યમાં ટિકિટની ઑનલાઇન નોંધણી માટે ચુકવણીના વિકલ્પ તરીકે કરી શકાય છે. હાલમાં ઉપયોગકર્તાને ટિકિટની નોંધણી માટે ક્રેડિટ/ડેબિટ-કાર્ડની જરૂર પડે છે. આ પ્રક્રિયામાં સમય લાગે છે તથા ગ્રાહકને ચુકવણી માટે બૅન્કના સર્વર પર સ્થાનાંતરિત કરવામાં આવે છે.

- **રુ-પે (RuPay)**

Rupee અને Payment એવાં બે પદ પરથી RuPay પદ તારવવામાં આવ્યું છે. નેશનલ પેમેન્ટ કૉર્પોરેશન ઑફ ઇન્ડિયા (NPCI)એ શરૂ કરેલી કાર્ડ દ્વારા ચુકવણીની આ એક નવી રીત છે. આકૃતિ 5.26માં RuPay કાર્ડ દર્શાવ્યું છે, જેનો ઉપયોગ ક્રેડિટકાર્ડ અને ડેબિટકાર્ડની જેમ જ કરી શકાય છે.

આકૃતિ 5.26 : RuPay કાર્ડ

હાલમાં, સમગ્ર વિશ્વમાં ફેલાયેલા કાર્ડધારકો, વ્યાપારીઓ અને ઇશ્યૂઇંગ બૅન્કો માસ્ટરકાર્ડ અને વીસા સાથે જોડાણ કરી આપે તેવી કોઈ સ્થાનિક ભારતીય બૅન્ક અસ્તિત્વમાં નથી. માસ્ટરકાર્ડ અને વીસાકાર્ડ ચુકવણીની દૃષ્ટિએ વૈશ્વિક અગ્રણીઓ છે. આ પ્રકારના તમામ વ્યવહાર તેમના દ્વારા કરવામાં આવે છે. સ્થાનિક બૅન્કો દ્વારા થતા ક્રેડિટકાર્ડ કે ડેબિટકાર્ડના વ્યવહારોને દેશબહાર આવેલા નેટવર્ક દ્વારા આગળ વધારવામાં આવે છે. આ સેવાઓ પૂરી પાડવા માટે તેઓ અતિરિક્ત કિંમત વસૂલ કરે છે, જેથી બૅન્કે તમામ ડેબિટ અને ક્રેડિટકાર્ડ ચુકવણીઓ માટેનો ખર્ચ ભોગવવો પડે છે. RuPay ચુકવણી માટે અન્ય કાર્ડનો સ્થાનિક વિકલ્પ છે. RuPay કાર્ડના ઉપયોગ દ્વારા થતા તમામ વ્યવહારો પર ભારતમાં જ પ્રક્રિયા કરવામાં આવે છે. વ્યવહારની પ્રક્રિયા સ્થાનિક સ્તરે થતી હોવાથી દરેક વ્યવહારની લેવડ-દેવડ અને કરારનો ખર્ચ ઘટાડી શકાય છે. આમ, ઓછું ખર્ચાળ હોવાને કારણે RuPay ગ્રાહક અને બૅન્ક માટે લાભદાયી છે.

સારાંશ

આ પ્રકરણમાં આપણે મોબાઇલ કૉમર્સ વિશે ચર્ચા કરી. ઇન્ટરનેટના ઉપયોગ દ્વારા મોબાઇલ ફોન, પર્સનલ ડિજિટલ આસિસ્ટન્ટ (PDA), સ્માર્ટફોન, ટેબ્લેટ PC અને પામટૉપ જેવાં વાયરલેસ સાધનોની મદદથી માલ અને સેવાના ખરીદ-વેચાણની પ્રક્રિયાને એમ-કૉમર્સ કહે છે. એમ-કૉમર્સના ફાયદાઓ અને મર્યાદાઓ વિશે પણ આપણે ચર્ચા કરી. મોબાઇલ ઉપયોગકર્તાઓની સંખ્યામાં દિન-પ્રતિદિન થતી વૃદ્ધિને કારણે એમ-કૉમર્સ વિનિયોગ ઉત્તરોત્તર પ્રચલિત બની રહ્યા છે. ઇ-કૉમર્સના ભવિષ્ય માટે સ્થાન આધારિત વ્યાવસાયિક વિનિયોગો અને સેવાઓ તેના સૂચિતાર્થો સહિત અગ્રેસર થઈ રહી છે. આ વિનિયોગ દ્વારા ઉત્પાદન કે સેવા પૂરી પાડવા માટે ઉપયોગકર્તાનું નિશ્ચિત સ્થાન શોધી કાઢવામાં આવે છે. આપણે વિવિધ પ્રકારના સુરક્ષાના મુદ્દા અને ભયસ્થાનો વિશે ચર્ચા કરી. ઇ-કૉમર્સની સુરક્ષામાં અવરોધરૂપ ધમકીઓના નિવારણ માટે લઈ શકાતાં પગલાં અંગેની ચર્ચા પણ કરવામાં આવી. ઇ-કૉમર્સને લગતા કાયદાકીય મુદ્દાઓ પણ આપણે જોયા. ઇ-કૉમર્સ દ્વારા પૂરાં પાડવામાં આવતાં ક્રેડિટકાર્ડ, ડેબિટકાર્ડ, સ્માર્ટકાર્ડ, રુ-પે, નેટબૅંકિંગ અને ઇ-વૉલેટ જેવા ચુકવણીના વિવિધ વિકલ્પો વિશે પણ ચર્ચા કરવામાં આવી.

સ્વાધ્યાય

1. એમ-કૉમર્સ એટલે શું ? એમ-કૉમર્સનાં ઉદાહરણોની યાદી બનાવો.
2. એમ-કૉમર્સ ઉપયોગકર્તાને શા માટે ઉપયોગી છે ?
3. એમ-કૉમર્સના ફાયદાની યાદી બનાવો.
4. એમ-કૉમર્સની મર્યાદાઓ કઈ છે ?
5. એમ-કૉમર્સના વિનિયોગની યાદી બનાવો.
6. નીચેના એમ-કૉમર્સ વિનિયોગો માટે વેબસાઇટનાં ઉદાહરણ આપો :
   (1) મોબાઇલ દ્વારા ટિકિટની નોંધણી (2) મોબાઇલ દ્વારા હરાજી
   (3) મોબાઇલ દ્વારા ખરીદી (4) મોબાઇલ દ્વારા માહિતી સેવા
   (5) મોબાઇલ દ્વારા વાણિજ્યિક સેવા
7. એલ-કૉમર્સ એટલે શું ?
8. GPS એટલે શું ? તે સાધનને કેવી રીતે શોધે છે ?
9. સ્થાનઆધારિત સેવાના વિનિયોગોની યાદી બનાવો.
10. ઇ-કૉમર્સની સુરક્ષાનાં ચાર મહત્ત્વના પાસાં ક્યાં છે ?
11. ઇન્ટરનેટની સુરક્ષાનાં નીચે જણાવેલ જોખમો વિશે ખ્યાલ આપો :
    (1) દૂષિત કોડ (2) સ્નિફર (3) સેવામાં અવરોધરૂપ આક્રમણ
12. સાયબર જંગલીપણું (vandalism) એટલે શું ?
13. 'સ્પૂફિંગ' (spoofing) પદની સમજૂતી આપો.
14. સુરક્ષા સામેની ધમકીઓ માટે લઈ શકાતાં પગલાંની યાદી બનાવો.
15. ડિજિટલ સર્ટિફિકેટમાં શેનો સમાવેશ કરવામાં આવે છે ?
16. સાંકેતીકરણ એટલે શું ? સાંકેતીકરણ પદ્ધતિનો ઉપયોગ કરી "Gandhi Ashram" લખાણના દરેક અક્ષરને તેની પહેલાંના અક્ષર સાથે બદલી ગુપ્ત ભાષામાં ફેરવો.

17. SSLનો ઉદ્દેશ શું છે ?

18. બૌદ્ધિક સંપત્તિને સંબંધિત મુદ્દાઓની યાદી બનાવો.

19. બૌદ્ધિક સંપત્તિની સુરક્ષા માટે જુદા-જુદા ક્યા માર્ગો ઉપલબ્ધ છે ?

20. ઇલેક્ટ્રૉનિક ચુકવણીની જુદી-જુદી પદ્ધતિઓની યાદી બનાવો.

21. ક્રેડિટકાર્ડના ફાયદા અને મર્યાદાઓની યાદી બનાવો.

22. ક્રેડિટકાર્ડથી સ્માર્ટકાર્ડ કેવી રીતે અલગ પડે છે ?

23. ઇલેક્ટ્રૉનિક ફંડનું હસ્તાંતરણ (transfer) એટલે શું ?

24. આપેલા વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :

(1) ઇન્ટરનેટ ધરાવતા વાયરલેસ સાધનનો ઉપયોગ કરી માલ કે સેવાના ખરીદ-વેચાણને શું કહે છે ?

(a) ઇન્ટરનેટ (b) એમ-કૉમર્સ (c) એમ-બૅન્કિંગ (d) WWW

(2) વ્યાવસાયિક હેતુ માટે સ્થાનની માહિતી પૂરી પાડતી તક્નિકના ઉપયોગને શું કહે છે ?

(a) ઇ-કૉમર્સ (b) એમ-કૉમર્સ

(c) એલ-કૉમર્સ (d) પરંપરાગત કૉમર્સ

(3) GPS એટલે શું ?

(a) Global Positioning System (b) Global Postal System

(c) Grand Positioning System (d) Google Positioning System

(4) અનધિકૃત ઉપયોગકર્તા વાંચી ન શકે તે માટે માહિતીને ગુપ્ત રાખવાની પદ્ધતિને સુરક્ષાના ક્યા પ્રકાર તરીકે ઓળખવામાં આવે છે ?

(a) ગુપ્તતા (Confidentiality) (b) અખંડિતતા (Integrity)

(c) અસ્વીકાર (Non-repudiation) (d) અધિકૃતતા (Authorization)

(5) વિતરણ દરમિયાન વિગતને અકસ્માતથી કે દૂષિત ઇરાદાઓથી બદલવામાં આવ્યા નથી કે તેની સાથે કોઈ ચેડાં કરવામાં આવ્યાં નથી – આને સુરક્ષાનો ક્યો પ્રકાર કહેવાય ?

(a) ગુપ્તતા (b) અખંડિતતા

(c) અસ્વીકાર (d) અધિકૃતતા

(6) માત્ર અધિકૃત ઉપયોગકર્તાને જ સિસ્ટમના ઉપયોગની પરવાનગી આપવામાં આવે – તે સુરક્ષાનો ક્યો પ્રકાર દર્શાવે છે ?

(a) અધિકૃતતા (b) ગુપ્તતા (c) અસ્વીકાર (d) અખંડિતતા

(7) નીચેનામાંથી સુરક્ષાનો ક્યો પ્રકાર ખાતરી આપે છે કે સંદેશ મોકલનાર સંદેશ મોકલ્યાનો ઇનકાર કરી શકશે નહીં ?

(a) અધિકૃતતા (b) ગુપ્તતા (c) અસ્વીકાર (d) અખંડિતતા

(8) ઇન્ટરનેટ પર કમ્પ્યૂટર કે રાઉટર દ્વારા પ્રેષકથી પ્રાપ્તકર્તા સુધી મોકલાતી માહિતીની નોંધ રાખતા પ્રોગ્રામને શું કહે છે ?

(a) Sniffer (b) Denial of service attack

(c) Malicious code (d) Spoofing

(9) ઉપયોગકર્તાના મશીન કે નેટવર્કને સ્થગિત કરી નાખી તેને નિરુપયોગી બનાવી દેવાનું આક્રમણ નીચેનાંમાંથી ક્યા નામે ઓળખાય છે ?

(a) Malicious code (b) Denial-of-Service

(c) Spoofing (d) Cyber vandalism

(10) હયાત વેબસાઇટના પાનાને ઇલેક્ટ્રૉનિક પદ્ધતિથી સુધારવાની પદ્ધતિને શું કહે છે ?

(a) Cyber vandalism (b) Denial-of-Service

(c) Spoofing (d) Malicious code

(11) કોઈની સમક્ષ તમે જે નથી તે સ્વરૂપે રજૂ થવું અથવા કોઈ નકલી વેબસાઇટને પ્રમાણભૂત વેબસાઇટ તરીકે રજૂ કરવી તેને શું કહેવાય છે ?

(a) Cyber vandalism (b) Malicious code

(c) Denial-of-Service (d) Spoofing

(12) વાયરસ, વૉર્મ અને ટ્રોજનહૉર્સ જેવા દૂષિત કોડને શોધી, અટકાવી અને દૂર કરવાની પ્રક્રિયા કરતા કમ્પ્યૂટર-પ્રોગ્રામને શું કહે છે ?

(a) ઍન્ટિવાઇરસ સૉફ્ટવેર (b) ડિજિટલ સર્ટિફિકેટ

(c) ફાયરવૉલ (d) ક્રિપ્ટોગ્રાફી

(13) એન્ક્રિપ્શન અલ્ગોરિધમનો ઉપયોગ કરી 'સાદા લખાણ' નામે ઓળખાતા મૂળ લખાણને 'ગુપ્ત લખાણ' તરીકે ઓળખાતા અવાચ્ય લખાણમાં પરિવર્તિત કરવાની ક્રિયાને શું કહે છે ?

(a) ફાયરવૉલ (b) એન્ક્રિપ્શન

(c) ઍન્ટિવાઇરસ સૉફ્ટવેર (d) ડિજિટલ સર્ટિફિકેટ

(14) ગુપ્ત લખાણને ફરી મૂળ લખાણમાં પરિવર્તિત કરવાની ક્રિયાને શું કહે છે ?

(a) ફાયરવૉલ (b) ડિજિટલ સર્ટિફિકેટ

(c) ડિક્રિપ્શન (d) વાઇરસ

(15) ઇન્ટરનેટ પર વેબવ્યવહારોને સુરક્ષિત રાખવા માટે ક્યા પ્રોટોકોલનો ઉપયોગ કરવામાં આવે છે ?

(a) TCP/IP (b) HTTP (c) Bluetooth (d) SSL

(16) SSL પ્રોટોકોલની રચના કોણે કરી ?

(a) Google (b) Netscape (c) Yahoo (d) Firefox

(17) નીચેનામાંથી વેબસાઇટની કઈ શરૂઆત દર્શાવે છે કે સાઇટને SSL દ્વારા સુરક્ષિત કરવામાં આવેલી છે ?

(a) http:// (b) ssl:// (c) https:// (d) http-ssl://

(18) સર્જકના મૂળભૂત કાર્યનો ઉપયોગ અન્ય અનધિકૃત ઉપયોગકર્તા દ્વારા થતો અટકાવવા સર્જકને કઈ સુવિધા પૂરી પાડવામાં આવે છે ?

(a) ટ્રેડમાર્ક (b) કૉપીરાઇટ (c) ડિજિટલ વૉટરમાર્કિંગ (d) સ્ટેગેનોગ્રાફી

(19) ઉત્પાદન કે સેવાને બજારના અન્ય ઉત્પાદન કે સેવાથી અલગ રાખવા વ્યક્તિ કે સંસ્થા દ્વારા ઉપયોગમાં લેવામાં આવતા નિશ્ચિત લોગો, શબ્દ, નિશાની, શૈલી કે શબ્દસમૂહ કે ચિત્રને શું કહે છે ?

(a) ટ્રેડમાર્ક (b) કૉપીરાઇટ (c) ડિજિટલ વૉટરમાર્કિંગ (d) સ્ટેગેનોગ્રાફી

(20) નીચેનામાંથી કઈ ટ્રેડમાર્કની નિશાની છે ?

(a) TM, MT and © (b) TM, MS and ®

(c) TM, SM and ® (d) TM, SM and ©

(21) એક માહિતીમાં અન્ય માહિતી સંતાડવાની ક્રિયાને શું કહે છે ?

(a) Squatting (b) Steganography (c) Name changing (d) Copyright

(22) ફાઇલની કૉપીરાઇટ માહિતીને ઓળખવા માટે ચિત્ર, ઑડિયો કે વીડિયોમાં ઉમેરવામાં આવતા ડિજિટલ કોડને શું કહે છે ?

(a) Image mark (b) Digital mark

(c) Code mark (d) Watermark

(23) ઉપયોગકર્તાને ક્રેડિટકાર્ડ તથા વેપારીને ચુકવણીની ખાતરી કોણ આપે છે ?

(a) વેપારી (b) ઇશ્યૂઇંગ બૅન્ક (c) એક્વાયરિંગ બૅન્ક (d) ગ્રાહક

(24) નીચેનાંમાંથી ચુકવણીના ક્યા કાર્ડની સપાટી પર માઇક્રોચિપ જડેલી હોય છે ?

(a) સ્માર્ટકાર્ડ (b) ડેબિટકાર્ડ (c) ક્રેડિટકાર્ડ (d) ચાર્જકાર્ડ

ઑબ્જેક્ટ આધારિત ખ્યાલો 6

આપણે આજે ઇન્ટરનેટ, વેબસાઇટ અને વેબ આધારિત પ્રક્રિયાઓના યુગમાં છીએ કે જ્યાં વિનિયોગનો ઝડપી વિકાસ તેમજ સૉર્સ કોડ (source code)નો પુનઃ ઉપયોગ ઘણો અગત્યનો છે. સૉફ્ટવેર સિસ્ટમનાં વિશ્લેષણ, ડિઝાઇન અને અમલીકરણમાં ઑબ્જેક્ટ (object) આધારિત પદ્ધતિ કે ક્રિયા નોંધપાત્ર ભૂમિકા ભજવે છે. ઑબ્જેક્ટ આધારિત પદ્ધતિનો ઉપયોગ કરીને બનાવવામાં આવેલા સૉફ્ટવેર વધુ વિશ્વસનીય છે અને તેની જાળવણી, પુનઃઉપયોગ અને તેના વિકાસનું કાર્ય ખૂબ સરળ છે. આ પ્રકરણમાં ઑબ્જેક્ટ આધારિત ખ્યાલોની સામાન્ય સમજ આપેલી છે. જાવા પ્રોગ્રામિંગ ભાષાનો ઉપયોગ કરીને આ ખ્યાલોનું અમલીકરણ હવે પછીનાં પ્રકરણોમાં આવરી લેવામાં આવ્યું છે.

પરિચય (Introduction)

ઑબ્જેક્ટ (object) આધારિત પ્રોગ્રામિંગની શરૂઆત 1960ના સમયગાળામાં થઈ અને 1980ના દાયકાની મધ્યથી નવા સૉફ્ટવેર બનાવવામાં પ્રોગ્રામિંગની તે મુખ્ય પદ્ધતિ બની ગઈ. સૉફ્ટવેર સિસ્ટમના અતિ ઝડપથી વધતા કદ અને જટિલતાનું નિયંત્રણ કરવા માટે તેમજ મોટી અને જટિલ સિસ્ટમને સમય સાથે સુધારવાના કાર્યને સરળ બનાવવાના એક માર્ગ તરીકે આ પદ્ધતિને વિકસાવવામાં આવી હતી. કેટલીક પ્રચલિત પ્રોગ્રામિંગ ભાષાઓ જેવી કે C++, Java, C#, VB.net, ASP.net અને PHP ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગને સમર્થન આપે છે.

પ્રોગ્રામિંગની રીતને આપણે બે પ્રકારમાં વહેંચી શકીએ, જેમકે પ્રક્રિયાગત પ્રોગ્રામિંગ (Structure/Procedural Programming) અને ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ. પ્રક્રિયાગત પ્રોગ્રામિંગમાં આપણું કેન્દ્રબિંદુ ડેટા ઉપરની કાર્યપ્રણાલી (functions) તેમજ પ્રક્રિયાઓ (procedures) લખવામાં કેન્દ્રિત રહે છે. ઉદાહરણ તરીકે, પુસ્તકાલય વિનિયોગ સૉફ્ટવેર માટે આપણે પુસ્તકાલયના વિનિયોગની બધી પ્રક્રિયા બાબત વિચારીશું અને આપણું ધ્યાન વિદ્યાર્થી-નોંધણી, પુસ્તક-વિતરણ, પુસ્તક પરત કરવું અને દંડની ગણતરી જેવા વિભાગો ઉપર કેન્દ્રિત રહેશે.

ઑબ્જેક્ટ આધારિત પદ્ધતિમાં આપણા કેન્દ્રબિંદુએ **ઑબ્જેક્ટ (object)** હોય છે કે જે ડેટા તેમજ ક્રિયાત્મકતા (functionality) એમ બંને એકસાથે ધરાવે છે. પુસ્તકાલય વિનિયોગ બનાવવામાં આપણું ધ્યાન વિનિયોગમાં સમાવેશ વિવિધ ઑબ્જેક્ટ ઉપર કેન્દ્રિત હોય છે. અહીં આપણે વિદ્યાર્થી, પુસ્તક અને ગ્રંથપાલ જેવા ઑબ્જેક્ટ વિચારી શકીએ. આવા ઑબ્જેક્ટ એકબીજા સાથે કઈ રીતે સંકળાયેલા છે (association) તે બાબત પણ વિચારવું જોઈએ. ઉદાહરણ તરીકે, વિદ્યાર્થી પુસ્તકાલયમાં પુસ્તક પરત કરે છે.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ભાષાઓની તાકાત (ક્ષમતા) પ્રોગ્રામરને વિભાગીય (modular), પુનઃઉપયોગમાં લઈ શકાય તેમજ તે પ્રોગ્રામનો વિકાસ કરી શકાય (extendable) તે પ્રકારનો પ્રોગ્રામનો કોડ લખવાનું સામર્થ્ય પૂરું પાડે છે. આ ક્ષમતાને કારણે પ્રોગ્રામર હયાત મૉડ્યુલમાં ફેરફાર કરીને નવા પ્રોગ્રામની રચના કરી શકે છે. સૉફ્ટવેરના અન્ય ભાગના કોડમાં ખલેલ પહોંચાડ્યા વગર મૉડ્યુલમાં ફેરફાર કરવા કે બદલવાની સમર્થતા વડે તે ભાષાઓને લવચિકતા બક્ષે છે. હયાત કોડનો ફરી ઉપયોગ તેમજ હયાત કોડને સુધારીને ઉપયોગ કરીને સૉફ્ટવેર બનાવવાની ઝડપ વધારી શકાય છે.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ઑબ્જેક્ટને (object) પાયાના એકમ તરીકે વાપરે છે. એકસરખા ઑબ્જેક્ટનું ક્લાસ (class)ના ખ્યાલ દ્વારા વર્ગીકરણ કરવામાં આવે છે. કમ્પ્યૂટરની જે ભાષાઓ ઑબ્જેક્ટના ચાર ચોક્કસ ગુણધર્મો (1) **ઍબ્સ્ટ્રૅક્શન** (abstraction) (2) **ઈનકૅપ્સ્યુલેશન** (encapsulation) (3) **પોલિમોર્ફિઝમ** (polymorphism) અને (4) **ઈનહેરીટન્સ** (inheritance) પૂરા પાડે છે, તે ભાષા ઑબ્જેક્ટ આધારિત ભાષા તરીકે ઓળખાય છે.

ઑબ્જેક્ટ (Object)

વાસ્તવિક વિશ્વમાં ઑબ્જેક્ટ એ આ દુનિયા જે વસ્તુઓ વડે બનેલી છે, તે વસ્તુ છે. આમાંની કેટલીક વસ્તુઓ વ્યક્તિ, કાર કે કૉફીના પ્યાલા જેવા કોઈ ભૌતિક સ્વરૂપમાં હયાતી ધરાવે છે. અન્ય વસ્તુઓ અમૂર્ત સ્વરૂપે (abstract) હોઈ શકે કે જેને સ્પર્શ ન કરી શકાય અથવા જોઈ ન શકાય, ઉદાહરણ તરીકે તારીખ અને સમય જેવા ખ્યાલો.

દરેક ઑબ્જેક્ટ (object)ની એક અનન્ય ઓળખ હોય છે અને દરેક ઑબ્જેક્ટને એકબીજાથી અલગ ઓળખી શકાય છે. ઉદાહરણ તરીકે, દરેક વ્યક્તિ નામ, શહેર, જાતિ, જન્મતારીખ અને વ્યવસાય જેવી લાક્ષણિકતાઓ ધરાવે છે. ઑબ્જેક્ટ આધારિત પરિભાષામાં આવાં લક્ષણો **પ્રોપર્ટી** (property) અથવા **ઍટ્રિબ્યૂટ** (attribute) તરીકે ઓળખાય છે. (જેને આપણે

ગુણધર્મ કે સંબંધિત ગુણધર્મ પણ કહી શકીએ.) એક વ્યક્તિને બીજી વ્યક્તિથી અલગ પાડવા માટે આપણે તેની લાક્ષણિકતાની કિંમતનો ઉપયોગ કરીએ છીએ. 'રામ' અને 'શ્યામ' નામ બે અલગ-અલગ વ્યક્તિઓની ઓળખ આપે છે, પણ જ્યારે બે વ્યક્તિઓનાં નામ એક જ હોય, ત્યારે જન્મતારીખ જેવી અન્ય લાક્ષણિકતાની મદદથી આપણે તેમને અલગ પાડી શકીએ. આ રીતે ઑબ્જેક્ટને ઓળખવા માટે આપણે આ લાક્ષણિકતા (attribute)ની કિંમત વાપરીએ છીએ. આ કિંમતો **સ્ટેટ** (state) તરીકે ઓળખાય છે. આ ઉપરાંત ઑબ્જેક્ટ સાથે **બિહેવ્યર** (behaviour) સંકળાયેલ હોય છે. ઉદાહરણ તરીકે, વ્યક્તિ જન્મ લે છે, નામ મેળવે છે, સ્થાન બદલે છે. બિહેવ્યર **મેથડ** (method) તરીકે પણ ઓળખાય છે. ઑબ્જેક્ટની કિંમત તેના બિહેવ્યરના કારણે બદલાઈ શકે છે. આ રીતે, વાસ્તવિક વિશ્વમાં કોઈ પણ વસ્તુ (ઑબ્જેક્ટ) તે કયા નામથી ઓળખાય છે (ઓળખ - identity), તે શું છે (તેની સ્થિતિ - state) અને તે શું કરે છે (behaviour) તેના દ્વારા વર્ણવી શકાય છે.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગમાં ઑબ્જેક્ટનું વર્ણન કરતી જુદી-જુદી લાક્ષણિકતાઓ **ડેટાફિલ્ડ** (data-field) તરીકે પણ ઓળખવામાં આવે છે. ઑબ્જેક્ટ સાથે સંકળાયેલા વિવિધ ડેટા ઍટ્રિબ્યૂટ અને બિહેવ્યર મેથડને સામૂહિક રીતે તેના **મેમ્બર** કે ફિચર (member અથવા feature) તરીકે ઓળખવામાં આવે છે.

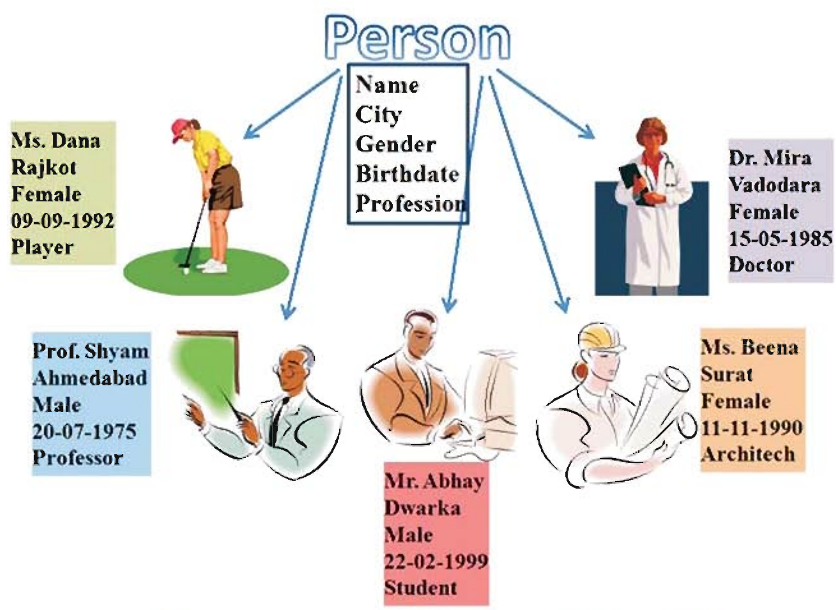
જ્યારે આપણે કોઈ સૉફ્ટવેર વિનિયોગની રચના કરીએ છીએ, ત્યારે આપણે જે ઑબ્જેક્ટ પસંદ કરીએ, તે વિનિયોગ માટે અર્થપૂર્ણ હોવા જોઈએ. ઉદાહરણ તરીકે, રેલ્વે-આરક્ષણ વિનિયોગ માટે રેલ્વે ટ્રેન, પ્રવાસી, ટિકિટ અને સ્ટેશન જેવા ઑબ્જેક્ટ યોગ્ય છે, પણ કાર, કમ્પ્યૂટર અને ઘડિયાળ જેવા ઑબ્જેક્ટ આ વિનિયોગ માટે અસંગત છે.

ક્લાસ (Class)

ઉપર જણાવેલાં person ઑબ્જેક્ટના ઉદાહરણમાં સહેલાઈથી જોઈ શકાય છે કે કેટલાક ઑબ્જેક્ટ એકસરખી લાક્ષણિકતા અને બિહેવ્યર ધરાવતા હોય છે પણ ફક્ત તેના સ્ટેટથી (state) (ડેટાની કિંમત) એકબીજાથી જુદા પડે છે. ઑબ્જેક્ટ આધારિત પદ્ધતિ ક્લાસ (class)ના ખ્યાલનો ઉપયોગ કરે છે, જે ફક્ત તેના ઍટ્રિબ્યૂટની કિંમતોથી જ અલગ હોય તેવા અમૂર્ત સમકક્ષ (abstractly equivalent) ઑબ્જેક્ટને અભિવ્યક્ત કરવા સમર્થ બનાવે છે. ક્લાસને અનેક જુદા-જુદા ઑબ્જેક્ટની એક બ્લ્યુપ્રિન્ટ (નકશો) તરીકે ગણી શકાય.

ક્લાસ એ સમાન લક્ષણો ધરાવતાં બહુવિધ ઑબ્જેક્ટનું એક ટેમ્પ્લેટ (template) છે. તે એકસમાન ઍટ્રિબ્યૂટ અને બિહેવ્યર ધરાવતાં વિવિધ ઑબ્જેક્ટનું એક જૂથ છે. એક જ ક્લાસના જુદા-જુદા ઑબ્જેક્ટનો સિમેન્ટિક હેતુ (semantic purpose) એકસરખો હોય છે. આ રીતે ક્લાસ એ કોઈ ચોક્કસ ઑબ્જેક્ટના જૂથની બધી જ સમાન લાક્ષણિકતાનો સમાવેશ કરવા માટેનો એક સર્વસાધારણ ખ્યાલ છે.

ઉદાહરણ તરીકે, આપણી પાસે બધી વ્યક્તિઓના સામાન્ય ઍટ્રિબ્યૂટ અને બિહેવ્યરનો સમાવેશ કરતો 'Person' નામનો ક્લાસ છે. અહીં દરેક વ્યક્તિ એ ઑબ્જેક્ટ છે અને તે તેના સ્ટેટ એટલે કે ઍટ્રિબ્યૂટના કિંમતથી તે ઓળખાય છે. આકૃતિ 6.1માં ક્લાસ અને તેના ઑબ્જેક્ટનો તફાવત દર્શાવેલો છે.



આકૃતિ 6.1 : 'Person' નામનો ક્લાસ અને તેના ઑબ્જેક્ટ

હવે આપણે અન્ય ઑબ્જેક્ટ આધારિત ખ્યાલો વિશે શીખતાં પહેલાં ક્લાસ-ડાયાગ્રામ (class-diagram) વિશે ટૂંકમાં પરિચય મેળવીએ.

ક્લાસ-ડાયાગ્રામનો પરિચય (Introduction to Class-Diagram)

ક્લાસ-ડાયાગ્રામ અનેક ક્લાસનો સમૂહ, અવરોધો (constraints) અને જુદા-જુદા ક્લાસ વચ્ચેના સંબંધની સ્થિતિ રજૂ કરે છે.

યુનિફાઇડ મૉડેલિંગ લૅંગ્વેજ (Unified Modelling Language - UML)નો ઉપયોગ ઑબ્જેક્ટ આધારિત સૉફ્ટવેરની પ્રતિકૃતિ (model) તૈયાર કરવામાં કરી શકાય કે જે વિનિયોગની રચના તૈયાર કરવામાં મદદરૂપ થાય. UML એ એક દૃશ્ય (visual) મૉડેલિંગ ભાષા છે અને તે ઑબ્જેક્ટ મૅનેજમૅન્ટ ગ્રૂપ (Object Management Group - OMG) દ્વારા વ્યાખ્યાયિત કરેલી છે અને તેની જાળવણી કરે છે. UML સૉફ્ટવેર વિનિયોગનાં વિવિધ પાસાંઓ રજૂ કરવા માટે નામનિર્દેશવાળી અનેક આકૃતિઓ જણાવે છે.

ક્લાસ-ડાયાગ્રામનો હેતુ વિનિયોગના સ્થાયી દેખાવની પ્રતિકૃતિ બનાવવાનો છે. ક્લાસ-ડાયાગ્રામ જ ફક્ત રેખાકૃતિઓ (diagrams) છે કે જે ઑબ્જેક્ટ આધારિત ભાષાઓ સાથે સીધેસીધી જોડી શકાય છે અને આથી સૉફ્ટવેર બનાવનાર વ્યક્તિઓમાં તે વ્યાપકપણે વપરાય છે.

ક્લાસ-ડાયાગ્રામમાં કોઈ પણ ક્લાસને રજૂ કરવા માટેનો આઇકોન નીચે જણાવ્યા પ્રમાણે name, attribute અને behaviourનો સમાવેશ કરવા માટે એક લંબચોરસ ત્રણ વિભાગમાં વિભાજિત કરેલો હોય છે :

1. સૌથી ઉપરના વિભાગમાં ક્લાસનું નામ (class name) હોય છે.
2. વચ્ચેના વિભાગમાં ક્લાસનાં ઍટ્રિબ્યૂટ કે પ્રોપર્ટી હોય છે.
3. સૌથી નીચેના વિભાગમાં ક્લાસનું બિહેવ્યર અથવા ઑપરેશન (operation) અથવા મેથડ (method) હોય છે.

આકૃતિ 6.2માં UML પ્રણાલિકા પ્રમાણે ક્લાસની સચિત્ર રજૂઆત દર્શાવી છે, જ્યારે આકૃતિ 6.3માં 'Person'નો ક્લાસ-ડાયાગ્રામ દર્શાવ્યો છે.

Class Name
Visibility attribute : data type = initial value
Visibility operation (argument list) : return type

આકૃતિ 6.2 : UML પ્રણાલિકાગત ક્લાસની રજૂઆત

Person
name : string city: string gender : char = 'M' -birthdate : date profession : string
setBirthdate (d:int,m:int,y:int): date changeCity (newCity:string) :string display ( )

આકૃતિ 6.3 : 'Person' ક્લાસની રેખાકૃતિ

'Person' ક્લાસનો હેતુ ક્લાસ-ડાયાગ્રામના સંદર્ભમાં ક્લાસની કામગીરી સમજવામાં મદદરૂપ માહિતી પૂરી પાડવાનો છે. તેણે ક્લાસનાં દરેક ઍટ્રિબ્યૂટ અને કાર્યોનો સમાવેશ કરવાની જરૂર નથી.

આકૃતિ 6.2માં દર્શાવ્યા પ્રમાણે, UMLની સંકેતલિપિમાં ઍટ્રિબ્યૂટની વાક્યરચના નીચે દર્શાવ્યા પ્રમાણે કરી શકાય :

[<visibility>] <attribute name> [: <attribute data type> [= <initial value>]]

અહીં, ચોરસ કૌંસ [ ]ની જોડીમાં લખવામાં આવેલી બાબત વૈકલ્પિક છે અને કોણીયકૌંસ < > ની જોડીમાં લખેલી બાબતની કિંમત વપરાશકર્તાએ જણાવવી પડે છે. અહીં દ્રશ્યતા (visibility) અંગત, સુરક્ષિત, જાહેર અથવા પૅકેજ (package) હોઈ શકે અને તે જણાવવા માટે અનુક્રમે –, #, + અને ~ સંકેતો વાપરવામાં આવે છે. આપણે દ્રશ્યતા વિશે હવે પછીના પ્રકરણમાં અભ્યાસ કરીશું. ઍટ્રિબ્યૂટ સામાન્ય રીતે ચલ (variable)નો નિર્દેશ કરે છે. ડેટાટાઇપ અને તેની પ્રારંભિક કિંમત પ્રોગ્રામની શરૂઆતમાં ક્યા પ્રકારનો ડેટા સંગ્રહ કર્યો છે અને તેની કિંમત શું છે તેનો નિર્દેશ કરે છે. અહીં જોઈ શકાય છે કે માત્ર ઍટ્રિબ્યૂટ નેઇમ (attribute-name) જ ફરજિયાત છે અને અન્ય તમામ બાબત વૈકલ્પિક છે.

ઍટ્રિબ્યૂટ ઘોષિત કરવા માટે નીચે કેટલાંક ઉદાહરણ આપેલાં છે :

name : string

- balance : float = 0.0

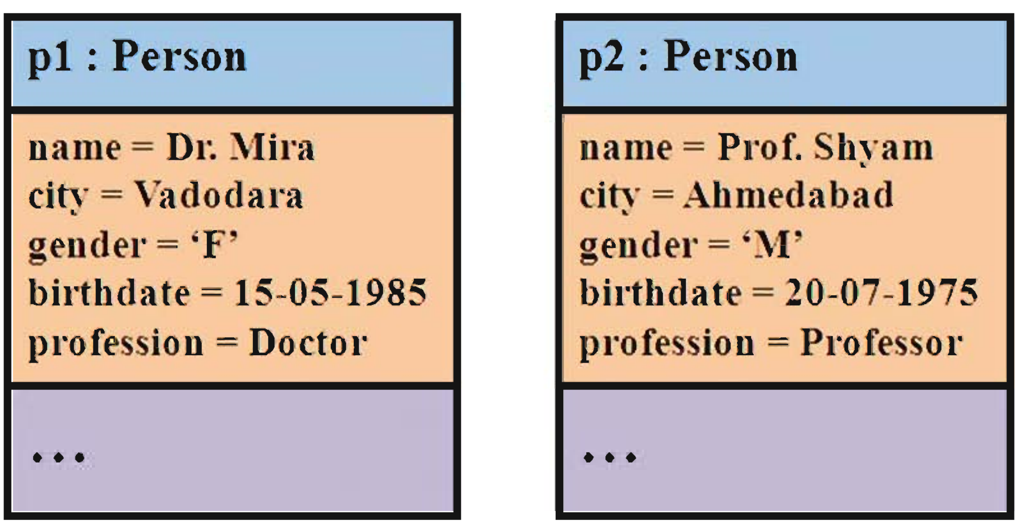
આકૃતિ 6.2માં દર્શાવ્યા પ્રમાણે, UMLની સંકેતલિપિમાં કોઈ કાર્યને નીચે જણાવેલી વાક્યરચના વાપરીને જણાવી શકાય છે :

[<visibility>] <method name> (parameter list separated by comma) : <return data type>

અગાઉના ઉદાહરણમાં મૅથડ (method)નું ઉદાહરણ setBirthdate(d:int, m:int, y:int) : date છે. અહીં પ્રાચલ ડેટાટાઇપ અને વૈકલ્પિક પ્રારંભિક કિંમત સાથે જણાવી શકાય છે; ઉદાહરણ તરીકે gender : char = 'M'. પ્રાચલ ફક્ત વાંચી જ શકાય (read only) તેવા છે કે નહીં, તેના આધારે તે નિવેશ અથવા નિર્ગમ તરીકે જણાવી શકાય છે.

UML રેખાકૃતિ વિનિયોગ કઈ પ્રોગ્રામિંગ ભાષામાં કોડ કરવાનો છે, તેના ઉપર આધારિત નથી. કેટલીક સૉફ્ટવેર બનાવનાર વ્યક્તિઓ ઍટ્રિબ્યૂટ અને કાર્યોને પ્રમાણભૂત UML સંકેતલિપિમાં જણાવવાને બદલે પ્રોગ્રામિંગ ભાષાના વધુ પરિચિત માળખામાં જણાવવાનું પસંદ કરે છે. આ પદ્ધતિ જ્યાં સુધી દરેક સંબંધિત વ્યક્તિને માટે અર્થપૂર્ણ રહે છે, ત્યાં સુધી તે બરાબર છે.

વિનિયોગના અમલ દરમિયાન ઑબ્જેક્ટને તેના સ્ટેટ (state) વડે રજૂ કરવામાં આવે છે. આ રીતે ઑબ્જેક્ટ ચલિત (dynamic) હોય છે. આકૃતિ 6.1ને અનુરૂપ 'person' ક્લાસના ઑબ્જેક્ટ (ઇન્સ્ટન્સ - instance તરીકે પણ ઓળખવામાં આવે છે)ને આકૃતિ 6.4માં દર્શાવ્યા પ્રમાણે રજૂ થાય છે.



આકૃતિ 6.4 : 'Person' ક્લાસના p1 અને p2 ઑબ્જેક્ટની રેખાકૃતિ

ઈનકૅપ્સ્યુલેશન (Encapsulation)

કોઈ પણ કમ્પ્યૂટર પ્રોગ્રામનાં બે મૂળ અંગ ડેટા અને કાર્ય છે. સંગઠિત પ્રોગ્રામિંગ (Structured Programming) આ બે ઘટકોને અલગ-અલગ ઘટક તરીકે ગણે છે જ્યારે ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ તેને એક ઘટક તરીકે જુએ છે.

પ્રક્રિયાગત પ્રોગ્રામિંગમાં (Procedural Programming) પ્રોગ્રામના કોઈ પણ ભાગમાં ડેટાની કિંમત બદલી શકાય છે. તે ફેરફારથી સુરક્ષિત નથી. આ સમસ્યાનો ઉકેલ ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગમાં ઈનકૅપ્સ્યુલેશન દ્વારા લાવી શકાય છે. અહીં ડેટા અને મેથડ વડે માહિતીની ગણતરી થાય તે સમયે પ્રોગ્રામના અન્ય ઘટકો વડે ડેટામાં ફેરફાર અથવા દુરુપયોગ સામે સુરક્ષિતતા બક્ષવામાં આવે છે. આ પ્રકારની રચના કે જેના વડે ડેટા અને મેથડ સામે રક્ષણ પૂરું પાડવામાં આવે છે તેને **ઈનકૅપ્સ્યુલેશન (Encapsulation)** કહેવામાં આવે છે. ડેટા અને મેથડને લપેટીને (વીંટીને) બનાવેલા એક એકમ જે ક્લાસના નામથી જાણીતા છે તેને અંગત (private) ઘોષિત કરવાથી શક્ય બને છે, અને ક્લાસના અંગત સભ્ય (private member) બહારના વિશ્વને સીધા ઉપલબ્ધ થવા દેતા નથી. જો તેની જરૂર હોય, તો સાર્વજનિક મેથડ (public method) વડે ડેટાને ઉપલબ્ધ બનાવી શકાય છે. આ રીતે, ઈનકૅપ્સ્યુલેશન ડેટાને છુપાવવાની ક્ષમતા પૂરી પાડે છે. આપણે આ પછીનાં પ્રકરણોમાં ક્લાસ અને અંગત/સાર્વજનિક ડેટા અને મેથડ વિશે અભ્યાસ કરીશું.

ઈનકૅપ્સ્યુલેશન બિનઈરાદાપૂર્વકની ક્રિયાઓ અને બહારના અન્ય ઑબ્જેક્ટ વડે જરૂર વગરના ડેટાને મેળવવાથી સલામત રાખે છે. પ્રક્રિયાગત પ્રોગ્રામિંગમાં સાર્વજનિક ડેટાવિસ્તાર માહિતીની વહેંચણી માટે સામાન્ય રીતે વપરાય છે, જ્યારે ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ તેનાથી વિરુદ્ધ અન્ય પ્રોગ્રામ્સ વડે સાર્વજનિક ડેટાને (વૈશ્વિક ચલ સિવાયના ડેટાના ઉપયોગને) સીધો મેળવવાની ક્રિયાને અટકાવે છે. ઑબ્જેક્ટ પોતાના ડેટાની કિંમતમાં જ ફેરફાર કરી શકે છે. અન્ય ઑબ્જેક્ટ તેને જોઈ શકે છે અથવા તે ઑબ્જેક્ટના માલિક (owner)ને સંદેશો મોકલીને આ ડેટાને બદલી શકે છે.

ડેટા-ઍબ્સ્ટ્રૅક્શન (Data-Abstraction)

ડેટા-ઍબ્સ્ટ્રૅક્શન એ ઑબ્જેક્ટની જરૂરી લાક્ષણિકતાઓને તેના અમલીકરણની વિગત સિવાયની માહિતીને રજૂ કરવાની પ્રક્રિયા છે. ઍબ્સ્ટ્રૅક્શન એ એક ખ્યાલ છે જે આંટીઘૂંટી કે ગૂંચને છુપાવે છે અને તે જે કાર્ય કરે છે, તે જણાવે છે પણ કઈ રીતે કરે છે, તે જણાવતો નથી.

હવે આપણે આપણી જિંદગીનું એક વાસ્તવિક ઉદાહરણ ટેલિવિઝનનું લઈએ. આપણે ટેલિવિઝનને ચાલુ અને બંધ કરી શકીએ છીએ, ચેનલ બદલી શકીએ છીએ તેમજ આપણને અનુકૂળ અવાજનું માપ રાખી શકીએ છીએ. પણ આપણે તેની આંતરિક રચના જાણતા નથી કે તે હવા કે તાર દ્વારા કઈ રીતે સંકેતો મેળવે છે, તે સંકેતોનું કઈ રીતે રૂપાંતર કરે છે અને અંતમાં તેને પડદા ઉપર પ્રદર્શિત કરે છે. આ રીતે ટેલિવિઝનનું આંતરિક કાર્ય તેના બાહ્ય સેતુથી તદ્દન અલગ છે. આપણે તેની આંતરિક રચનાની સમજ કે જ્ઞાન વિના તેનો ઉપયોગ પાવર-બટન, ચેનલ બદલવાનાં બટન અને અવાજના વૉલ્યુમમાં ફેરફાર તેના બાહ્ય સેતુ વડે કરી શકીએ છીએ.

આ રીતે ડેટા-ઍબ્સ્ટ્રૅક્શન એ એક એવી કાર્યરીતિ (ટેક્નિક) છે જે તે વાપરવાના સેતુ અને અમલીકરણને અલગ કરવાના ખ્યાલ પર આધારિત છે. તે પ્રોગ્રામિંગમાં નવો ખ્યાલ નથી. અનેક વ્યક્તિઓ કદાચ આ બાબતની જાણ વગર અમુક અંશે તેનો ઉપયોગ અત્યાર સુધી કરે જ છે. ઉદાહરણ તરીકે, આપણે જ્યારે C પ્રોગ્રામિંગમાં sqrt(25) અને printf("Hello world") જેવા વિધેયનો ઉપયોગ કરીએ છીએ ત્યારે આપણે કદી પણ એ નથી વિચાર્યું કે તે કઈ રીતે કાર્ય કરતા હશે.

જરૂરી નિવેશ ડેટા પ્રાચલો સાથે ઉપયોગકર્તા નિર્મિત (user-defined) વિધેય પણ ડેટા-ઍબ્સ્ટ્રૅક્શન પૂરું પાડે છે. હવે આપણે સ્ટ્રક્ચર (structure)નો ઉપયોગ કરીને એક C પ્રોગ્રામ વડે 'date' પ્રકારના ડેટા જણાવીએ. અહીં ડેટા ત્રણ ઘટકોનો બનેલો છે : દિવસ, માસ અને વર્ષ. આપણે આ ઘટકો માટે ડેટાનો મૂળભૂત પૂર્ણાંક સંખ્યા પ્રકાર (primitive integer datatype)નો ઉપયોગ કરીએ. હવે 'dateDiff' નામનું વિધેય લો, જેમાં બે તારીખ પ્રાચલ તરીકે લેવામાં આવે છે અને આ બે તારીખ વચ્ચે કેટલા દિવસ છે તે કિંમત પરત આપે છે. આ વિધેયના ઉપયોગકર્તાએ જાણવાની જરૂર નથી કે બે તારીખના તફાવતની ગણતરી કઈ રીતે થાય છે. પણ ઉપયોગકર્તા તે વિધેયની ફક્ત સિગ્નેચર(signature), એટલે કે વિધેયનું નામ, પ્રાચલની સંખ્યા અને તેનો પ્રકાર તેમજ પરત મળતી કિંમતના પ્રાચલનો પ્રકાર જાણે તે જરૂરી છે. જો કોઈ કારણથી તફાવત શોધવાની પ્રક્રિયામાં ફેરફાર થાય, તો આ વિધેય વાપરનાર પ્રોગ્રામના કોઈ પણ ભાગમાં તેની અસર થતી નથી.

આ રીતે ડેટા-ઍબ્સ્ટ્રૅક્શન આપણને વાપરવા માટેની એક રૂપરેખા કે ઢાંચો (templates) પૂરો પાડે છે. સિસ્ટમ ડેટાનો સંગ્રહ, નિર્માણ અને જાળવણી કઈ રીતે થાય છે, તેની કેટલીક માહિતી ગુપ્ત રાખે છે. બહારની દુનિયાને જે કંઈ દૃશ્યમાન

છે, તે ડેટા પ્રકારની અમૂર્ત (abstract) રીતભાત (behaviour) છે; પણ તેનું અમલીકરણ કઈ રીતે થયેલું છે, તે ગોપનીય રહે છે અને આથી તેનો ઉપયોગ કરનાર વ્યક્તિને અસર કર્યા વગર જરૂરિયાત પ્રમાણે તેમાં ફેરફાર કરી શકાય છે.

C અને C++ ના ઍબ્સ્ટ્રૅક્ટ ડેટાટાઇપ્સ (Abstract Data Types - ADT) અથવા સ્ટ્રક્ચર (structures - struct) અને C++/જાવાના ક્લાસ એ ડેટા-ઍબ્સ્ટ્રૅક્શનનાં ઉદાહરણ છે. ADTમાં આપણે ડેટાનો પ્રકાર વ્યાખ્યાયિત કરીને તેના ઉપરની પ્રક્રિયાઓ ફક્ત જણાવીએ છીએ. આપણે તે પ્રક્રિયાઓનું અમલીકરણ કઈ રીતે થાય છે, તે બતાવતા નથી.

ડેટા ઇનકૅપ્સ્યુલેશન અને ડેટા-ઍબ્સ્ટ્રૅક્શનનો મૂળભૂત તફાવત : ઇનકૅપ્સ્યુલેશન પ્રોગ્રામની બહારથી ડેટા મેળવવાનું અટકાવીને (inaccessible) ડેટાને સુરક્ષિત કરે છે, જ્યારે ઍબ્સ્ટ્રૅક્શન પ્રક્રિયાના અમલીકરણની માહિતી ગુપ્ત રાખીને ડેટાની રજૂઆત વડે ડેટાને સુરક્ષિત બનાવે છે.

મૅસેજિંગ (Messaging)

ઑબ્જેક્ટ આધારિત પરિભાષામાં મેથડ (method) બોલાવવાની ક્રિયાને **મૅસેજ (message)** કહેવામાં આવે છે. ઇનકૅપ્સ્યુલેશનને કારણે બધી મૅથડકૉલનું નિયંત્રણ ઑબ્જેક્ટ વડે થાય છે, જે (ઑબ્જેક્ટ) મૅથડને ઓળખે છે.

જુદા-જુદા ક્લાસમાં સમાન નામ ધરાવતી એકસરખી મૅથડ હોઈ શકે. ઉદાહરણ તરીકે, 'date' ક્લાસમાં 'display' નામની મૅથડ છે જે 'date' ઑબ્જેક્ટને અમુક ચોક્કસ સ્વરૂપમાં પ્રદર્શિત કરે છે. ધારો કે 'time' અને 'person' નામના અન્ય ક્લાસ છે, અને સમય ચોક્કસ સ્વરૂપમાં પ્રદર્શિત કરવા અને વ્યક્તિની માહિતી પ્રદર્શિત કરવા માટે બંનેની એક જ નામની મૅથડ 'display' છે. જ્યારે પ્રોગ્રામમાં 'display' મૅથડ વાપરવામાં આવે છે, ત્યારે કઈ મૅથડનો અમલ કરવાનો છે તે કઈ રીતે જાણવું ? જે ઑબ્જેક્ટ વડે મૅથડ વાપરવામાં આવે તેની મદદથી તે નક્કી થાય છે. ઉદાહરણ તરીકે, જો 'person' ક્લાસના ઑબ્જેક્ટ વડે 'display' વપરાય, તો તે 'person' ક્લાસની 'display' મૅથડનો અમલ કરશે.

પોલિમૉર્ફિઝમ (Polymorphism)

પોલિમૉર્ફિઝમ એટલે 'અનેક સ્વરૂપ' કે 'બહુરૂપતા'. એક મૅથડ કે પ્રક્રિયાનાં જુદાં-જુદાં સ્વરૂપ (form) હોઈ શકે.

ધારો કે આપણે બે સંખ્યામાંથી મોટી સંખ્યા શોધવાનું વિધેય 'max' લખેલું છે. આ વિધેય પ્રાચલ તરીકે બે પૂર્ણાંક સંખ્યા લે છે અને મોટી પૂર્ણાંક કિંમત પરત આપે છે. હવે ધારો કે 'max' નામનું એક વધારે વિધેય આપણે ઉમેરવા ઇચ્છીએ છીએ, જે કોઈ ઍરે (array)માં સંગ્રહ કરેલી પૂર્ણાંક સંખ્યાઓમાંથી મહત્તમ સંખ્યા શોધે. આ વિધેય તે ઍરે અને તેના કદને પ્રાચલ તરીકે લેશે અને મહત્તમ પૂર્ણાંક સંખ્યા પરત આપશે. શું એક જ નામનાં એક કરતાં વધુ વિધેયો વ્યાખ્યાયિત કરવા શક્ય છે ? કેટલીક પ્રોગ્રામિંગ ભાષામાં આ પ્રશ્નનો જવાબ 'ના' છે, પણ ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગમાં જ્યાં સુધી મૅથડ તેની સિગ્નેચર (પ્રાચલની સંખ્યા અને પ્રકાર)થી અલગ છે, ત્યાં સુધી તે પ્રશ્નનો જવાબ 'હા' છે.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ એક જ ક્લાસમાં એક કરતાં વધારે મૅથડ કે જેનાં નામ સરખાં હોય પણ સિગ્નેચરથી અલગ હોય તેને વ્યાખ્યાયિત કરવાની મંજૂરી આપે છે. આ લાક્ષણિકતાને **ફંક્શન કે મૅથડ ઓવરલોડિંગ (function or method overloading)** કહેવામાં આવે છે.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ઑબ્જેક્ટ ઉપર પ્રક્રિયકો સાથેની પદાવલિ લખવાની પણ છૂટ આપે છે. ઉદાહરણ તરીકે, આપણે 'date1-date2' જેવી પદાવલિ પણ વાપરી શકીએ, જ્યાં બંને સંકાર્ય (operands) એ 'date' ક્લાસના ઑબ્જેક્ટ છે. અહીં '-' પ્રક્રિયા એ બે સંખ્યાની બાદબાકી કરવાની પ્રક્રિયા જેમકે n1-n2, કરતાં અલગ રીતે કરવામાં આવે છે. અહીં એ જ પ્રક્રિયાનો અર્થ સંકાર્ય (operands)ના ડેટા પ્રકારના આધારે અલગ કરવામાં આવે છે. આ પ્રકારની બહુરૂપતા (પોલિમૉર્ફિઝમ) **ઑપરેટર ઓવરલોડિંગ (operator overloading)**ને કારણે શક્ય બને છે.

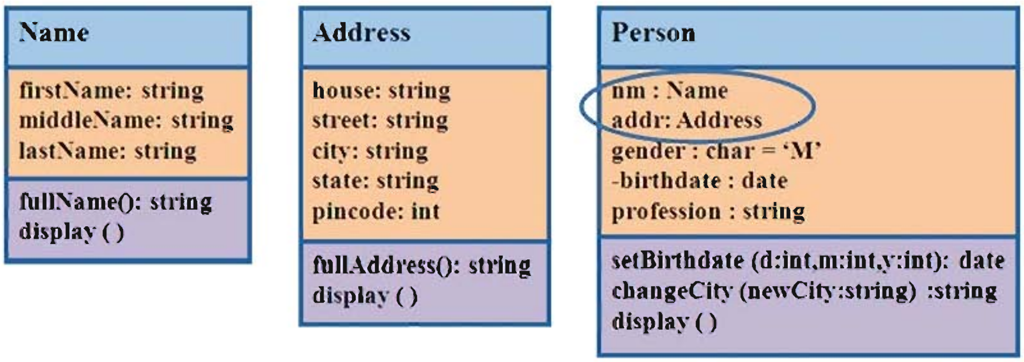
આ રીતે બે પ્રકારનાં ઓવરલોડિંગ વડે પોલિમૉર્ફિઝમ શક્ય બને છે : (1) ફંક્શન ઓવરલોડિંગ (function overloading) અને (2) ઑપરેટર ઓવરલોડિંગ (operator overloading). સામાન્ય રીતે, અલગ-અલગ સંદર્ભમાં એક જ નામના જુદા-જુદા અર્થની ક્ષમતા કે સામર્થ્યને ઓવરલોડિંગ (overloading) કહેવામાં આવે છે.

એગ્રિગેશન અને કૉમ્પોઝિશન (Aggregation and Composition)

જ્યારે એક ક્લાસના ઑબ્જેક્ટ બીજા ક્લાસના ઑબ્જેક્ટથી બનેલા હોય, ત્યારે તેને એગ્રિગેશન (aggregation) અથવા કોમ્પોઝિશન (composition) કહેવામાં આવે છે. તે અલગ-અલગ ક્લાસ વચ્ચેનો 'પૂર્ણ' અથવા 'તેમાંનો અંશ' સંબંધ રજૂ કરે છે. ઉદાહરણ તરીકે, મધરબોર્ડ કમ્પ્યૂટરનો એક ભાગ છે.

આપણે જાણીએ છીએ કે કમ્પ્યૂટર મધરબોર્ડ, સ્ક્રીન, કી-બોર્ડ અને માઉસ જેવા અનેક ઘટકોનું બનેલું હોય છે. સ્ક્રીન સ્વયં જ લંબાઈ, પહોળાઈ અને મૉડલ જેવા ગુણધર્મો (ઍટ્રિબ્યૂટ) સાથેનો એક ક્લાસ હોઈ શકે. એ જ રીતે મધરબોર્ડને પણ મોડલ અને કંપની જેવા ગુણધર્મો સાથેનો ક્લાસ બનાવી શકાય. જ્યારે આપણે 'computer' નામનો ક્લાસ વ્યાખ્યાયિત કરીએ, ત્યારે તેમાં 'motherboard' અને 'screen' ક્લાસના ઑબ્જેક્ટ જેવા ઍટ્રિબ્યૂટનો સમાવેશ કરેલો હશે. આ રીતે તેમાં સમાવેશનો નિર્દેશ સૂચવે છે.

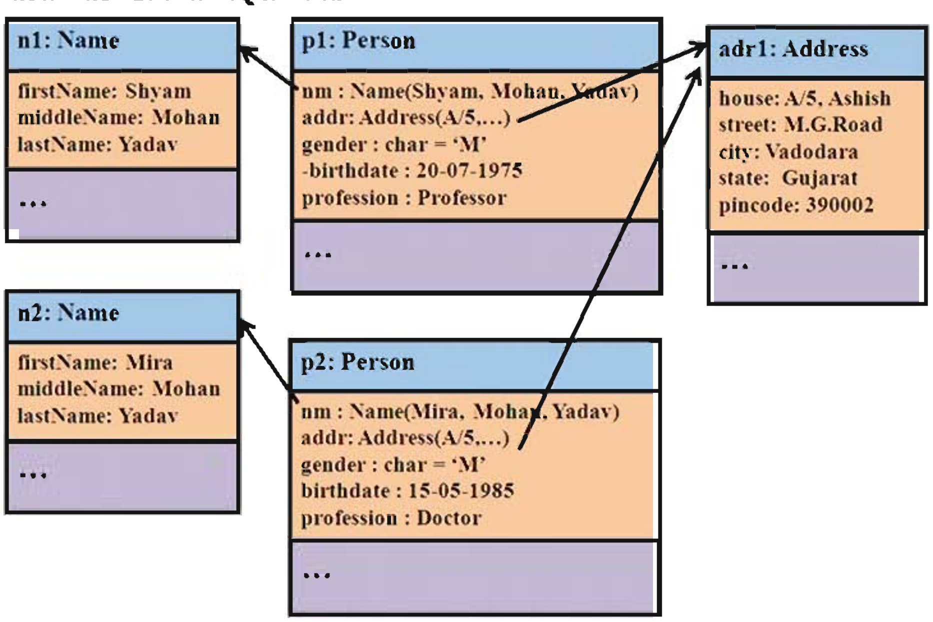
હવે આપણે અગાઉ ચર્ચા કરેલા 'Person' ક્લાસમાં ફેરફાર કરીએ. સૌપ્રથમ આકૃતિ 6.5માં દર્શાવ્યા પ્રમાણે આપણે બે નવા ક્લાસ 'Name' અને 'Address' બનાવીએ. આ 'Name' ક્લાસમાં firstname, middlename અને lastname ઍટ્રિબ્યૂટ છે અને 'Address' ક્લાસમાં house (જે ઘરની વિગતોનો નિર્દેશ કરે છે), street, city, state અને pin code ઍટ્રિબ્યૂટ છે. હવે 'Person' ક્લાસને બદલીએ અને ઍટ્રિબ્યૂટનાં નામ Name અને Addressને બદલી અનુક્રમે nm અને addr રાખીએ. nm અને addr ઍટ્રિબ્યૂટનો ડેટાપ્રકાર અનુક્રમે 'Name' અને 'Address' ક્લાસ છે. આ રીતે 'Person' ક્લાસમાં 'Name' અને 'Address' ક્લાસના ઑબ્જેક્ટ છે.



આકૃતિ 6.5 : 'Name' અને 'Address' ક્લાસના ઍટ્રિબ્યૂટ સાથેનો 'Person' ક્લાસ

ઍગ્રિગેશન અને કૉમ્પોઝિશનની તુલના (Aggregation Vs Composition)

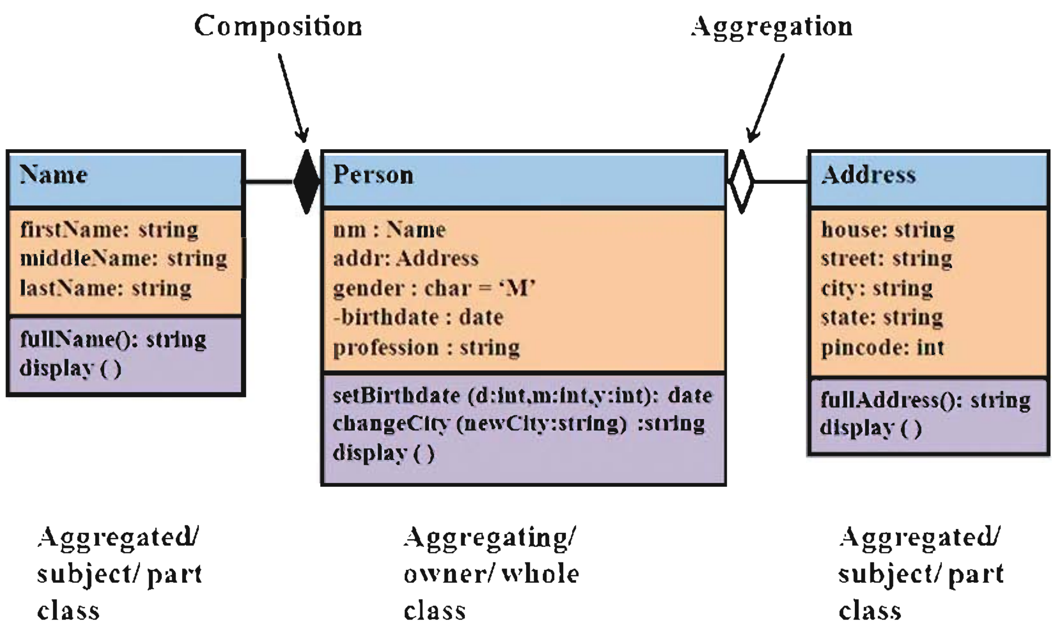
ઍગ્રિગેશન બે ક્લાસ વચ્ચેના ભિન્ન (non-exclusive) સંબંધને રજૂ કરે છે. ઍગ્રિગેશનમાં કોઈ ક્લાસ કે જે ઓનર ક્લાસનો એક ભાગ છે, તે સ્વતંત્ર રીતે અસ્તિત્વ ધરાવે છે. આ આંશિક ક્લાસ ઑબ્જેક્ટનો કાર્યકાળ ઓનર ક્લાસ (ownerclass)થી નક્કી થતો નથી. ઉદાહરણ તરીકે, મધરબોર્ડ જોકે કમ્પ્યૂટરનો એક ભાગ છે, પણ તે કમ્પ્યૂટરથી સ્વતંત્ર એક અલગ ઘટક તરીકે અસ્તિત્વ ધરાવે છે. આ રીતે મધરબોર્ડ કમ્પ્યૂટર સાથે અભિન્ન રીતે (exclusively) સંકળાયેલ નથી. આ જ રીતે આકૃતિ 6.6માં દર્શાવ્યા પ્રમાણે બે અથવા વધારે વ્યક્તિઓ address ઑબ્જેક્ટનો ઉપયોગ કરે છે. આથી, address એ કોઈ એક જ વ્યક્તિ માટે હોય તે જરૂરી નથી.



આકૃતિ 6.6 : અલગ-અલગ name ધરાવતા પણ એક જ address ધરાવતા બે 'Person' ઑબ્જેક્ટ

આકૃતિ 6.7માં દર્શાવ્યા પ્રમાણે મૂળભૂત એગ્રિગેશનને પૂર્ણ ક્લાસને અડીને એક ખાલી હીરાના ચિહ્નનો ઉપયોગ કરીને બનાવવામાં આવે છે.

કોમ્પોઝિશન બે ક્લાસ વચ્ચેનો અજોડ (exclusive) સંબંધ જણાવે છે. કોમ્પોઝિશન એક પ્રબળ કે મજબૂત પ્રકારનું એગ્રિગેશન છે, જેમાં આંશિક ક્લાસનો કાર્યકાળ ઓનર ક્લાસના અસ્તિત્વ આધારિત હોય છે. જો એકત્ર ક્લાસ (aggregating class)નો ઑબ્જેક્ટ દૂર કરવામાં આવે, તો તેના આંશિક ક્લાસ ઑબ્જેક્ટ પણ નષ્ટ થશે. ઉદાહરણ તરીકે, 'Person' ક્લાસનો કોઈ ઑબ્જેક્ટ ડિલીટ કરવામાં આવે, તો 'Name' ક્લાસનો ઑબ્જેક્ટ પણ નષ્ટ થશે. આકૃતિ 6.6માં દર્શાવ્યા પ્રમાણે Name અજોડ રીતે માત્ર એક જ વ્યક્તિ સાથે જોડાયેલું છે. કોમ્પોઝિશનનો સંબંધ આકૃતિ 6.7માં દર્શાવ્યા પ્રમાણે એક ભરાયેલા હીરાના ચિહ્નને પૂર્ણ ક્લાસને અડીને બતાવવામાં આવે છે.



આકૃતિ 6.7 : કોમ્પોઝિશન અને એગ્રિગેશન

ઉપર આપેલા ઉદાહરણમાં, 'Person' ક્લાસ અને 'Name' ક્લાસ વચ્ચેનો સંબંધ કોમ્પોઝિશન રિલેશનશિપ છે, જ્યારે 'Person' ક્લાસ અને 'Address' ક્લાસ વચ્ચેનો સંબંધ એગ્રિગેશન રિલેશનશિપ છે. એક જ ઍડ્રેસ એક કરતાં વધારે વ્યક્તિઓનું હોઈ શકે. આથી, જ્યારે કોઈ વ્યક્તિ નષ્ટ કરવામાં આવે છે, ત્યારે તેને સુસંગત 'Name' ઑબ્જેક્ટને નષ્ટ કરવામાં આવે છે, પણ 'Address' નષ્ટ કરવામાં આવતું નથી.

નોંધ : કોઈ ક્લાસ જ્યારે અન્ય ક્લાસના ઑબ્જેક્ટ ધરાવતા હોય ત્યારે તે ઓનર ક્લાસ (ownerclass) અથવા પૂર્ણ ક્લાસ (whole class) અથવા એકત્ર ક્લાસ (aggregating class) તરીકે ઓળખાય છે. ઉદાહરણ તરીકે, આકૃતિ 6.7માં દર્શાવેલ 'Person' ક્લાસ એગ્રિગેટિંગ ક્લાસનું ઉદાહરણ છે.

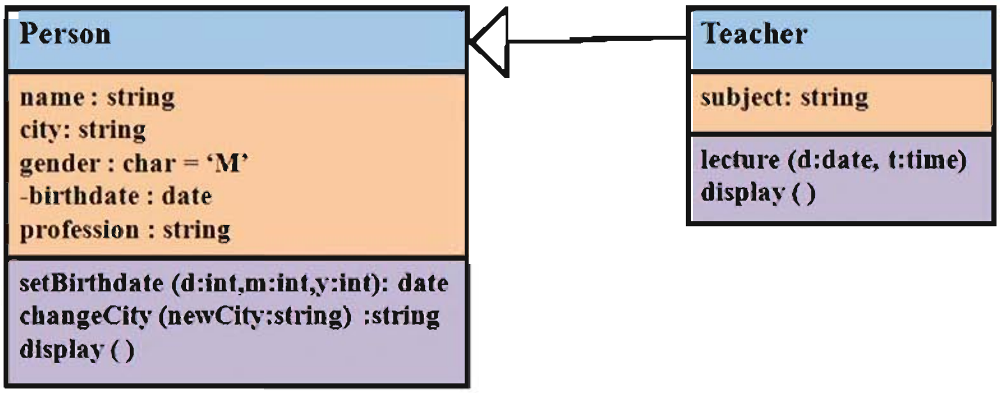
જે ક્લાસ ઓનરક્લાસમાં સમાયેલ છે, તેને સબ્જેક્ટ ક્લાસ (subject class) અથવા આંશિક ક્લાસ (part class) અથવા એકત્રિત ક્લાસ (aggregated class) તરીકે ઓળખાય છે. ઉદાહરણ તરીકે, આકૃતિ 6.7માં દર્શાવેલ 'Name' અને 'Address' ક્લાસ એગ્રિગેટેડ ક્લાસનાં ઉદાહરણ છે.

ઈનહેરિટન્સ (Inheritance)

ઈનહેરિટન્સ સામાન્ય રીતે બે ક્લાસ વચ્ચે 'એક પ્રકારનો' ('is-a-kind-of') સંબંધ જણાવે છે. જ્યારે એક ક્લાસ અન્ય ક્લાસનો પ્રકાર હોય ત્યારે આ યોગ્ય છે. ઉદાહરણ તરીકે, શિક્ષક એક પ્રકારની વ્યક્તિ છે. આથી, 'Person' ક્લાસના બધા ઍટ્રિબ્યૂટ અને મેથડ 'Teacher' ક્લાસને પણ લાગુ પડે છે. બીજા શબ્દોમાં કહીએ તો 'Person' ક્લાસના બધા ઍટ્રિબ્યૂટ અને બિહેવ્યર (ગુણધર્મ) 'Teacher' ક્લાસમાં વારસા (inherit)માં મળે છે. આ ઉપરાંત 'Teacher' ક્લાસના વધારાના ઍટ્રિબ્યૂટ જેવાં કે subject અને મેથડ જેમકે વિષયનાં lecture હોઈ શકે. આવા સંજોગોમાં 'Person' ક્લાસનો ઉપયોગ કરીને 'Teacher' ક્લાસ વ્યાખ્યાયિત કરી શકાય.

ઈનહેરિટન્સ અન્ય હયાત ક્લાસના ગુણધર્મોને વારસામાં મેળવીને ઑબ્જેક્ટના નવા ક્લાસને વ્યાખ્યાયિત કરવાના સામર્થ્યનો નિર્દેશ કરે છે. ઑબ્જેક્ટ આધારિત પરિભાષામાં નવા ક્લાસને સબક્લાસ (subclass) અથવા ચાઇલ્ડક્લાસ (childclass) અથવા ડિરાઇવ્ડ ક્લાસ (derived class) કહેવામાં આવે છે, જ્યારે હયાત ક્લાસને સુપર ક્લાસ (super class) અથવા પૅરેન્ટક્લાસ (parent class) અથવા બેઇઝ ક્લાસ (base class) કહેવામાં આવે છે. સુપર ક્લાસનાં ડેટા ઍટ્રિબ્યૂટ અને મેથડ ઑબ્જેક્ટનો સબક્લાસમાં તેનાં declarations ફરી લખ્યા વિના ઉપલબ્ધ હોય છે. જ્યાં હયાત મેથડ ફરી વ્યાખ્યાયિત કર્યા સિવાય વાપરવાની હોય ત્યાં આ ગુણધર્મ પુનઃઉપયોગી સુવિધા પૂરી પાડે છે. વધારામાં સબક્લાસમાં નવો ડેટા અને મેથડ સભ્યનો ઉમેરો વિસ્તૃત કરવા માટે કરી શકાય છે. સબક્લાસમાં જરૂર પ્રમાણે મેથડને ફરી વ્યાખાયિત કરવાની મંજૂરી આપે છે.

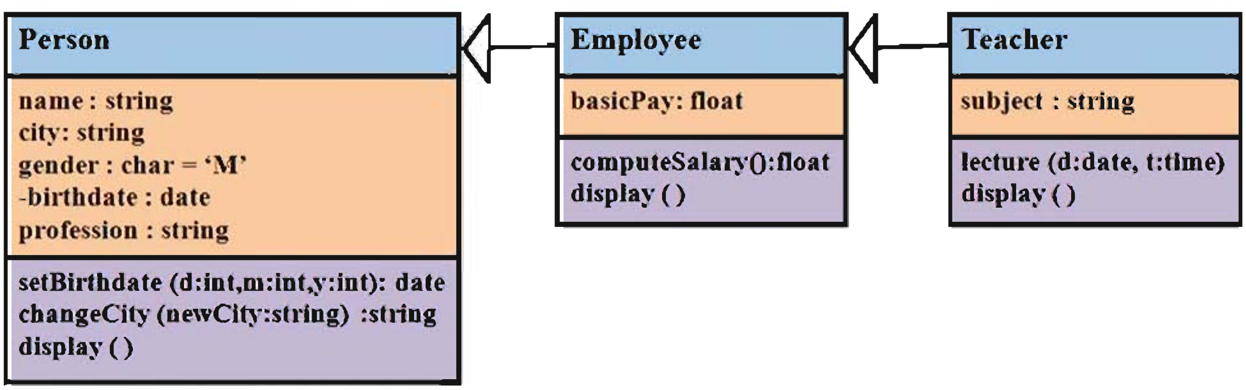
ક્લાસ ડાયાગ્રામમાં ઈનહેરિટન્સ બતાવવા માટે આકૃતિ 6.8માં દર્શાવ્યા પ્રમાણે સુપરક્લાસ તરફ નિર્દેશિત કરતા તીરનો ઉપયોગ કરવામાં આવે છે. ઉપરના ઉદાહરણમાં 'Person' એ સુપરક્લાસ છે અને 'Teacher' સબક્લાસ છે.



આકૃતિ 6.8 : ઈનહેરિટન્સનું ઉદાહરણ

સર્વસામાન્ય નિયમ એ ઈનહેરિટન્સનું બીજું નામ છે અથવા એક સંબંધ છે. બે ક્લાસમાંથી જ્યાં એક ક્લાસ બીજા ક્લાસની વિશિષ્ટ આવૃત્તિ છે, તેની વચ્ચેના સંબંધનો નિર્દેશ કરે છે. સુપરક્લાસમાં સામાન્ય ઍટ્રિબ્યૂટ અને મૅથડને વ્યાખ્યાયિત કરવામાં આવે છે. સબક્લાસ એ વધારાના ઍટ્રિબ્યૂટ અને મેથડ સાથેની વિશિષ્ટ આવૃત્તિ છે.

અમુક સમયે વિવિધ ક્લાસ વચ્ચે ઈનહેરિટન્સનો આદર્શ પદાનુક્રમ હોઈ શકે. ઉદાહરણ તરીકે, 'Person' ક્લાસમાંથી 'Employee' ક્લાસ મેળવી શકાય અને પછી 'Employee' ક્લાસમાંથી 'Teacher' ક્લાસ મેળવી શકાય. અહીં કર્મચારી એ એક વ્યક્તિ છે અને શિક્ષક એ એક કર્મચારી છે. આ પ્રકારના ઈનહેરિટન્સ **મલ્ટિલેવલ ઈનહેરિટન્સ (multilevel inheritance)** કહેવાય છે. આકૃતિ 6.9માં મલ્ટિલેવલ ઈનહેરિટન્સનું ઉદાહરણ આપેલું છે.



આકૃતિ 6.9 : પદાનુક્રમિત ઉત્તરાધિકાર (Multilevel inheritance)

ક્લાસને એક કરતા વધુ પેરન્ટ ક્લાસનો ઉપયોગ કરીને પણ તારવી શકાય છે. ઉદાહરણ તરીકે, બાળક વારસામાં તેના માતા અને પિતા બંનેની લાક્ષણિકતાઓ ધરાવે છે; એરપ્લેન એક પ્રકારનું વાહન છે તથા ઊડી શકે તેવો ઑબ્જેક્ટ પણ છે. જ્યારે કોઈ ક્લાસ બે કે વધુ ક્લાસ પરથી તારવવામાં આવે, તો તેને **મલ્ટિપલ ઈનહેરિટન્સ (multipal inheritance)** કહેવામાં આવે છે.

કૉમ્પોઝિશન અને ઇનહેરિટન્સની તુલના (Composition Vs Inheritance)

ઇનહેરિટન્સમાં ક્લાસની ક્રિયાત્મકતા (functionality) ભેગી વાપરવા, પુન:ઉપયોગ કરવા અથવા વધારવા માટે અન્ય ક્લાસમાંથી વારસામાં મેળવે છે. અહીં સુપર ક્લાસ અને સબક્લાસ વચ્ચે 'એક પ્રકારનો' ('a kind of') સંબંધ હોય છે.

કૉમ્પોઝિશનમાં ક્લાસ અન્ય ક્લાસમાંથી વારસામાં બનતો નથી, પણ બીજા ક્લાસ વડે 'બનેલો' ('composed of') હોય છે. ક્લાસ અમુક ઍટ્રિબ્યૂટ ધરાવે છે, જેમાંના કેટલાક ઍટ્રિબ્યુટ બીજા પ્રકારના ક્લાસના હોય છે.

અહીં નોંધ કરો કે ક્લાસ-ડાયાગ્રામમાં બીજી અન્ય પ્રકારની રિલેશનશિપ (relationship) અને કન્સ્ટ્રેઇન્ટ (constraints) પણ દર્શાવવામાં આવે છે. આ બધા ખ્યાલોનો અભ્યાસ આ પુસ્તકના કાર્યક્ષેત્રની મર્યાદા બહાર છે.

સારાંશ

સૉફ્ટવેર સિસ્ટમનાં વિશ્લેષણ, ડિઝાઇન અને અમલીકરણમાં ઑબ્જેક્ટ આધારિત પદ્ધતિ ઘણી નોંધપાત્ર ભૂમિકા ભજવે છે. આ ફેરફારમાં **ઑબ્જેક્ટ (object)** કેન્દ્રબિંદુ છે કે જેમાં ડેટા અને ક્રિયાત્મકતા બંનેનો સમાવેશ થયેલો હોય છે. ક્લાસ ઍટ્રિબ્યૂટ (ડેટા) અને મેથડ (બિહેવ્યર અથવા ક્રિયાત્મકતા)ને ભેગી કરીને (encapsulates) માળખા તરીકે (template) બધા ઑબ્જેક્ટ વચ્ચે સહિયારો ઉપયોગ કરે છે. ઑબ્જેક્ટ એકબીજાથી તેના સ્ટેટ (state)થી જુદા પડે છે, એટલે કે તેના ઍટ્રિબ્યૂટની કિંમતોથી એક કરતાં વધારે ક્લાસ એકબીજા સાથે જોડાયેલો હોય છે. જ્યારે બે ક્લાસ વચ્ચે 'પૂર્ણ' અથવા 'આંશિક' સંબંધની સ્થિતિ હોય છે, ત્યારે તેને એગ્રિગેશન અથવા કૉમ્પોઝિશન કહેવાય છે. જ્યારે એક ક્લાસ અન્ય ક્લાસના ઑબ્જેક્ટ ધરાવે, ત્યારે તે ધરાવનાર ક્લાસ ઑનર ક્લાસ (owner class) અથવા હોલક્લાસ (whole class) અથવા એગ્રિગેટિંગ ક્લાસ (aggregating class) કહેવાય છે. ઓનર ક્લાસમાં સમાયેલ ક્લાસ સબ્જેક્ટ ક્લાસ (subject class) અથવા પાર્ટક્લાસ (part class) અથવા એગ્રિગેટેડ ક્લાસ (aggregated class) કહેવાય છે. એગ્રિગેશન બે ક્લાસ વચ્ચે ભિન્ન (non-exclusive) સંબંધની સ્થિતિ રજૂ કરે છે. કૉમ્પોઝિશન બે ક્લાસ વચ્ચે અભિન્ન સંબંધની સ્થિતિ જણાવે છે. જ્યારે બે ક્લાસ વચ્ચે 'એક પ્રકારનો' સંબંધ હોય, ત્યારે તેને ઇનહેરિટન્સની સ્થિતિ કહેવામાં આવે છે. સામાન્ય લાક્ષણિકતાઓને સુપરક્લાસમાં રાખવામાં આવે છે અને વિશિષ્ટ લાક્ષણિકતાઓને સબક્લાસમાં રાખવામાં આવે છે.

સ્વાધ્યાય

1. ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ભાષામાં ઉપલબ્ધ લાક્ષણિકતાઓની યાદી બનાવો.
2. ક્લાસ અને ઑબ્જેક્ટનો તફાવત જણાવો.
3. 'ઇનકૅપ્સ્યુલેશન' અને 'ડેટા-ઍબ્સ્ટ્રૅક્શન' વચ્ચેની ભિન્નતા જણાવો.
4. પોલિમૉર્ફિઝમનો અર્થ શો છે ? પોલિમૉર્ફિઝમ મેળવવા માટે વપરાતા બે પ્રકારનાં ઓવરલોડિંગનાં નામ જણાવો.
5. એગ્રિગેશન અને કૉમ્પોઝિશનનો ઉપયોગ સમજાવો.
6. ઇનહેરિટન્સ ક્યારે વાપરવું જોઈએ ? ઉદાહરણ આપો.
7. વિવિધ પ્રકારના ઇનહેરિટન્સ સમજાવો.
8. **નીચેના પ્રશ્નોનો સાચો જવાબ પસંદ કરો :**

   (1) ઑબ્જેક્ટ આધારિત પદ્ધતિમાં કઈ વસ્તુ કેન્દ્રબિંદુ ઉપર હોય છે ?

   (a) ડેટા (b) વિધેય
   (c) ઑબ્જેક્ટ (d) ઉપરના બધા વિકલ્પ

(2) જાવા માટે નીચેનામાંથી શું વધારે યોગ્ય છે ?

(a) પ્રક્રિયાગત પ્રોગ્રામિંગ ભાષા

(b) ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ભાષા

(c) ક્વેરી-લૅંગ્વેજ

(d) ઉપરના બધા વિકલ્પ

(3) નીચેનામાંથી શું ઑબ્જેક્ટને એકબીજાથી જુદા પાડે છે ?

(a) એટ્રિબ્યૂટ (b) સ્ટેટ

(c) બિહેવ્યર (d) ઉપરના બધા વિકલ્પ

(4) એકસમાન ઑબ્જેક્ટની સામાન્ય લાક્ષણિકતાઓને વ્યાખ્યાયિત કરવા માટે નીચેનામાંથી શું વપરાય છે ?

(a) ક્લાસ (b) ઑબ્જેક્ટ

(c) મેથડ (d) ઉપરના બધા વિકલ્પ

(5) નીચેનામાંથી ક્યું દશ્યતાનું ચિહ્ન નથી ?

(a) ~ (b) * (c) # (d) -

(6) ઇનકેપ્સ્યુલેશન વડે નીચેનામાંથી શું પૂરું પાડવામાં આવે છે ?

(a) ડેટાની સુરક્ષિતતા (b) ડેટાનો સહિયારો ઉપયોગ

(c) ડેટા અને મેથડ છૂટા પાડવા (d) આપેલ બધા વિકલ્પ

(7) ડેટા-ઍબ્સ્ટ્રૅક્શન વડે શું શક્ય બનાવી શકાય છે ?

(a) ડેટા-સુરક્ષિતતા (b) ડેટા છુપાવવો

(c) ડેટાની ગણતરી કરવા બાબતની માહિતીનું અમલીકરણ છુપાવવું

(d) ઉપરના બધા વિકલ્પ

(8) નીચેનામાંથી ક્યો વિકલ્પ પોલિમૉર્ફિઝમ વડે પ્રાપ્ત થતો નથી ?

(a) મેથડ ઓવરલોડિંગ (b) ઑપરેટર ઓવરલોડિંગ

(c) ડેટા-હાઇડિંગ (d) આપેલ બધા વિકલ્પ

(9) એગ્રિગેશન ક્યા પ્રકારના સંબંધની સ્થિતિ જણાવે છે ?

(a) 'પૂર્ણ' સંબંધ ('is-a' relationship) (b) સમાન સંબંધ ('is-like' relationship)

(c) અંશતઃ સંબંધ (a-part-of' relationship) (d) ઉપરના બધા વિકલ્પ

(10) ઇનહેરિટન્સ ક્યા પ્રકારના સંબંધની સ્થિતિ જણાવે છે ?

(a) 'પૂર્ણ' સંબંધ ('is-a' relationship) (b) 'એક છે' સંબંધ ('has-a' relationship)

(c) અંશતઃ સંબંધ ('a-part-of' relationship) (d) ઉપરના બધા વિકલ્પ

(11) ક્લાસ-ડાયાગ્રામમાં કોમ્પોઝિશનને ક્યા ચિહ્ન વડે દર્શાવવામાં આવે છે ?

(a) ખાલી હીરાના ચિહ્ન વડે (b) ભરેલ હીરાના ચિહ્ન વડે

(c) ખાલી ત્રિકોણ ચિહ્ન વડે (d) ઉપરના બધા વિકલ્પ

જાવાની મૂળભૂત બાબતો 7

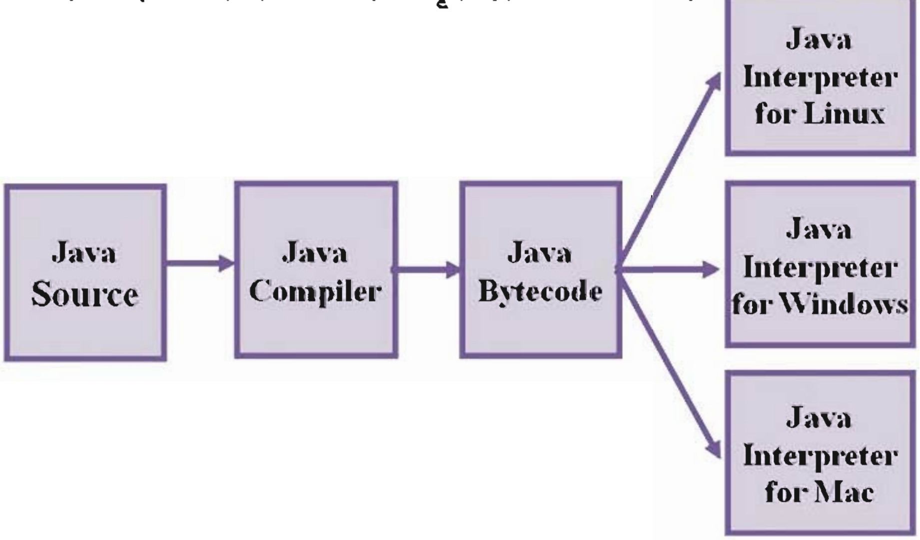
જાવા એક ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ભાષા છે જે ઉચ્ચ કક્ષાના યુનિક્સ વર્કસ્ટેશન માટે પ્રખ્યાત Sun Microsystems દ્વારા વિકસાવવામાં આવી છે. C++ પછી વિકાસ પામેલી જાવા ભાષા સૉર્સ પ્રોગ્રામ અને દ્વિઅંકી બંને સ્તરે વિવિધ પ્લેટફૉર્મ (જુદાં-જુદાં હાર્ડવેર) અને ઑપરેટિંગ સિસ્ટમ ઉપર નાની, સરળ અને સુવાહ્ય બની રહે તેવી બનાવેલી છે. આ પ્રકરણમાં આપણે જાવા સાથે પ્રોગ્રામિંગ શીખવાની શરૂઆત કરીશું. આપણે જાવામાં main() મેથડ વડે કરી શકાતાં પાયાનાં કાર્યો અને સાદાં જાવા વિધાનો બાબત ચર્ચા કરીશું. આપણે અગાઉ C ભાષા ભણી ગયા હોવાથી જાવાના પાયારૂપ બંધારણો સમજવા માટે ખૂબ જ સરળ બનશે. મોટાભાગનાં જાવાવિધાનોની વાક્યરચના (syntax) C ભાષા જેવી જ છે.

જાવાનો પરિચય (Introduction to Java)

જાવા ભાષા 1991માં Sun Microsystems દ્વારા વિકસાવવામાં આવી છે. જાવા નાની, ઝડપી, કાર્યક્ષમ અને હાર્ડવેર ઉપકરણોની વિશાળ શ્રેણી માટે સુવાહ્ય છે. તેને વર્લ્ડવાઇડ વેબ (World Wide Web) દ્વારા અમલમાં મૂકી શકાય તેવા (executable) પ્રોગ્રામ્સના વિતરણ માટેની એક આદર્શ ભાષા અને વિવિધ પ્લેટફૉર્મ ઉપર સરળતાથી વાપરી શકાય અને સુવાહ્ય હોય તેવા પ્રોગ્રામ્સ બનાવવા માટેની સામાન્ય ઉદ્દેશવાળી પ્રોગ્રામિંગ ભાષા (General Purpose Programming Language) પણ ગણવામાં આવે છે.

જાવા એક ઑબ્જેક્ટ આધારિત ભાષા છે અને અહીં તે C ભાષાથી અલગ પડે છે. જાવાનો ઉપયોગ કરીને આપણે ઑબ્જેક્ટ આધારિત પદ્ધતિ અને લવચિક, વિભાગીય (modular) અને ફરી વાપરી શકાય તેવા પ્રોગ્રામના કોડ લખવાના સામર્થ્યનો પૂરેપૂરો લાભ લઈ શકીએ છીએ. તે વિવિધ પ્રકારના ક્લાસની લાઇબ્રેરી ધરાવે છે, જે મૂળભૂત ડેટાપ્રકારો, સિસ્ટમની નિવેશ અને નિર્ગમ ઉપયોગી ક્ષમતા તેમજ અન્ય યુટિલિટી કાર્યો માટેની સગવડ પૂરી પાડે છે. આ મૂળભૂત ક્લાસનો સમૂહ જાવા ડેવલપમેન્ટ કિટ (JDK)નો ભાગ છે. JDK માં નેટવર્કિંગ માટેના, સામાન્ય ઇન્ટરનેટ પ્રોટોકોલના અને વપરાશકર્તાના સેતુ ટૂલકિટનાં કાર્યો માટેના અનેક ક્લાસ હોય છે. લાઇબ્રેરીના આ બધા ક્લાસ જાવામાં લખાયેલા હોવાથી બધા જાવા વિનિયોગની જેમ સમગ્ર પ્લેટફૉર્મ ઉપર તે સુવાહ્ય (portable) હોય છે.

જાવા સૉર્સપ્રોગ્રામ અને દ્વિઅંકી બંને સ્તરે પ્લેટફૉર્મ સ્વતંત્ર છે. પ્લેટફૉર્મ સ્વતંત્રતા એ પ્રોગ્રામની એક કમ્પ્યૂટર-સિસ્ટમથી બીજી ઉપર સરળતાથી લઈ જવાની ક્ષમતા છે. સૉર્સપ્રોગ્રામના સ્તરે જાવાના પ્રાથમિક ડેટા પ્રકારોનું કદ બધા વિકાસ પ્લેટફૉર્મ ઉપર અચળ છે. અને દ્વિઅંકી સ્તરે બાઇટકોડ ઇન્ટરપ્રિન્ટર (bytecode interpreter)ને કારણે પ્લેટફૉર્મ સ્વતંત્રતા શક્ય છે. જાવાના રચયિતા કમ્પાઇલેશન અને ઇન્ટરપ્રિટેશન (compilation and interpretation)ના મિશ્રણનો ઉપયોગ કરવાનું પસંદ કરે છે. પ્લેટફૉર્મ સ્વતંત્રતાની સ્થિતિ આકૃતિ 7.1માં દર્શાવેલ છે.



આકૃતિ 7.1 : જાવા : પ્લેટફૉર્મ સ્વતંત્ર

જાવામાં લખેલા પ્રોગ્રામનું એવા કમ્પ્યૂટરની મશીનની ભાષામાં કંપાઈલેશન (compilation) કરવામાં આવે છે, જે હકીકતમાં અસ્તિત્વમાં નથી. આ પ્રકારનાં "virtual" કમ્પ્યૂટર જાવા વર્ચ્યુઅલ મશીન (Java Virtual Machine - JVM) તરીકે ઓળખાય છે. જાવા વર્ચ્યુઅલ મશીનની યાંત્રિક ભાષાને જાવા બાઈટકોડ (bytecode) કહેવામાં આવે છે. દરેક પ્રકારના કમ્પ્યૂટર માટે અલગ-અલગ જાવા બાઈટકોડ ઈન્ટરપ્રિટરની જરૂર રહે છે. જાવા બાયનરી ફાઈલ્સ વાસ્તવમાં બાઈટકોડના સ્વરૂપમાં હોય છે જે કોઈ પણ એક પ્રોસેસર અથવા કોઈ પણ ઑપરેટિંગ સિસ્ટમ માટે ચોક્કસ નથી. બાઈટકોડ વાપરવાનો ગેરલાભ તેના અમલની ઓછી ઝડપ છે. એ પ્રકારનાં ટૂલ્સ ઉપલબ્ધ છે, જે જાવા બાઈટકોડનું નેટિવ કોડ (native code)માં રૂપાંતર કરે છે. નેટિવ કોડ અમલ કરવામાં ઝડપી છે પણ પછી તે મશીન સ્વંતત્ર રહેતાં નથી.

સાદા જાવા-વિનિયોગનું નિર્માણ (Creating Simple Java Application)

જાવાપ્રોગ્રામિંગ ભાષાની મૂળભૂત બાબતો શીખતાં પહેલાં, ચાલો આપણે એક સરળ પ્રોગ્રામ દ્વારા જાવાનું નિરીક્ષણ કરવાનું શરૂ કરીએ કે જે ફોનકૉલના ખર્ચની ગણતરી કરે અને પહેલાંથી ચૂકવેલી રકમ (pre-paid balance)માં સુધારો કરે. અહીં આપણે કઈ રીતે જાવાપ્રોગ્રામ બનાવવા, કંપાઈલ કરવા અને તેનો અમલ કરવો તે બાબત શીખીશું.

જાવાપ્રોગ્રામ અનેક ક્લાસનો બનેલો હોય છે. તેમાં ઓછામાં ઓછો એક ક્લાસ હોવો જોઈએ અને main મેથડ હોવી જ જોઈએ. C ભાષાના પ્રોગ્રામરો ક્લાસને struct અને typedefનો ઉપયોગ કરીને નવો સંયુક્ત ડેટાપ્રકાર (composite data type) બનાવવા જેવો વિચાર કરી શકે. જોકે ક્લાસ એ માત્ર માહિતીનો સંચય કરતા ઘણું વધારે આપી શકે છે. અહીં એ નોંધ કરો કે જાવામાં typedef ઉપલબ્ધ નથી.

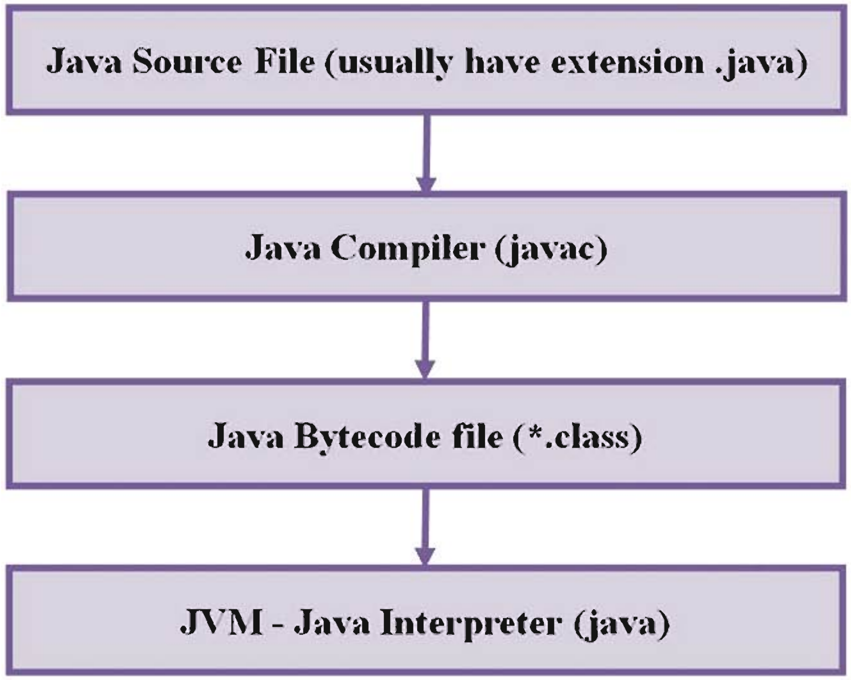
આપણે જાવા-વિનિયોગ કે જે ફોનકૉલના ખર્ચની ગણતરી કરી સિલક રકમમાં સુધારો કરે છે તેને 'CallCost' નામ આપીએ. આ 'CallCost' વિનિયોગ બનાવીને તેનો અમલ કરવા માટે આપણે નીચે જણાવેલાં પગલાં પ્રમાણે કાર્ય કરવું જરૂરી છે :

1. કોઈ પણ સાદા ASCII ટેક્સ્ટ-એડિટર વડે જાવા સૉર્સફાઈલ બનાવો.
   - કોઈ પણ ટેક્સ્ટ-એડિટર પસંદ કરી કોડલિસ્ટિંગ 7.1માં આપેલો પ્રોગ્રામ ટાઈપ કરો.
   - આ સૉર્સફાઈલનો 'CallCost.java' નામથી સંગ્રહ કરો. રૂઢિપરંપરાથી, જે નામથી ક્લાસ વ્યાખ્યાયિત કરેલો હોય, તે જ નામ અને '.java' અનુલંબન સાથેનું નામ જાવા સૉર્સફાઈલને આપવામાં આવે છે. અહીં એ નોંધો કે, ક્લાસનું નામ અને ફાઈલનું નામ કેસ સેન્સિટિવ હોય છે. માટે, જો ક્લાસનું નામ CallCost હોય તો, ફાઈલનું નામ પણ CallCost.java હોવું જરૂરી છે. (SciTE એડિટર માટે)
2. જાવા-કંપાઈલરનો ઉપયોગ કરીને સૉર્સફાઈલને કંપાઈલ (compile) કરો.
   - આ કાર્ય માટે javac પછી સૉર્સફાઈલનું નામ એટલે કે javac CallCost.java ટાઈપ કરીને જાવાપ્રોગ્રામ કંપાઈલ કરો.
   - જો કંપાઈલર કંઈ પણ ભૂલ દર્શાવે, તો પાછા જાઓ અને ખાતરી કરો કે કોડલિસ્ટિંગ 7.1માં જણાવ્યા પ્રમાણે તમે પ્રોગ્રામ બરાબર ટાઈપ કર્યો છે કે કેમ.
   - પ્રોગ્રામ કંઈ પણ ભૂલ વિના કંપાઈલ થાય, ત્યારે તમારી સૉર્સફાઈલની ડિરેક્ટરીમાં કંપાઈલર .class અનુલંબન સાથેની ફાઈલ બનાવશે. ખાતરી કરો કે કંપાઈલર વડે 'CallCost.class' ફાઈલ બનેલી છે. આ આપણી જાવા બાઈટકોડ ફાઈલ છે, જેનો અમલ કરવામાં આવશે.
3. જાવા-ઈન્ટરપ્રિટર વડે વિનિયોગનો અમલ કરો.
   - JDKમાં ફક્ત java આપીને જાવા ઈન્ટરપ્રિટર વાપરી શકાય છે. java CallCost ટાઈપ કરો.
   - પ્રોગ્રામનો અમલ કરવા માટે આપણે સૉર્સકોડની જરૂર નથી, પણ ફક્ત કંપાઈલ કરેલી ક્લાસફાઈલની જરૂર છે.
   - જાવા ઈન્ટરપ્રિટર બાઈટકોડ CallCost.classનો ઉપયોગ કરીને તેનો અમલ કરે છે.

```
/**
* This class implements a simple program that
* will compute the cost of phone call and update balance
*/
public class CallCost
{
public static void main(String[] args)
{
/* declare variables */
double balance;         // balance amount in rupees
double rate;            // call rate; rupees per second
double duration;        // call duration in seconds
double cost;            // cost of last call

/* computations. */
balance = 170;
rate = 1.02;
duration = 37;
cost = duration * rate;                 // compute the cost
balance = balance - cost;     // update balance amount

/* display results */
System.out.print("Call Duration: ");
System.out.print(duration);
System.out.println(" Seconds");
System.out.println("Balance: " + balance + "Rupees ");

} // end of main()
} // end of class CallCost
```

કોડલિસ્ટિંગ 7.1 : નમૂનારૂપ જાવાપ્રોગ્રામ

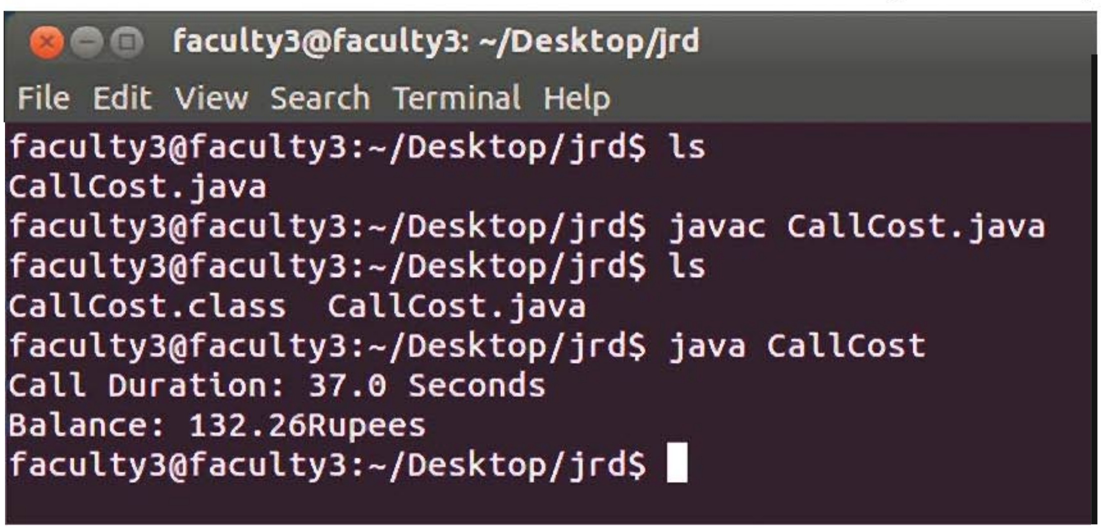
જાવાપ્રોગ્રામને કંપાઇલ કરવાની અને અમલ કરવાની પ્રક્રિયા આકૃતિ 7.2માં દર્શાવી છે.



આકૃતિ 7.2 : કંપાઇલેશન પ્રક્રિયા

જાવા સૉર્સપ્રોગ્રામ ફાઇલનું અનુલંબન '.java' હોવું જરૂરી છે. અને તે ફાઇલનું નામ જ્યારે ક્લાસ public હોય ત્યારે ક્લાસનું નામ હોવું જોઈએ. અહીં નોંધ કરો કે નામ કેસ-સૅન્સિટિવ (case sensitive) હોય છે. 'javac' નામનું કંપાઇલર સૉર્સફાઇલને કંપાઇલ કરી બાઇટકોડ ફાઇલ બનાવે છે. main મેથડ ધરાવતા ક્લાસનું નામ જ બાઇટકોડ ફાઇલનું હોય છે અને તેનું અનુલંબન '.class' હોય છે. 'java' ઇન્ટરપ્રિટર બાઇટકોડનું અર્થઘટન કરે છે અને તેનો અમલ કરે છે.

ટર્મિનલના ઉપયોગ દ્વારા linuxમાં પ્રોગ્રામને કંપાઇલ કરી તેનો અમલ કરી શકાય છે તે આકૃતિ 7.3માં દર્શાવ્યું છે.



આકૃતિ 7.3 : linux ટર્મિનલમાં જાવાપ્રોગ્રામનું કંપાઇલેશન અને અમલ

કોડલિસ્ટિંગ 7.1માં બે મુખ્ય ભાગ પર ધ્યાન આપો :

1. પ્રોગ્રામ ક્લાસની વ્યાખ્યામાં અહીં, 'CallCost' નામના ક્લાસમાં સમાયેલો છે.

2. પ્રોગ્રામની બૉડી (body) main() નામના રૂટિન (routine)માં સમાયેલી હોય છે. જાવાવિનિયોગમાં જ્યારે પ્રોગ્રામનો અમલ કરવામાં આવે છે, ત્યારે main() નામનું રૂટિન સૌપ્રથમ અમલમાં આવે છે.

કોડલિસ્ટિંગ 7.1ની સમજૂતી (Explanation of Code Listing 7.1)

પ્રોગ્રામમાં // પછી લખાયેલી શાબ્દિક માહિતી (text) અને /* અને */ વચ્ચે બંધ હોય તેને કૉમેન્ટ (comment) કહેવામાં આવે છે. આપણે જાણીએ છીએ કે કૉમેન્ટ કંપાઇલ કે ઇન્ટરપ્રિટ થતી નથી.

- ડેટાપ્રકાર (datatype) પછી ચલનું નામ (variable name) જણાવી ચલને ઘોષિત કરવામાં આવે છે.
- ગણતરીના ભાગમાં એસાઇન્મેન્ટ વિધાનો સાથેની પદાવલીઓ હોય છે.
- અહીં, પ્રોગ્રામની માહિતી ઉપયોગકર્તાને પ્રદર્શિત કરવા માટે અનેક સબરૂટિન (જાવામાં ફંક્શન અથવા મેથડ પણ કહેવામાં આવે છે.) કૉલ (subroutine call)વિધાનો વાપર્યાં છે.
  - પરિણામ પ્રદર્શિત કરવા માટે વાપરેલી મેથડનાં નામ : System.out.print અને System.out.println. આ બંને મેથડ જે કિંમત પ્રદર્શિત કરવાની છે, તે ચલ (argument) તરીકે લે છે.
  - System.out.println મેથડ જે માહિતી પ્રદર્શિત કરે છે, તેના પછી એક નવી લીટી (linefeed) ઉમેરે છે, જ્યારે System.out.print ફક્ત માહિતી જ પ્રદર્શિત કરે છે.
  - ફોનકૉલની અવધિ પ્રોગ્રામમાં ત્રણ Call વિધાન વડે પ્રદર્શિત થાય છે. પહેલાં Callથી શાબ્દિક માહિતી 'Call Duration:' પ્રદર્શિત થાય છે, બીજા Callથી 'duration' ચલની કિંમત પ્રદર્શિત થાય છે અને ત્રીજા Callથી 'Seconds' લખાણ પ્રદર્શિત થાય છે અને તે પછી કર્સર આગળની લીટીમાં લાવે છે. અહીં નોંધ કરો કે શાબ્દિક લખાણ (string literal) બેવડા અવતરણ (" ") વચ્ચે ઉમેરવામાં આવે છે.
  - એક Callથી balance પ્રદર્શિત થાય છે. પ્રાચલ તરીકે જાવામાં + પ્રક્રિયકનો ઉપયોગ ધ્યાન ઉપર લો. પ્રાચલ તરીકે આપેલી પદાવલીનું પ્રથમ મૂલ્યાંકન કરવામાં આવે છે અને પછી તે પ્રદર્શિત થાય છે.

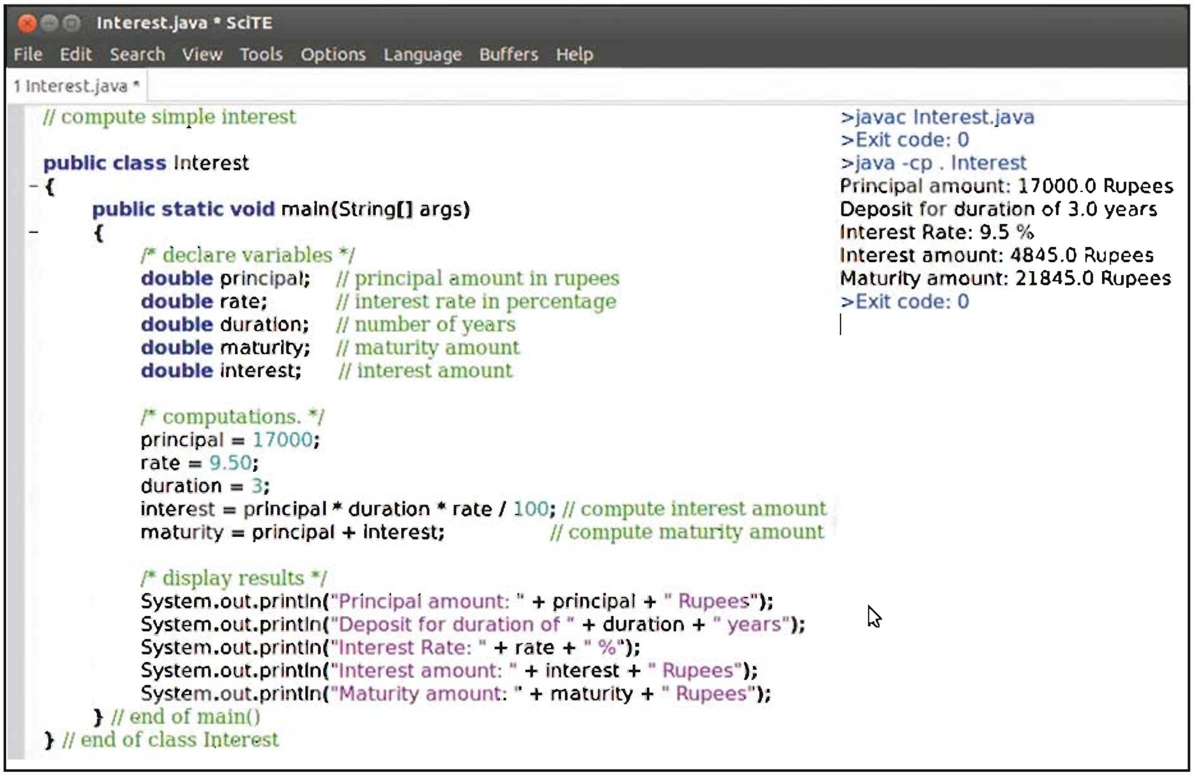
જ્યારે આપણે પ્રોગ્રામનો અમલ કરીએ છીએ, ત્યારે જાવા ઇન્ટરપ્રિટર પ્રથમ main() મેથડ ઉપર જાય છે અને તેમાં રહેલાં વિધાનોનો અમલ કરે છે. આ વિધાનો જ્યારે પ્રોગ્રામનો અમલ થાય ત્યારે કમ્પ્યૂટરે ચોક્કસ શું કરવાનું છે તે જણાવે છે. main() રૂટિન એ જ ક્લાસમાં અથવા અન્ય ક્લાસમાં પણ વ્યાખ્યાયિત સબરૂટિનને બોલાવી શકે છે, પણ ફક્ત main() રૂટિન જ કેવી રીતે અને ક્યા ક્રમમાં અન્ય સબરૂટિન વાપરવા તે નક્કી કરે છે.

main()ની પ્રથમ લીટીના શબ્દ "public"નો અર્થ એ છે કે આ પ્રોગ્રામની બહારથી આ રૂટિન બોલાવી શકાય છે. આ અનિવાર્ય છે, કારણકે જાવા ઇન્ટરપ્રિટર વડે main() રૂટિન બોલાવવામાં આવે છે કે જે પ્રોગ્રામની બહાર છે. રૂટિનની પ્રથમ લીટીની શેષ બાબત આ ક્ષણે સમજાવવી મુશ્કેલ છે; આથી હાલ પૂરતું એવું વિચારો કે તે વાક્યરચના (syntax)નો જરૂરી ભાગ છે.

SciTEનો ઉપયોગ (Using SciTE)

હવે આપણે SciTE એડિટર (editor)નો ઉપયોગ કરીને એક વધુ જાવા-વિનિયોગ બનાવીએ. અહીં આપણા વિનિયોગમાં સાદા વ્યાજની ગણતરી કરીને પરિણામ પ્રદર્શિત કરશે. આકૃતિ 7.4માં પ્રોગ્રામનું કોડલિસ્ટિંગ અને તેનો આઉટપુટ દર્શાવ્યો છે. નીચેનાં પગલાંને અનુસરો :

- SciTE વિનિયોગ ચાલુ કરો. હવે પસંદ કરો : **File → New.**
- આકૃતિ 7.4માં આપેલા કોડલિસ્ટિંગ પ્રમાણે જાવાપ્રોગ્રામ ટાઇપ કરો અને Interest.java નામ સાથે આ સૉર્સફાઇલનો સંગ્રહ કરો. ફાઇલનો સંગ્રહ કરવા માટે **File → Save** આદેશ આપો.
- **Tools → Compile** આદેશ આપીને સૉર્સપ્રોગ્રામને કંપાઇલ કરો.
- જો પ્રોગ્રામ કોઈ પણ ભૂલ (error) વિના કંપાઇલ થઈ જાય, તો **Tools → Go** આદેશ આપીને તેનો અમલ કરો.



```
Interest.java * SciTE
File Edit Search View Tools Options Language Buffers Help
1 Interest.java *

// compute simple interest

public class Interest
{
	public static void main(String[] args)
	{
		/* declare variables */
		double principal;	// principal amount in rupees
		double rate;	// interest rate in percentage
		double duration;	// number of years
		double maturity;	// maturity amount
		double interest;	// interest amount

		/* computations. */
		principal = 17000;
		rate = 9.50;
		duration = 3;
		interest = principal * duration * rate / 100; // compute interest amount
		maturity = principal + interest;		// compute maturity amount

		/* display results */
		System.out.println("Principal amount: " + principal + " Rupees");
		System.out.println("Deposit for duration of " + duration + " years");
		System.out.println("Interest Rate: " + rate + " %");
		System.out.println("Interest amount: " + interest + " Rupees");
		System.out.println("Maturity amount: " + maturity + " Rupees");
	} // end of main()
} // end of class Interest

>javac Interest.java
>Exit code: 0
>java -cp . Interest
Principal amount: 17000.0 Rupees
Deposit for duration of 3.0 years
Interest Rate: 9.5 %
Interest amount: 4845.0 Rupees
Maturity amount: 21845.0 Rupees
>Exit code: 0
```

આકૃતિ 7.4 : SciTEનો ઉપયોગ કરીને જાવાપ્રોગ્રામનો અમલ

જાવાપ્રોગ્રામનું બંધારણ (Structure of a Java Program)

પ્રોગ્રામિંગ ભાષાઓ સામાન્ય ભાષાઓથી અલગ એ રીતે છે કે તે સંપૂર્ણપણે સ્પષ્ટ (unambiguous) અને પ્રોગ્રામમાં કઈ બાબતને મંજૂરી છે અને કઈ બાબતને નથી, તે વિશે ખૂબ જ દૃઢ છે. એ નિયમો કે જે નક્કી કરે કે શું માન્ય છે, તેને ભાષાની વાક્યરચના (syntax) કહેવામાં આવે છે.

વાક્યરચનાના નિયમો (syntax rules) ભાષાના મૂળભૂત શબ્દભંડોળનો ઉલ્લેખ કરે છે અને ચલ, પદાવલીઓ, વિધાનો, બ્રાન્ચ (branches), લૂપ (loops) અને મેથડ (methods)નો ઉપયોગ કરીને કઈ રીતે પ્રોગ્રામની રચના કરી શકાય તે જણાવે છે. વાક્યરચનાની રીતે (syntactically) સાચો પ્રોગ્રામ એ છે કે જે સફળતાપૂર્વક કંપાઇલ કે ઇન્ટરપ્રિટ થઈ શકે.

જાવાપ્રોગ્રામનું માળખું આકૃતિ 7.5માં દર્શાવ્યું છે. અહીં, < અને > કોણીય કૌંસ એક પ્લેસહોલ્ડર (place holder) તરીકે વપરાય છે, જેમાં ખરેખર પ્રોગ્રામ લખતા સમયે કંઈક વાસ્તવિક ટાઇપ કરવાનું હોય છે. જાવામાં મેથડ (ફંક્શન)ની વ્યાખ્યા ફંક્શન હેડર તથા { અને } કૌંસ વચ્ચે વિધાનોની શ્રેણી હોય છે.

જાવા ઑબ્જેક્ટ આધારિત ભાષા હોવાથી અહીં તમામ ક્લાસના ભાગ તરીકે વ્યાખ્યાયિત કરવામાં આવે છે. આ રીતે, મેથડ સ્વતંત્ર રીતે હયાતી ધરાવતી નથી. તે ક્લાસનો એક ભાગ હોઈ શકે છે.

```
public class <class-name>
{
	<optional-variable-declarations-and-methods>
	public static void main(String[] args)
	{
		<statements >
	}
	<optional-variable-declarations-and-methods>
}
```

આકૃતિ 7.5 : જાવાપ્રોગ્રામનું માળખું

- પ્રથમ લીટીમાં <class-name> એ ક્લાસનું નામ છે, જેમાં main મેથડ હોય છે.

  જો ક્લાસનું નામ CallCost હોય તો રૂઢિપરંપરાથી જાવા સૉર્સફાઇલનો સંગ્રહ CallCost.java નામથી કરવો જોઈએ. જ્યારે આ ફાઇલને કંપાઇલ કરવામાં આવે છે, ત્યારે CallCost.class નામની બીજી ફાઇલ બને છે. ક્લાસના નામનો ઉપયોગ કરીને ક્લાસફાઇલ (class file)ને નામ આપવામાં આવે છે. આ CallCost.class નામની ક્લાસ ફાઇલ પ્રોગ્રામનું જાવા બાઇટકોડમાં થયેલું રૂપાંતર (અનુવાદ) ધરાવે છે જેનો જાવા-ઇન્ટરપ્રિટર દ્વારા અમલ કરી શકાય છે.

- main() મેથડ પહેલાં અને પછી ચલ (variable) અને મેથડ (method) ઘોષિત કરવા વૈકલ્પિક છે.

- દરેક પ્રોગ્રામમાં એક ક્લાસ હોવો જ જોઈએ જે પબ્લિક મેથડ main() ધરાવે.

- જાવા કોઈ ચોક્કસ બંધારણની ભાષા નથી, એટલે કે તે ફ્રી-ફૉર્મેટ (free-format) ભાષા છે. કોડલિસ્ટિંગ 7.1માં આપેલું પ્રોગ્રામનું માળખું (જેમ કે ખાલી લાઇન અને ઇન્ડેન્ટેશનનો ઉપયોગ) ભાષાની વાક્યરચના અથવા સિમેન્ટિક્સનો ભાગ નથી. કમ્પ્યૂટર પ્રોગ્રામના માળખાને ધ્યાનમાં લેતું નથી. આપણે આખો પ્રોગ્રામ એક જ લીટીમાં પણ લખી શકીએ. જોકે પ્રોગ્રામનું માળખું માનવીય વાચકો માટે મહત્ત્વનું છે.

- પ્રોગ્રામ main() સાથે બીજી મેથડ તેમજ અન્ય ચલ ધરાવી શકે છે. આપણે આ બાબત હવે પછી શીખીશું.

હવે, આપણે સાદાં જાવાવિધાનો વિશે શીખીએ જેના દ્વારા આપણે main() મેથડમાં સામાન્ય કાર્યો કરી શકીએ. આપણે વાક્યરચના (syntax) સાથે નીચે જણાવેલી બાબતો વિશે શીખીએ :

- ડેટાપ્રકાર (Data types)
- ચલ (Variables)
- લિટરલ (Literals)
- કૉમેન્ટ (Comments)
- જાવાવિધાનો અને પદાવલીઓ (Java statements and expressions)
- અંકગણિતીય પ્રક્રિયકો (Arithmetic operators)
- સરખામણી (Comparisons)
- તાર્કિક પ્રક્રિયકો (Logical operators)

ડેટાપ્રકારો (Data-types)

ડેટાપ્રકાર મેમરીનું જરૂરી કદ, મેમરીમાં રહેલી કિંમતનો પ્રકાર, કિંમતનો ક્ષેત્રવિસ્તાર (રેન્જ) અને તેની ઉપર શક્ય ક્રિયાઓનો પ્રકાર નક્કી કરે છે. જાવા આઠ પ્રકારના પ્રાથમિક (પ્રિમિટિવ-primitive) ડેટાપ્રકાર પૂરા પાડે છે, જે સામાન્ય પ્રકારના પૂર્ણાંકો, અપૂર્ણાંકો, કૅરેક્ટર (character-અક્ષર) અને બુલિયન કિંમતો (true અથવા false)ને સંભાળે છે.

પ્રાથમિક ડેટાપ્રકારોમાં **byte, short, int, long, float, double, char અને boolean**નો સમાવેશ થાય છે. પ્રથમ ચાર પ્રકારોમાં પૂર્ણાંક સંખ્યા (જેમકે 17, -38477 અને 0) હોય છે, તે પછીના બે પ્રકારોમાં અપૂર્ણાંક સંખ્યા (જેમકે 5.8, -129.35)નો સમાવેશ કરી શકાય છે, char પ્રકારમાં યુનિકોડ અક્ષરસમૂહ (Unicode character set)માંથી એક અક્ષર હોય છે અને બુલિયન (boolean) પ્રકાર બે તાર્કિક કિંમતો true અથવા falseમાંથી એક કિંમત હોય છે.

આ ડેટાપ્રકારોને પ્રાથમિક ડેટાપ્રકાર કહેવામાં આવે છે, કારણકે તે સિસ્ટમમાં જ સમાવિષ્ટ હોય છે. અહીં નોંધ કરશો કે આ ડેટાપ્રકાર મશીન સ્વતંત્ર છે, એટલે કે બધાં મશીન ઉપર તમામ જાવાપ્રોગ્રામ્સમાં તેનાં કદ અને લાક્ષણિકતાઓ સુસંગત (consistent) હોવાથી આપણે તેના ઉપર ભરોસો રાખી શકીએ છીએ. કોષ્ટક 7.1માં ડેટાપ્રકારોની વિગત આપેલી છે.

ડેટાપ્રકાર	સંગ્રહની જગ્યા	કિંમતનો પ્રકાર	કિંમતનો ક્ષેત્ર વિસ્તાર (range)	પૂર્વનિર્ધારિત કિંમત
byte	1 byte	Integer	-128 and 127	0
short	2 bytes	Integer	-32768 to 32767	0
int	4 bytes	Integer	-2147483648 to 2147483647	0
long	8 bytes	Integer	-9223372036854775808 to 9223372036854775807	0
float	4 bytes	Real	10 ±38 with about 7 significant digits	0
double	8 bytes	Real	10 ±308 with about 15 significant digits	0
char	2 bytes	Character	16-bit Unicode character	0
boolean	1 byte	Boolean	true, false	false

કોષ્ટક 7.1 : જાવામાં ડેટાના પ્રકારો

b બીટની યથાર્થતા (precision) સાથેની પૂર્ણાંક સંખ્યામાં ($-2^{b-1}-1$ થી 2^{b-1})ની રેન્જમાં ચિહ્નિત કિંમતોનો સંગ્રહ કરી શકાય છે. જ્યારે તેની આગળ unsigned ચાવીરૂપ શબ્દ (keyword) હોય, ત્યારે તેની કિંમત (0 થી $2^{b}-1$)ની રેન્જમાં હોઈ શકે. જાવામાં અપૂર્ણાંક સંખ્યા IEEE 754 (અપૂર્ણાંક સંખ્યાઓ અને અંકગણિતીય પ્રક્રિયાને વ્યાખ્યાયિત કરવા માટેના આંતરરાષ્ટ્રીય ધોરણો) સુસંગત છે. જાવા યુનિકોડ અક્ષરસમૂહનો ઉપયોગ કરે છે. char ડેટાપ્રકારની 16 બીટની પ્રીસિઝન (precision) અને અચિહ્નિત (unsigned) હોય છે. આ કારણે જાવા અનેક વિવિધ ભાષાઓના અલગ-અલગ મૂળાક્ષરોના હજારો અક્ષરોનો ઉપયોગ કરી શકે છે. boolean ડેટાપ્રકાર એ સંખ્યા નથી અને તેને સંખ્યા તરીકે ગણી પણ ન શકાય.

ચલ (Variable)

જો પ્રોગ્રામના અમલ દરમિયાન તમે કમ્પ્યૂટર દ્વારા કંઈ પણ યાદ રાખવા માંગો છો તો તે કમ્પ્યૂટરની મેમરીમાં સંગૃહીત કરવાની જરૂર રહે છે. મેમરીમાં સંગ્રહ કરેલા ડેટા ઉપર પ્રોગ્રામ ગણતરી કરે છે. યાંત્રિક ભાષામાં (મશીન-લૅંગ્વેજ) ડેટાનો નિર્દેશ જે મેમરી જગ્યાએ સંગ્રહ કરેલો છે, તેના આંકડાકીય સ્થાનાંક વડે કરવામાં આવે છે. જાવા જેવી ઉચ્ચ-સ્તરીય ભાષામાં ડેટાનો નિર્દેશ કરવા માટે મેમરીની જગ્યાનાં આંકડાકીય સ્થાનાંકના બદલે નામનો ઉપયોગ કરવામાં આવે છે. પ્રોગ્રામરે ફક્ત નામ જ યાદ રાખવું પડે છે. મેમરીમાં સંગ્રહ કરેલા ડેટાનો નિર્દેશ કરવા માટે વાપરેલા નામને **ચલ (variable)** કહેવામાં આવે છે.

પ્રોગ્રામના અમલ દરમિયાન જુદા-જુદા સમયે ચલમાં અલગ-અલગ ડેટાકિંમત હોઈ શકે પણ તે હંમેશાં એક જ મેમરી-સ્થાનાંકનો નિર્દેશ કરે છે. જાવાપ્રોગ્રામમાં જો પહેલાં ચલ ઘોષિત કરેલો હોય (declared) તો જ વાપરી શકાય છે.

જાવામાં એક અથવા વધારે ચલ નીચે જણાવેલી વાક્યરચના પ્રમાણે ઘોષિતવિધાન (declaration statement) દ્વારા ઘોષિત કરી શકાય :

<type-name> {variable-names};

અહીં વાક્યરચનામાં ઉપયોગમાં લીધેલી પદ્ધતિ નીચે પ્રમાણે છે :

- કોણીય કૌંસ < > ઉપયોગકર્તા દ્વારા જણાવવાના ઘટકનો નિર્દેશ કરે છે.
- છગડિયા કૌંસ { } અલ્પવિરામથી અલગ કરેલા ઘટકોની યાદીનો નિર્દેશ કરે છે.

અહીં {variable-names} ચલનાં નામોની યાદીનો નિર્દેશ કરે છે. જ્યારે યાદીમાં એક કરતાં વધારે ઘટક હોય, ત્યારે અલ્પવિરામથી જુદી પાડવામાં આવે છે.

<type-name>ને ચલના ડેટાપ્રકાર સૂચવતા ચાવીરૂપ શબ્દ (keyword) વડે બદલવામાં આવે છે. તે ચલનું કદ નક્કી કરવા તેમાં ક્યા પ્રકારની કિંમતો તે ધરાવી શકે તેમજ તેના ઉપર કાર્યો કરી શકાય તે જણાવવા માટે વપરાય છે. જ્યારે કમ્પ્યૂટર ચલઘોષિત વિધાનનો (variable declaration statement) અમલ કરે છે, ત્યારે ચલ માટે અલગ મેમરી સુયોજિત (સેટ) કરે છે અને તે મેમરી સાથે ચલનું નામ જોડે છે.

ચલનાં કેટલાંક ઉદાહરણ નીચે આપેલાં છે :

int marks;

double amount, interest;

float rate;

char grade;

boolean isPass;

ચલનું નામ વ્યાખ્યાયિત કરવા માટે આપણે નીચે જણાવેલા કેટલાક નિયમોને અનુસરવાની જરૂર છે :

- ચલના નામની શરૂઆત કોઈ મૂળાક્ષર, અન્ડરસ્કોર (_) અથવા ડૉલરના ચિહ્ન ($)થી કરવી જોઈએ. પહેલા અક્ષર પછી તેમાં અંક, મૂળાક્ષર, $ અને અન્ડરસ્કોર હોઈ શકે.

  કેટલાક માન્ય ચલનાં નામ : birth_date, result, CallCost, top5students, date, amount$, $price

  કેટલાક અમાન્ય ચલનાં નામ : 4me, %discount, birth date
- ચલનાં નામમાં વચ્ચે ખાલી જગ્યા (space) માન્ય નથી. આથી, birth date એ માન્ય ચલનું નામ નથી.
- ચલ તરીકે જાવાનો આરક્ષિત શબ્દ (reserved word) ન હોઈ શકે. આરક્ષિત શબ્દનો જાવામાં વિશિષ્ટ ઉપયોગ હોય છે અને પ્રોગ્રામર દ્વારા અન્ય હેતુ માટે તેનો ઉપયોગ કરી શકાતો નથી. કેટલાક આરક્ષિત શબ્દનાં ઉદાહરણ class, public, static અને while છે.

ચલનું નામ આપવાની માર્ગદર્શિકા

- ચલનું નામ અર્થપૂર્ણ પસંદ કરો. જાવા ચલનાં નામ કોઈ પણ લંબાઈનાં હોઈ શકે છે.
- જ્યારે ચલનાં નામમાં વધારે શબ્દો હોય જેમકે 'balance amount' તો નીચે જણાવેલામાંથી કોઈ પણ એક પદ્ધતિને અનુસરો :
  - પહેલા શબ્દ સિવાયના દરેક શબ્દનો પહેલો અક્ષર કૅપિટલ રાખો. આ રીતને અમુક સમયે કૅમલકેસ (camel case) કહેવામાં આવે છે કારણકે નામની વચ્ચેના કૅપિટલ અક્ષરો ઊંટની ખૂંધ જેવા લાગે છે. ઉદાહરણ : balanceAmount, birthDate

- શબ્દોને અન્ડરસ્કોરથી છૂટા પાડો. ઉદાહરણ : balance_amount, birth_date

- યાદ રાખો કે જાવા કેસ-સેન્સિટિવ (case-sensitive) છે. આથી કેપિટલ અને સ્મોલ અક્ષરો જુદા ગણવામાં આવે છે. આથી ચલનાં નામ balance અને Balance જુદા છે.
- પ્રણાલિકાગત, ક્લાસ (class)નાં નામ કેપિટલ (upper case)થી શરૂ કરવામાં આવે છે, જ્યારે ચલનાં અને મેથડનાં નામ સ્મોલ (lower case)થી શરૂ કરવામાં આવે છે.

ચલને ઘોષિત કરવા માટેની સારી પ્રોગ્રામિંગની શૈલી નીચે મુજબ છે :

- ચલને ઘોષિત કરવાના વિધાનમાં ફક્ત એક જ ચલ આપો.
- દરેક ચલને ઘોષિત કરવા સાથે પ્રોગ્રામમાં તેના હેતુની સમજ માટે કૉમેન્ટનો સમાવેશ કરો.
- અગત્યના ચલ ફંક્શનની શરૂઆતમાં ઘોષિત કરો. જે ચલ ફંક્શનના સમગ્ર તર્ક માટે અગત્યના ન હોય, તેને તે પહેલી વખત વપરાય ત્યારે ઘોષિત કરો.

જાવામાં ત્રણ પ્રકારના ચલ હોય છે : (1) ઇન્સ્ટન્સ વેરિયેબલ (Instance variable) (2) ક્લાસ વેરિયેબલ (class variable) અને (3) લોકલ વેરિયેબલ (local variable). ફંક્શનમાં ઘોષિત કરેલા ફંક્શન પેરામીટર (વિધેય પ્રાચલો) અને વેરિયેબલ (ચલ) એ લોકલ વેરિયેબલ છે. આપણે ઇન્સ્ટન્સ વેરિયેબલ અને ક્લાસ વેરિયેબલ વિશે પછી અભ્યાસ કરીશું.

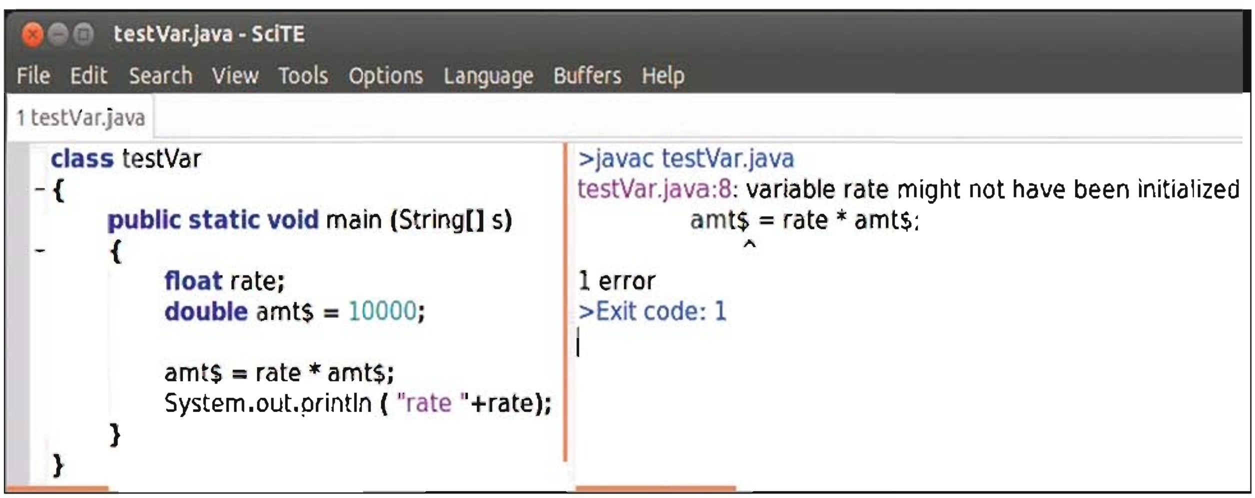
```
int marksObtained, totalMarks=100, counter=0;

boolean isPass = true;
```

સામાન્ય રીતે ચલને ઘોષિત કરવાની વાક્યરચના નીચે પ્રમાણે હોય છે :

<type name> {variable [= <value>,]};

અહીં ચોરસ કૌંસ [ ]માં દર્શાવેલ ઘટક વૈકલ્પિક હોય છે.

અહીં નોંધ કરો કે લોકલ વેરિયેબલને પૂર્વનિર્ધારિત કિંમત સાથે આરંભિક કિંમત આપી શકતા નથી. આવા ચલનો ઉપયોગ કરતાં પહેલા કોઈ કિંમત આપવાની જવાબદારી પ્રોગ્રામરની રહે છે. આકૃતિ 7.6માં જાવાપ્રોગ્રામમાં ચલનો ઉપયોગ દર્શાવ્યો છે.



આકૃતિ 7.6 : જાવાપ્રોગ્રામમાં લોકલ વેરિયેબલ

અહીં તમે જોઈ શકશો કે 'amt$ = rate * amt$' વિધાન વાપરતાં પહેલાં લોકલ વેરિયેબલ 'rate'ને કોઈ કિંમત આપી નથી. જ્યારે આપણે આ પ્રોગ્રામને કંપાઇલ કરીશું, ત્યારે આકૃતિ 7.6માં દર્શાવ્યા પ્રમાણે તે પ્રોગ્રામમાં ભૂલ (error) દર્શાવશે.

લિટરલ (Literals)

અચળ કિંમત માટે વાપરવામાં આવતું નામ લિટરલ તરીકે ઓળખાય છે. જાવામાં સંખ્યા (number), અક્ષર (character), સ્ટ્રિંગ (string - જે શાબ્દિક લખાણ છે) અને બુલિયન કિંમતો માટે અલગ-અલગ પ્રકારનાં લિટરલ હોય છે.

ન્યુમરિક લિટરલ (Numeric Literals)

પૂર્ણાંક અને અપૂર્ણાંક સંખ્યાઓ જણાવવા માટે ન્યુમરિક લિટરલ વપરાય છે. ઉદાહરણ : 157 અને 17.42 ન્યુમરિક લિટરલ છે.

ઇન્ટિજર લિટરલ (Integer literals) એ લિટરલ છે, જે પૂર્ણ સંખ્યા (whole number) છે. જાવામાં દશાંશ (base 10), અષ્ટાંકી (base-8) અને સોળ-અંકી (base-16) અને યુનિકોડ (unicode) ઇન્ટિજર લિટરલ છે.

સામાન્ય પૂર્ણાંક સંખ્યાઓ જેમકે 4, 157, 17777 અને -32 એ તેના કદ પ્રમાણે byte, short અથવા int ડેસિમલ ઇન્ટિજર લિટરલ છે. int કરતાં મોટા ડેસિમલ ઇન્ટિજર લિટરલ આપોઆપ long પ્રકારના બની જાય છે. આપણે નાની સંખ્યાને સંખ્યાની પાછળ L અથવા l લખીને long બનાવી શકીએ છીએ. (દા.ત. 4L એ long integer છે, જેની કિંમત 4 છે). ઋણ પૂર્ણાંક સંખ્યાઓ આગળ ઋણનું ચિહ્ન (-) લખવામાં આવે છે, દા.ત. -45.

અષ્ટાંકી સંખ્યા ફક્ત 0થી 7 અંકનો ઉપયોગ કરે છે. જાવામાં ન્યુમરિક લિટરલ આગળ 0 (શૂન્ય) લખવાથી તે સંખ્યા અષ્ટાંકી ગણવામાં આવે છે. ઉદાહરણ તરીકે, 045 લિટરલ એ અષ્ટાંકી પૂર્ણાંક છે, જેની દશાંશકિંમત 37 છે.

સોળ-અંકી સંખ્યા 16 અંકનો ઉપયોગ કરે છે, જેમાં 0થી 9 અંક અને અક્ષર A, B, C, D, E અને F હોય છે. આ સંદર્ભમાં કેપિટલ અક્ષર કે નાના અક્ષર અદલબદલ કરી શકાય છે. અક્ષર Aથી F અનુક્રમે 10 થી 15 સંખ્યા રજૂ કરે છે. જાવામાં સોળ-અંકી લિટરલ લખવા માટે સંખ્યાની શરૂઆત 0x અથવા OXથી થાય છે. સોળ-અંકી લિટરલનાં ઉદાહરણ 0x45 અથવા 0xFF7A છે. કેરેક્ટર લિટરલમાં સોળ-અંકી સંખ્યા પણ મનસ્વી (arbitrary) યુનિકોડ અક્ષર રજૂ કરવા વપરાય છે.

યુનિકોડ લિટરલ \u પછી ચાર સોળ-અંકી અંક વડે બનેલો હોય છે. ઉદાહરણ તરીકે, '\u00E9' કેરેક્ટર લિટરલ છે, જે તીવ્ર ઉચ્ચારણના સ્વરભાર સાથેનો "e" યુનિકોડ અક્ષર છે.

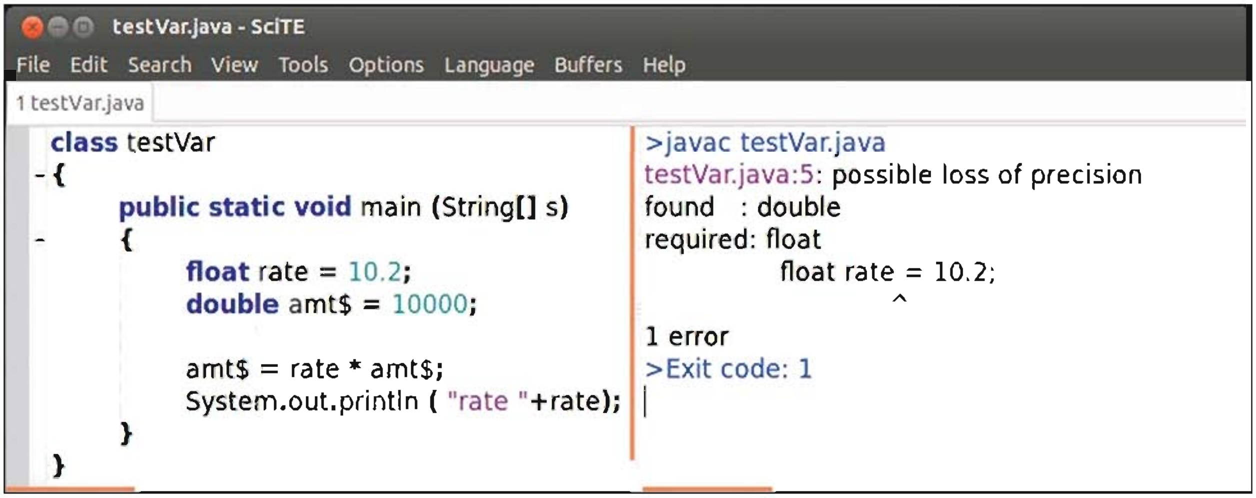
જાવા 7 દ્વિઅંકી સંખ્યા, જેમાં 0 અને 1 અંક વપરાય છે, ને રજૂ કરવા માટે સંખ્યાની પહેલા 0b (અથવા 0B) લખવામાં આવે છે. ઉદાહરણ : 0b10110 અથવા 0b101011001011.

રિયલ નંબર લિટરલને ફ્લોટિંગ-પૉઇન્ટ લિટરલ (Floating Point Literals) પણ કહેવામાં આવે છે. આ સંખ્યા બે પ્રકારની સંકેતલિપિ વડે લખી શકાય છે, પ્રમાણભૂત (standard) અને વૈજ્ઞાનિક (scientific).

પ્રમાણભૂત સંકેતલિપિમાં પૂર્ણાંક ભાગ અને અપૂર્ણાંક ભાગને દશાંશચિહ્ન (.)થી અલગ કરવામાં આવે છે. ઉદાહરણ તરીકે 12.37.

વૈજ્ઞાનિક સંકેતલિપિમાં સંખ્યા પછી અક્ષર e (અથવા E) અને ચિહ્નિત પૂર્ણાંક ઘાતાંક (Signed Integer Exponent) લખવામાં આવે છે. ઉદાહરણ તરીકે 1.3e12 અને 12.3737e-108. અહીં "e12" અને "e-108" એ 10નો ઘાતાંક રજૂ કરે છે. આથી, 1.3e12 એટલે 1.3 વખત10^{12} અને 12.3737e-108 એટલે 12.3737 વખત 10^{-108}. વૈજ્ઞાનિક માળખું અતિ મોટી સંખ્યા અને ખૂબ જ નાની સંખ્યા જણાવવા માટે વાપરી શકાય.

જાવામાં ફ્લોટિંગ-પૉઇન્ટ લિટરલનો પૂર્વનિર્ધારિત પ્રકાર double છે. લિટરલનો પ્રકાર float કરવા માટે સંખ્યાની પાછળ "F" અથવા "f" ઉમેરવો પડે છે. ઉદાહરણ તરીકે, "1.2F" નો અર્થ 1.2 લિટરલનો પ્રકાર float છે.



આકૃતિ 7.7 : કંપાઈલેશન એરર

આકૃતિ 7.7નું નિરીક્ષણ કરો જેમાં આપણે float પ્રકારના ચલને લિટરલ 10.2ની કિંમત આપેલી છે, જેથી આપણે કંપાઈલેશન એરર મેળવીએ છીએ. આ પ્રોગ્રામ સફળતાપૂર્વક ચલાવવા માટે આપણે એ વિધાનને "float rate = 10.2f" લખવું પડે. સંખ્યાના અંતમાં લખેલ પ્રત્યય F પહેલી કે બીજી એબીસીડીમાં લખી શકાય છે.

બુલિયન લિટરલ (Boolean Literals)

બુલિયન પ્રકાર માટે સ્પષ્ટ રીતે બે લિટરલ છે : true અને false. આ લિટરલ અવતરણ વિના ટાઈપ કરવામાં આવે છે. તે ચલ નથી પણ કિંમત છે. બુલિયન કિંમતો મોટે ભાગે શરતી પદાવલીઓમાં જોવા મળે છે. c ભાષામાં 0ને false તરીકે અને બિન-શૂન્ય કિંમતને true તરીકે ગણવામાં આવે છે. જાવામાં true અને false લિટરલ કોઈ આંકડાકીય કિંમત સાથે સંકળાયેલ નથી.

કૅરેક્ટર લિટરલ (Character Literals)

કૅરેક્ટર લિટરલને એક અવતરણચિહ્ન વચ્ચે સમાવાયેલા એક અક્ષર દ્વારા જણાવવામાં આવે છે. ઉદાહરણ : 'a', '#' અને '3'. અક્ષરને 16 બીટ યુનિકોડ કૅરેક્ટર તરીકે સંગ્રહ કરવામાં આવે છે. અમુક વિશિષ્ટ અક્ષરો માટે વિશેષ લિટરલ છે જે "escape character" તરીકે બૅક્સ્લેશ (\)નો ઉપયોગ કરે છે. કોષ્ટક 7.2માં વિશેષ કોડ કે જેના દ્વારા છાપી ન શકાય તેવા અક્ષરો તથા યુનિકોડ કૅરેક્ટર સેટમાં આવેલા અક્ષરોની યાદી આપવામાં આવી છે.

Escape code	Meaning
\n	New line
\t	Tab
\b	Backspace
\r	Carriage return
\f	Form feed (New page)
\\	Back slash character
\'	Single quote character
\"	Double quote character
\ddd	Character represented by three octal digits (d: 0 to 7)
\xdd	Character represented by two hexadecimal digits (d: 0 to 9, a to f)
\udddd	Character represented by Unicode number dddd (d: hexadecimal digit)

કોષ્ટક 7.2 : એસ્કેપકોડ

સ્ટ્રિંગ લિટરલ (String Literals)

જાવામાં અન્ય બધા ડેટાપ્રકારો "primitive" ડેટાને બદલે ઑબ્જેક્ટ રૂપમાં છે. આપણે હમણાં થોડા સમય માટે વિશે વિચારતા નથી. જોકે અહીં આપણે પૂર્વવ્યાખ્યાયિત ઑબ્જેક્ટ કે જે ઘણો અગત્યનો છે, તેના વિશે વિચારીશું, જે સ્ટ્રિંગ (string) પ્રકાર છે. સ્ટ્રિંગ એ અક્ષરોની એક શ્રેણી છે.

આપણે કોડલિસ્ટિંગ 7.1માં "Balance:" સ્ટ્રિંગ લિટરલ વાપર્યો છે. સ્ટ્રિંગ લિટરલ એ બે અવતરણચિહ્ન વચ્ચે અક્ષરોની એક શ્રેણી છે. સ્ટ્રિંગની અંદર વિશિષ્ટ અક્ષરો કોષ્ટક 7.2માં દર્શાવ્યા પ્રમાણે ઊંધા સ્લેશચિહ્ન વાપરીને રજૂ કરી શકાય છે.

ઉદાહરણ તરીકે, Many "Congratulations!" સ્ટ્રિંગ આપવા માટે આપણે સ્ટ્રિંગ લિટરલ "Many, \"Congratulations!\"" ટાઈપ કરવું પડે.

આ જ પ્રમાણે, "This string brought to you by Java\u2122" સ્ટ્રિંગમાં યુનિકોડની શ્રેણી \u2122 ટ્રેડમાર્કનું ચિહ્ન (™) બનાવશે.

કૉમેન્ટ (Comments)

પ્રોગ્રામમાં કોમેન્ટ (comment) ફક્ત માનવ-વાચકો માટે જ છે. તેની કમ્પ્યૂટર દ્વારા સંપૂર્ણપણે અવગણના થાય છે. દરેક વ્યક્તિ દ્વારા પ્રોગ્રામ સમજવો સરળ બનાવવા માટે કૉમેન્ટ લખવી અગત્યની છે.

જાવામાં કૉમેન્ટના નીચે લખેલ પ્રકાર હોય છે :

- **એક-લીટી કૉમેન્ટ :** તેની શરૂઆત ડબલસ્લેશ (//) થી થાય છે અને તે લીટીના અંત સુધી વિસ્તૃત થાય છે. કમ્પ્યૂટર // અને તેના પછી એ જ લીટીના તમામ લખાણની અવગણના કરે છે.
- **બહુ-લીટી કૉમેન્ટ :** તેની શરૂઆત /* થી અને અંત */ થી થાય છે. આ પ્રકારની કૉમેન્ટનો ઉપયોગ સામાન્ય રીતે એક કરતાં વધારે લીટીની કૉમેન્ટ હોય ત્યારે થાય છે. હકીકતમાં /* અને */ વચ્ચે શબ્દસમૂહ કે એક અથવા વધારે લીટીની કૉમેન્ટ કોઈ પણ રાખી શકે છે. બે સીમારેખાઓ /* અને */ વચ્ચેનું તમામ લખાણ કૉમેન્ટ ગણવામાં આવે છે. કૉમેન્ટનું નેસ્ટિંગ (nesting) કરી શકાતું નથી, એટલેકે એક કૉમેન્ટની અંદર બીજી કૉમેન્ટ ન હોઈ શકે.
- **દસ્તાવેજીકરણ કૉમેન્ટ :** આ પ્રકારની કૉમેન્ટ /** થી શરૂ થાય છે અને */ થી તેનો અંત આવે છે. તેનો ઉપયોગ કોડમાંથી API ડૉક્યુમેન્ટેશન બનાવવા માટે થાય છે. આ વિશિષ્ટ પ્રકારની કૉમેન્ટ છે, જે javadoc સિસ્ટમ માટે વપરાય છે. javadocની ચર્ચા આ પુસ્તકની મર્યાદાબહાર છે.

પ્રોગ્રામમાં કૉમેન્ટ સિવાય તમામે જાવાની વાક્યરચના (syntax)ને અનુસરવું પડે છે.

પદાવલી (Expressions)

પદાવલીઓ પ્રોગ્રામિંગના અનિવાર્ય ભાગ છે. પદાવલીના પાયાના ઘટકો લિટરલ (જેવા કે 674, 3.14, true અને 'X'), ચલ અને ફંક્શન કૉલ છે. યાદ રહે કે વિધેય એ એક સબરૂટિન છે, જે કિંમત પરત કરે છે.

સાદી પદાવલી એક લિટરલ, ચલ કે ફંક્શનકોલ હોઈ શકે. વધારે જટિલ પદાવલીઓ સાદી પદાવલીઓને પ્રક્રિયકો વાપરી ભેગા કરીને બનાવી શકાય છે.

પ્રક્રિયકોમાં (operators) ગાણિતિક પ્રક્રિયકો (+, -, *, /, %), સરખામણી-પ્રક્રિયકો (<, >, =, ...), તાર્કિક પ્રક્રિયકો (and, or, not...)નો સમાવેશ થાય છે. જ્યારે પદાવલીમાં થોડા પ્રક્રિયકો હોય, ત્યારે અગ્રતાક્રમ (precedence)નો પ્રશ્ન થાય છે કે ગણતરી કરવા માટે કઈ રીતે પ્રક્રિયકોનાં જૂથ બનાવવાં. ઉદાહરણ તરીકે, "A + B * C" પદાવલીમાં પહેલાં B*Cની ગણતરી કરવામાં આવે છે અને તેનું પરિણામ Aમાં ઉમેરવામાં આવે છે. આપણે અહીં સરવાળા (+) કરતાં ગુણાકાર (*) ને વધારે અગ્રતાક્રમ આપીએ છીએ. જો આપણે ઇચ્છતા હોય તે પૂર્વનિર્ધારિત અગ્રતાક્રમ ન હોય, તો આપણે જૂથ સ્પષ્ટ રીતે જણાવવા માટે કૌંસનો ઉપયોગ કરી શકીએ. ઉદાહરણ તરીકે,

જો આપણે "(A + B) * C" પદાવલીની કિંમત શોધવી હોય, તો પહેલાં Aને Bનો સરવાળો કરી તેના પરિણામનો C સાથે ગુણાકાર કરવામાં આવે.

પ્રક્રિયક (Operators)

પદાવલી બનાવવા માટે વપરાતાં વિશિષ્ટ ચિહ્નો પ્રક્રિયક છે. જાવા અનેક પ્રકારના પ્રક્રિયક પૂરા પાડે છે. આ પ્રકરણમાં આપણે નીચે જણાવેલા પ્રક્રિયકોની ચર્ચા કરીશું :

- અંકગણિતીય પ્રક્રિયક (Arithmetic operators)
- તુલનાત્મક પ્રક્રિયક (Comparison operators)
- તાર્કિક પ્રક્રિયક (Logical operators)
- શરતી પ્રક્રિયક (Conditional operator)
- એસાઇન્મેન્ટ પ્રક્રિયક (Assignment operator)

અંકગણિતીય પ્રક્રિયક (Arithmetic Operators)

જાવામાં મૂળભૂત અંકગણિતીય પ્રક્રિયક સરવાળો (+), બાદબાકી (–), ગુણાકાર (*), ભાગાકાર (/) અને મૉડ્યુલસ (%) છે. આ બધા પ્રક્રિયક બાયનરી છે, તેમાં બે સંકાર્ય (operand) હોય છે. પ્રક્રિયક + અને – યુનરી (unary) (ફક્ત એક જ સંકાર્ય સાથે) તરીકે પણ વપરાય છે. આ બધા જ પ્રક્રિયકો તમામ પ્રકારના આંકડાકીય ડેટા ઉપર લાગુ કરી શકાય છે, જેમકે byte, short, int, long, float અને double. તે char પ્રકારની કિંમત સાથે પણ વાપરી શકાય છે. આ સંદર્ભમાં તે પૂર્ણાંક સંખ્યા તરીકે ગણવામાં આવે છે. જ્યારે char પ્રકારના ડેટા હોય, ત્યારે તેની કિંમત unicode સંખ્યા પ્રમાણે ગણવામાં આવે છે, જ્યારે તે અંકગણિતીય પ્રક્રિયક સાથે વાપરવામાં આવે. કોષ્ટક 7.3માં અંકગણિતીય પ્રક્રિયકોનું વર્ણન આપેલું છે :

પ્રક્રિયક	અર્થ	ઉદાહરણ	પરિણામ
+	સરવાળો	2 + 8	10
-	બાદબાકી	2 - 8	-6
*	ગુણાકાર	2 * 8	16
/	ભાગાકાર	8 / 2	4
%	મૉડ્યુલસ (ભાગાકાર પછી શેષ જણાવે છે, ભાગફળ પૂર્ણાંક સંખ્યા હોય છે.)	8 % 3 25.8 % 7	2 4.8

કોષ્ટક 7.3 : અંકગણિતીય પ્રક્રિયકો

બાયનરી અંકગણિતીય પ્રક્રિયા પછી પરિણામનો ડેટાપ્રકાર નીચે પ્રમાણે હોય છે :

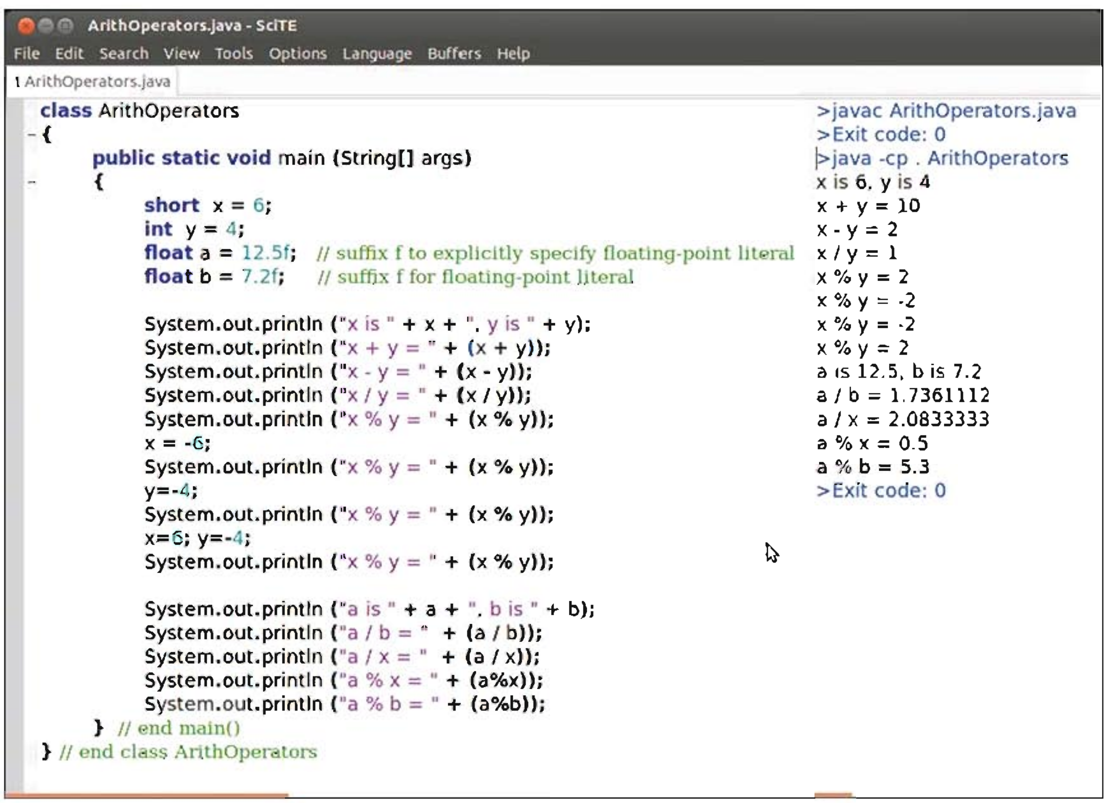
- જો બન્ને સંકાર્ય (operands) એક જ પ્રકારના ડેટા હોય, તો પરિણામનો ડેટાપ્રકાર પણ સંકાર્યના પ્રકાર સમાન જ રહેશે.
  - જો બન્ને સંકાર્ય પૂર્ણાંક હશે તો પરિણામ પૂર્ણાંક રહેશે. ઉદાહરણ તરીકે 9/2નું પરિણામ 4.5 નહીં, પણ 4 છે. પૂર્ણાંક સંખ્યાના ભાગાકારનું પરિણામ પણ પૂર્ણાંક કિંમત રહેશે અને શેષ છોડી દેવામાં આવે છે.
  - જો બંને સંકાર્ય અપૂર્ણાંક હોય, તો પરિણામ અપૂર્ણાંક રહેશે. ઉદાહરણ તરીકે, 9f/2f નું પરિણામ 4.5 છે.

- જ્યારે બન્ને સંકાર્ય અલગ-અલગ પ્રકારના ડેટા હોય, ત્યારે :
  - સૌપ્રથમ જે ડેટાપ્રકારની નાની રેન્જ છે, તેને મોટી રેન્જના ડેટા પ્રકારમાં બદલી નાખવામાં આવે છે, જેથી બન્ને સંકાર્યનો સમાન પ્રકાર બની રહે. આ પ્રકારના પરિવર્તનને પ્રમોશન (promotion) પણ કહેવામાં આવે છે.
  - હવે, પદાવલીનું પરિણામ મોટી રેન્જના સંકાર્ય સમાન બની રહેશે.
  - ઉદાહરણ : 4 + 3.5નું પરિણામ 7.5 મળશે. 9/2.0નું પરિણામ 4.5 મળશે અને 9f/2નું પરિણામ 4.5 મળશે.

યાદ રાખવા જેવા મુદ્દાઓ :

- મૉડ્યુલસ પ્રક્રિયક %ના કિસ્સામાં, જો પહેલો સંકાર્ય ઋણ હોય તો, પરિણામ ઋણ છે.
- જાવામાં % પ્રક્રિયક સાથે અપૂર્ણાંક ડેટાપ્રકાર પણ વાપરી શકીએ છીએ. પરિણામ પૂર્ણાંક ભાગફળ પછી રહેલી શેષ છે. આથી, 25.8 % 7ના જવાબ માટે પૂર્ણાંક ભાગફળ 3 છે અને શેષ 25.8 – (3*7) = 4.8 છે.
- પ્રક્રિયક +નો ઉપયોગ શાબ્દિક લખાણ (string)ને જોડવા (concatenate) માટે પણ કરી શકાય છે.

જ્યારે બે સંકાર્યમાંથી એકનો પ્રકાર string હોય અને તેના ઉપર + પ્રક્રિયક લાગુ કરવામાં આવે, ત્યારે બીજા સંકાર્યનો ડેટા-પ્રકાર આપોઆપ string બની જાય છે. આ પણ ગર્ભિત (implicit) પ્રકારનું રૂપાંતર છે. આ પ્રકારનું રૂપાંતર આપણે balance રકમ છાપતા સમયે '"Balance :" + balance' પદાવલીમાં કોડલિસ્ટિંગ 7.1માં જોઈ શકીએ છીએ. આકૃતિ 7.8માં દર્શાવેલા કોડલિસ્ટિંગમાં પણ તેનો ઉપયોગ કરેલો છે.




```
class ArithOperators
{
    public static void main (String[] args)
    {
        short  x = 6;
        int  y = 4;
        float a = 12.5f;   // suffix f to explicitly specify floating-point literal
        float b = 7.2f;    // suffix f for floating-point literal

        System.out.println ("x is " + x + ", y is " + y);
        System.out.println ("x + y = " + (x + y));
        System.out.println ("x - y = " + (x - y));
        System.out.println ("x / y = " + (x / y));
        System.out.println ("x % y = " + (x % y));
        x = -6;
        System.out.println ("x % y = " + (x % y));
        y=-4;
        System.out.println ("x % y = " + (x % y));
        x=6; y=-4;
        System.out.println ("x % y = " + (x % y));

        System.out.println ("a is " + a + ", b is " + b);
        System.out.println ("a / b = "  + (a / b));
        System.out.println ("a / x = "  + (a / x));
        System.out.println ("a % x = " + (a%x));
        System.out.println ("a % b = " + (a%b));
    } // end main()
} // end class ArithOperators
```

```
>javac ArithOperators.java
>Exit code: 0
>java -cp . ArithOperators
x is 6, y is 4
x + y = 10
x - y = 2
x / y = 1
x % y = 2
x % y = -2
x % y = -2
x % y = 2
a is 12.5, b is 7.2
a / b = 1.7361112
a / x = 2.0833333
a % x = 0.5
a % b = 5.3
>Exit code: 0
```

આકૃતિ 7.8 : અંકગણિતીય પ્રક્રિયકોનો ઉપયોગ દર્શાવતો જાવાપ્રોગ્રામ

આકૃતિ 7.8માં દર્શાવેલા પ્રોગ્રામની સમજૂતી (Explanation of Program Shown in Figure 7.8)

- આપણે main() મેથડની શરૂઆતમાં ચાર ચલ વ્યાખ્યાયિત કરીએ છીએ. જેમાં x અને y એ પૂર્ણાંક પ્રકારના (short અને int પ્રકાર) અને a તથા b અપૂર્ણાંક પ્રકારના (float પ્રકાર) છે.

- તમને યાદ હશે કે અપૂર્ણાંક પ્રકારના લિટરલ (જેમકે 12.5)નો પૂર્વનિર્ધારિત પ્રકાર double હોય છે. આથી તેનો પ્રકાર float તરીકે ગણવા માટે લિટરલ 12.5 અને 7ના અંતમાં (suffix) f લખવામાં આવેલ છે.

- system.out.println() મેથડ પ્રમાણભૂત નિર્ગમ એકમ ઉપર માત્ર સંદેશો છાપે છે. આ મેથડમાં માત્ર એક જ આર્ગ્યુમેન્ટ, એક સ્ટ્રિંગ, હોય છે, પણ આપણે આ શાબ્દિક લખાણ સાથે જોડવા માટે +નો ઉપયોગ કરી શકીએ છીએ. પહેલાં જણાવ્યું હતું તે પ્રમાણે જાવા સંખ્યાનું સ્ટ્રિંગમાં રૂપાંતર કરે છે અને પછી તે બંનેને જોડે છે.

- જ્યારે x=6 અને y=–4 હોય, ત્યારે x%yનું પરિણામ 2 મળે છે, કારણકે પ્રથમ સંકાર્ય ધન છે.

- જ્યારે a=12.5 અને x=6 હોય ત્યારે a%xનું પરિણામ 0.5 મળે છે. અહીં ભાગાકારનું પૂર્ણાંક ભાગફળ 2 અને શેષ 0.5 છે. એ જ રીતે જો a=12.5 અને b=7.2 હોય, તો a%bમાં પૂર્ણાંક ભાગફળ 1 અને શેષ 5.3 છે.

- System.out.println ("a / x = " + (a / x)); વિધાનમાં મિશ્રિત પ્રકારનાં સંકાર્ય જુઓ. અહીં, ચલ a એ float છે અને x ચલ int છે, ભાગાકાર પછી તેનું પરિણામ float છે. પદાવલીની ગણતરી સમયે કિંમતોને ઉચ્ચ ડેટા-પ્રકાર સંકાર્યમાં રૂપાંતર કરવામાં આવેલ છે.

એ જ પ્રોગ્રામના System.out.println મેથડના બીજા callમાં રહેલી પદાવલી x+yના કૌંસને દૂર કરીને પ્રયોગ કરી જુઓ, એટલે કે આકૃતિ 7.8ના કોડલિસ્ટિંગની 11મી લીટીમાં રહેલી પદાવલી "x + y = " + (x + y) ને પદાવલી "x + y = " + x + y વડે બદલો અને આકૃતિ 7.9માં દર્શાવેલા પરિણામનું વિશ્લેષણ કરો.

ArithOperators.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 ArithOperators.java

```java
class ArithOperators
{
    public static void main (String[] args)
    {
        short x = 6;
        int y = 4;
        float a = 12.5f;  // suffix f to explicitly specify floating-point literal
        float b = 7.2f;   // suffix f for floating-point literal

        System.out.println ("x is " + x + ", y is " + y);
        System.out.println ("x + y = " + x + y);
        System.out.println ("x - y = " + (x - y));
        System.out.println ("x / y = " + (x / y));
        System.out.println ("x % y = " + (x % y));
        x = -6;
        System.out.println ("x % y = " + (x % y));
        y=-4;
        System.out.println ("x % y = " + (x % y));
        x=6; y=-4;
        System.out.println ("x % y = " + (x % y));

        System.out.println ("a is " + a + ", b is " + b);
        System.out.println ("a / b = " + (a / b));
        System.out.println ("a / x = " + (a / x));
        System.out.println ("a % x = " + (a%x));
        System.out.println ("a % b = " + (a%b));
    } // end main()
} // end class ArithOperators
```

```
>javac ArithOperators.java
>Exit code: 0
>java -cp . ArithOperators
x is 6, y is 4
x + y = 64
x - y = 2
x / y = 1
x % y = 2
x % y = -2
x % y = -2
x % y = 2
a is 12.5, b is 7.2
a / b = 1.7361112
a / x = 2.0833333
a % x = 0.5
a % b = 5.3
>Exit code: 0
```

આકૃતિ 7.9 : આકૃતિ 7.8ના કોડલિસ્ટિંગની 11મી લીટીમાં રહેલ (x+y)માંથી કૌંસ ( ) કાઢી નાખવાની અસર

ઇન્ક્રિમેન્ટ અને ડિક્રિમેન્ટ પ્રક્રિયક (Increment and Decrement Operators)

એકપદી પ્રક્રિયક ++ અને ––ને અનુક્રમે વૃદ્ધિ પ્રક્રિયક (ઇન્ક્રિમેન્ટ ઑપરેટર) અને હ્રાસપ્રક્રિયક (ડિક્રિમેન્ટ ઑપરેટર) કહેવામાં આવે છે. ++ પ્રક્રિયક ચલમાં 1 ઉમેરે છે અને –– પ્રક્રિયક ચલમાંથી 1 બાદ કરે છે.

આ પ્રક્રિયકો કોઈ પણ પ્રકારના પૂર્ણાંક ચલ અને char પ્રકારના ચલ ઉપર પણ વાપરી શકાય છે. જો x એ પૂર્ણાંક ચલ હોય, તો આપણે x++, ++x, x--, --x નો ઉપયોગ એક પદાવલી તરીકે અથવા કોઈ મોટી પદાવલીના ભાગ તરીકે કરી શકીએ છીએ.

જ્યારે ચલના નામ પાછળ ++ અથવા -- પ્રક્રિયકનો ઉપયોગ કરવામાં આવે, ત્યારે તેને પોસ્ટ-ઇન્ક્રિમેન્ટ અથવા પોસ્ટ-ડિક્રિમેન્ટ કહેવાય છે. આવા સંજોગોમાં પદાવલીની ગણતરી કરતાં સમયે ચલની જૂની કિંમત વપરાય છે અને તે પછી ચલની કિંમત વધારવામાં કે ઘટાડવામાં આવે છે. ઉદાહરણ તરીકે, ધારો કે y=4+x++; વિધાનના અમલ પહેલાં ચલ xની કિંમત 3 છે. અહીં પદાવલીની જમણી બાજુની ગણતરી કરતાં સમયે xની જૂની કિંમત વપરાશે અને તે પછી xની કિંમતની વૃદ્ધિ (increment) કરવામાં આવશે. આથી, આપણને પરિણામ xની કિંમત 4 અને yની કિંમત 7 મળશે.

જ્યારે ચલનાં નામ પહેલાં ++ અથવા -- પ્રક્રિયકનો ઉપયોગ કરવામાં આવે, ત્યારે તેને પ્રી-ઇન્ક્રિમેન્ટ અથવા પ્રી-ડિક્રિમેન્ટ કહેવાય છે. આ કિસ્સામાં ચલની કિંમત પહેલાં વધારવામાં કે ઘટાડવામાં આવે છે અને તે પછી આ નવી કિંમત પદાવલીની ગણતરીમાં વપરાય છે. ઉદાહરણ તરીકે, y=4+ ++x વિધાનમાં xની જૂની કિંમત 3 છે, તો પહેલા xની કિંમત વધારવામાં આવશે અને આ નવી કિંમતનો ઉપયોગ પદાવલીની જમણી બાજુની ગણતરીમાં થશે. આથી xની કિંમત 4 અને yની કિંમત 8 થશે.

જ્યારે આ પ્રક્રિયક ફક્ત એક જ પદના વિધાનમાં વાપરવામાં આવે, ત્યારે ચલની પહેલાં કે પછીમાં કોઈ તફાવત નથી. ઉદાહરણ તરીકે, વિધાન x++; અને ++x; એ સ્ટેન્ડએલોન સ્ટેટમેન્ટ (standalone statement) છે.

તુલનાત્મક પ્રક્રિયક (Comparison Operators)

તુલનાત્મક પ્રક્રિયક (comparison operators)ને સંબંધાત્મક પ્રક્રિયક (relational operators) પણ કહેવાય છે. જાવામાં તુલનાત્મક પ્રક્રિયકો ==, !=, <, >, <=, અને >= છે. આ બધા પ્રક્રિયકોનો અર્થ નીચે મુજબ છે :

A == B	A અને B "સમાન" છે ?
A != B	A અને B "સમાન નથી" ?
A < B	B "કરતાં A નાનો" છે ?
A > B	B "કરતાં A મોટો" છે ?
A <= B	B "કરતાં A નાનો અથવા સમાન" છે ?
A >= B	B "કરતાં A મોટો અથવા સમાન" છે ?

આ પ્રક્રિયકો કોઈ પણ પ્રકારના આંકડાકીય (numeric) પ્રકારોની અને char પ્રકારની કિંમતોની સરખામણી કરવા માટે વપરાય છે. char પ્રકાર માટે તેની યુનિકોડ આંકડાકીય કિંમત સરખામણીમાં વપરાય છે.

તુલનાત્મક પ્રક્રિયકના ઉપયોગથી પદાવલીનું પરિણામ બુલિયન મળે છે, એટલે કે પરિણામ true અથવા false હોય છે. આથી આવી પદાવલીઓ બુલિયન-કિંમત ધરાવતી પદાવલી (boolean-valued expression) પણ કહેવાય છે.

આપણે બુલિયન-કિંમત ધરાવતી પદાવલીઓ કોઈ બુલિયન ચલને પણ આપી શકીએ (assign), જે રીતે કોઈ સંખ્યા આંકડાકીય ચલને (numeric variable).

== અને != પ્રક્રિયકો બુલિયન-કિંમતોની સરખામણી કરવા માટે થાય છે. ઉદાહરણ તરીકે,

```
boolean bothPositive;   bothPositive = ((x > 0) == (y > 0));
```

સામાન્ય રીતે, તુલનાત્મક પ્રક્રિયકોનો ઉપયોગ if વિધાનો અને લૂપ (loops)માં થાય છે.

તાર્કિક પ્રક્રિયકો (Logical Operators)

તાર્કિક પ્રક્રિયકો બુલિયન ઑપરેટર (boolean operators) પણ કહેવાય છે, કારણકે તે બુલિયન પ્રકારના સંકાર્ય ઉપર કાર્ય કરે છે. જાવામાં AND, OR, XOR અને NOT જેવાં તાર્કિક કાર્યો કરવા માટે અનુક્રમે &&, ||, ^ અને ! પ્રક્રિયકો વપરાય છે.

તાર્કિક AND કાર્ય માટે બે બુલિયન કિંમતોને જોડવા માટે બુલિયન પ્રક્રિયક && વપરાય છે. પદાવલી A && Bનું પરિણામ true છે, માત્ર જ્યારે બંને સંકાર્ય A અને B true હોય. ઉદાહરણ તરીકે, જો બંને x બરાબર 0 અને y બરાબર 0 હોય તો જ (x == 0) && (y == 0) ની કિંમત true મળે.

તાર્કિક OR માટે બુલિયન પ્રક્રિયક || (બે ઊભી લીટી અક્ષરો) છે. જો Aની કિંમત true હોય અથવા Bની કિંમત true હોય અથવા બન્ને true હોય તો પદાવલી A || Bની કિંમત true મળે છે. જો A અને B બંને false હોય, તો જ પદાવલીનું પરિણામ false મળે. ઉદાહરણ તરીકે, જો x અને y બંને 0 બરાબર ન હોય તો જ (x == 0) && (y == 0)નું પરિણામ false મળે.

તાર્કિક NOT માટે બુલિયન પ્રક્રિયક ! છે અને તે એકપદી (unary) પ્રક્રિયક છે. તેનું પરિણામ પૂરકમાં (complement) પરિણમે છે. જો સંકાર્ય true હોય, તો પરિણામ false મળે અને એથી વિપરીત રીતે પણ.

આ ઉપરાંત, તાર્કિક કાર્ય XOR (exclusive OR) માટે પ્રક્રિયક ^ છે. તે તેના પ્રક્રિયકો અલગ-અલગ હોય, તો જ true કિંમત પરત કરે છે અન્યથા false પરત કરે છે. આથી જ્યારે બંને સંકાર્યની એક સમાન બુલિયન કિંમત હોય ત્યારે પરિણામ false મળે છે. ઉદાહરણ તરીકે, (x == 0) ^ (y == 0) ની કિંમત true મળશે, જો x અથવા yમાંથી ફક્ત એકની કિંમત શૂન્ય હોય.

શૉર્ટસર્કિટિંગ (Short Circuiting)

જ્યારે શરતી પ્રક્રિયકો AND અથવા OR (&& અને ||) વાપરવામાં આવે છે, ત્યારે જાવા બીજા સંકાર્યની ગણતરી કરશે નહીં, જ્યાં સુધી પરિણામ મેળવવા તે જરૂરી ન હોય. જો &&ના કિસ્સામાં પહેલું સંકાર્ય false હોય, તો બીજા સંકાર્યની ગણતરી કરવાની જરૂર નથી. એ જ પ્રમાણે ||ના કિસ્સામાં જો પહેલું સંકાર્ય true હોય, તો બીજા સંકાર્યની ગણતરી કરવાની જરૂર નથી.

ઉદાહરણ : પદાવલી (x != 0) && (y/x > 1) લો. જો xની કિંમત 0 હોય, તો આપેલી પદાવલીના ભાગ (x !=0)ની કિંમત false થશે. આપણે જાણીએ છીએ કે તાર્કિક પ્રક્રિયક &&નું પરિણામ false મળશે જ્યારે કોઈ પણ એક પ્રક્રિયકની કિંમત false હોય, આથી આપેલી પદાવલીના બાકીના ભાગ (y/x > 1)ની ગણતરી કરવાની જરૂર નથી. અહીં ટૂંકો રસ્તો લઈને ગણતરી કરેલી છે અને શૂન્ય વડે ભાગાકાર ટાળેલ છે. શૉર્ટસર્કિટિંગ વિના (ઉપર જણાવેલો ટૂંકો રસ્તો ન લેવાથી), અમલ કરતાં સમયે 'શૂન્ય વડે ભાગાકાર' (Division by Zero) ભૂલ આવી શકે.

શરતી પ્રક્રિયક (Conditional Operator)

જાવામાં શરતી પ્રક્રિયક ત્રિપદી પ્રક્રિયક (ternary operator) છે જેમાં ત્રણ સંકાર્ય હોય છે. તે ત્રણ સંકાર્ય જુદા પાડવા માટે પદાવલીમાં બે ચિહ્નો *?* અને *:* નો ઉપયોગ કરે છે. તેનું સ્વરૂપ નીચે પ્રમાણે હોય છે :

<boolean-expression> ? <expression1> : <expression2>

અહીં, પ્રથમ સંકાર્ય બુલિયન પદાવલી છે, અને તેની સૌપ્રથમ ગણતરી કરવામાં આવે છે. જો તેની કિંમત true હોય તો આખી પદાવલીની કિંમત એ બીજા સંકાર્ય expression1ની કિંમત સમાન થશે, અન્યથા ત્રીજા સંકાર્ય expression2 સમાન થશે.

ઉદાહરણ : વિધાન ધ્યાનમાં લો : next = (N % 2 == 0) ? (N/2) : (3*N+1);

ધારો કે Nની કિંમત 8 છે. પ્રથમ સંકાર્ય (N % 2 == 0) ની કિંમત true છે આથી જમણી બાજુની પદાવલીની કિંમત પદાવલી (N/2)ની કિંમત બરાબર થશે એટલે કે 4 થશે. જો Nની કિંમત એકી સંખ્યા હોય તો પહેલી પદાવલીની કિંમત false થશે અને (3*N+1)ની કિંમત nextને એસાઈન (assign) કરવામાં આવશે. અહીં નોંધ કરો કે આ ઉદાહરણમાં કૌંસની જરૂર નથી પણ તે સમીકરણને વાંચવામાં સરળ બનાવે છે.

એસાઈન્મેન્ટ (Assignment)

જાવામાં જે પદાવલી એસાઈન્મેન્ટ પ્રક્રિયક (=) ધરાવતી હોય, તેને સામાન્ય રીતે એસાઈન્મેન્ટ સ્ટેટમેન્ટ (assignment statement - સોંપણી કરતું વિધાન) કહેવામાં આવે છે.

આપણે એક વખત ચલ ઘોષિત કરીએ (declare) તે પછી આપણે તેને કિંમત એસાઈન્મેન્ટ પ્રક્રિયક '=' વડે આપી શકીએ છીએ. જાવામાં કોઈ ચલની અંદર ડેટા મેળવવાની એક રીત એસાઈન્મેન્ટ સ્ટેટમેન્ટ છે. એસાઈન્મેન્ટ સ્ટેટમેન્ટનું સ્વરૂપ નીચે મુજબ હોય છે :

< ચલ > = < પદાવલી >; એટલે કે <variable> = <expression>;

અહીં <expression> એ ડેટાકિંમતનો નિર્દેશ કે ગણતરીનો ઉલ્લેખ છે.

જ્યારે એસાઈન્મેન્ટ વિધાનનો અમલ થાય છે, ત્યારે સૌ પ્રથમ = ચિહ્નની જમણી બાજુની પદાવલીની કિંમત શોધવામાં આવે છે અને તે પરિણામને = ચિહ્નની ડાબી બાજુના ચલમાં મૂકવામાં આવે છે.

ઉદાહરણ : એક સાદું એસાઈન્મેન્ટ વિધાન લો : rate = 10.02f;

અહીં, એસાઈન્મેન્ટ વિધાનમાં ચલ 'rate' છે અને પદાવલી કે expression સંખ્યા '10.02f' છે. આ વિધાનનો અમલ થવાથી float પ્રકારના ચલ 'rate'ની કિંમત બદલાઈને 10.02 થશે.

હવે, બીજું એસાઈન્મેન્ટ વિધાન લો : balance = balance - cost;

અહીં, સૌ પ્રથમ ચલ balance અને costની મેમરીમાં રહેલી કિંમતોનો ઉપયોગ કરીને પદાવલી balance-costની કિંમત શોધવામાં આવે છે. તે પછી ચલ balanceની જૂની કિંમતને બદલીને પરિણામની કિંમત મૂકવામાં આવે છે.

જ્યારે પદાવલીની જમણી બાજુએ કોઈ ચલ વાપરવામાં આવે છે, ત્યારે તેનો અર્થ એ છે કે ચલમાં કિંમત સંગ્રહ કરેલી છે. જ્યારે એસાઈન્મેન્ટ સ્ટેટમેન્ટની ડાબી બાજુએ ચલ વાપરવામાં આવે છે ત્યારે તે કોઈ મેમરી સ્થાનાંકનો નિર્દેશ કરે છે કે જેને ચલનું નામ આપેલું હોય છે, જેમાં કિંમત મૂકવામાં આવે છે. આથી, પદાવલીની ડાબી બાજુના ચલને *lvalue* કહેવામાં આવે છે જે મેમરીના સ્થાનાંકનો નિર્દેશ કરે છે.

સામાન્ય રીતે, એસાઈન્મેન્ટ સ્ટેટમેન્ટની જમણી બાજુની પદાવલીનો પ્રકાર ડાબી બાજુના ચલના પ્રકાર જેવો હોવો જોઈએ. જો તે મેળ ખાતા ન હોય તો પદાવલીની કિંમત આપમેળે ચલના પ્રકાર સાથે મેળ ખાય (match) તે પ્રમાણે રૂપાંતરિત થાય છે. જો ડાબી બાજુના ચલના કરતા પદાવલીનો ડેટાપ્રકાર ઉચ્ચ હોય (larger) તો તે ચોકસાઈ સમસ્યા (precision problem)ને કારણે ભૂલ (error)માં પરિણમી શકે છે. ઉદાહરણ તરીકે, intમાંથી short કે doubleમાંથી floatમાં આપોઆપ રૂપાંતર હોઈ શકે નહીં.

શૉર્ટહેન્ડ એસાઈન્મેન્ટ પ્રક્રિયક (Shorthand Assignment Operators)

જાવા પણ એસાઈન્મેન્ટની શૉર્ટહેન્ડ આવૃત્તિ પૂરી પાડે છે. તે ટાઈપ કરવાનો સમય બચાવે છે. તેનું સ્વરૂપ <variable> <operator> = <expression> હોય છે. તેની અસર <variable> = <variable> <operator> <expression> સમાન જ છે. અહીં પ્રક્રિયક બે સંકાર્ય વાપરતું દ્વિપદી (binary) પ્રક્રિયક હોવું જોઈએ.

શૉર્ટહેન્ડ એસાઈન્મેન્ટ પ્રક્રિયકનાં કેટલાંક ઉદાહરણો નીચે મુજબ છે : a += b અને q &&= p. અહીં a += b એ a = a + b સમાન છે, અને q &&= p એ q = q && p સમાન છે.

ટાઈપકાસ્ટ (Type-cast)

કેટલાક કિસ્સાઓમાં આપણે ફરજિયાતથી રૂપાંતર ઇચ્છતા હોઈએ કે જે આપમેળે ન બને. આ માટે આપણે જે વાપરીએ છીએ, તેને ટાઇપ કાસ્ટ (type-cast) કહેવામાં આવે છે. આપણે જે કિંમતનું રૂપાંતર કરવા ઇચ્છતા હોય, તેની આગળ કૌંસમાં ટાઇપ નેઇમ (type-name) લખીને ટાઇપકાસ્ટ જણાવી શકીએ છીએ. તેનું સ્વરૂપ (<data-type>) <expression> હોય છે.

ઉદાહરણ : int a; short b;

a = 17; b = (short)a;

અહીં, ચલ aનું ટાઇપકાસ્ટ વાપરીને short પ્રકારની કિંમતથી ચોક્કસપણે (explicitly) રૂપાંતર કરેલું છે.

જાવા પ્રક્રિયકોના પ્રિસિડન્સ અને એસોસિએટિવિટી (Precedence and Associativity of Java Operators) :

જ્યારે પદાવલીમાં કેટલાક પ્રક્રિયકો હોય ત્યારે પદાવલીના પ્રક્રિયકોની ક્યા ક્રમમાં ગણતરી કરવી તે જણાવતા સારી રીતે વ્યાખ્યાયિત કરેલા નિયમો જાવા ધરાવે છે. આ પ્રક્રિયકોની **અગ્રતા** (priority અથવા precedence) પ્રમાણે પદાવલીની ગણતરી કરવામાં આવે છે.

અગ્રતાક્રમ કે પ્રિસિડન્સ ઑર્ડર (Precedence Order)

જ્યારે બે પ્રક્રિયકની અલગ અલગ અગ્રતા હોય, ત્યારે ઉચ્ચ અગ્રતા (higher precedence) ધરાવતા પ્રક્રિયક ઉપર પહેલાં પ્રક્રિયા કરવામાં આવે છે. ઉદાહરણ : પદાવલી a + b * cમાં ગુણાકારનો સરવાળા કરતાં વધારે પ્રિસિડન્સ હોવાથી પહેલાં b*cની ગણતરી કર્યા પછી તેનું પરિણામ aમાં ઉમેરવામાં આવે છે. આથી તે a + (b*c) લખવા સમાન છે. બાહ્ય કૌંસ દ્વારા અગ્રતાનિયમો (પ્રિસિડન્સ રૂલ્સ) બિનઅસરકારક બનાવી શકાય છે. આથી ગણતરી કરતાં સમયે ગૂંચવાડો ઊભો કરવાને બદલે છૂટથી કૌંસનો ઉપયોગ કરો.

સહચારિતા કે એસોસિએટિવિટી (Associativity)

જ્યારે પદાવલીમાં એક સમાન પ્રિસિડન્સ ધરાવતાં બે પ્રક્રિયકો હોય, ત્યારે પદાવલીની ગણતરી તેના **એસોસિએટિવિટી** પ્રમાણે કરવામાં આવે છે. એસોસિએટિવિટી કઈ દિશામાં (ડાબી બાજુથી જમણી બાજુ અથવા જમણી બાજુથી ડાબી બાજુ) કાર્ય કરવાનું છે, તે નક્કી કરે છે. મોટા ભાગના કિસ્સાઓમાં એસોસિએટિવિટી ડાબી બાજુથી જમણી બાજુ હોય છે. એકપદી કાર્યો અને એસાઇન્મેન્ટમાં તે જમણી બાજુથી ડાબી બાજુ હોય છે.

આ પ્રકરણમાં ચર્ચા કરેલાં પ્રક્રિયકોની યાદી કોષ્ટક 7.4માં આપેલી છે, જેનો ક્રમ સૌથી વધારે પ્રિસિડન્સ (જેની ગણતરી સૌપ્રથમ થાય)થી ઓછામાં ઓછી પ્રિસિડન્સ (જેની ગણતરી સૌથી છેલ્લી થાય) પ્રમાણે છે :

Operations	Operators	Associativity
Unary operations	++, --, !, unary – and +, type-cast	Right-to-left
Multiplication, division, modulus	*, ?, %	Left-to-right
Addition and subtraction	+, -	Left-to-right
Relational operators	<, >, <=, >=	Left-to-right
Relational operators (Equality and inequality)	==, !=	Left-to-right
Logical AND	&&	Left-to-right
Logical OR	||	Left-to-right
Conditional operator	? :	Right-to-left
Assignment operators	=, +=, -=, *=, /=, %=	Right-to-left

કોષ્ટક 7.4 : પ્રક્રિયકો અને તેનું પ્રિસિડન્સ

ઉદાહરણ : x = y = z = 7ને x = (y = (z = 7)) તરીકે ગણીને ત્રણે ચલમાં કિંમત 7 મળશે. આ એસાઈન્મેન્ટ પ્રક્રિયકની જમણી બાજુથી ડાબી બાજુની એસોસિએટિવિટીને કારણે છે. તમને યાદ હશે કે એસાઈન્મેન્ટ એ એક પ્રક્રિયક છે, આથી એસાઈન્મેન્ટ સ્ટેટમેન્ટની કિંમત જમણી બાજુની પદાવલી બરાબર છે. આથી, x=y=z=7માં પહેલા z=7ની ગણતરી થાય છે. તે ચલ zને કિંમત 7 એસાઈન કરે છે અને પદાવલી z=7ની કિંમત પણ 7 છે, આથી તે પદાવલી x=y=7 બનાવે છે.

બીજી બાજુએ, 72 / 2 / 3 એ (72 / 2) / 3 તરીકે ગણવામાં આવે છે કારણકે / પ્રક્રિયકની ડાબી બાજુથી જમણી બાજુની એસોસિએટિવિટી છે. અહીં એ નોંધવું જોઈએ કે જાવા ભાષાના વિગતવાર વર્ણનમાં કોઈ ચોક્કસ પ્રક્રિયક અગ્રતાક્રમ કોષ્ટક નથી અને વેબ પર તથા પાઠ્યપુસ્તકોમાં ઉપલબ્ધ કોષ્ટકો કેટલીક નાની બાબતોમાં સંમત નથી.

કન્ટ્રોલ સ્ટ્રક્ચર (Control Structures)

સામાન્ય રીતે, વિધાનોનો અમલ એક પછી એક એમ ક્રમિક રીતે થાય છે. અમુક સમયે પ્રોગ્રામના તર્કને આ ક્રમનો પ્રવાહ બદલવાની જરૂર પડે છે. જે વિધાનો અમલના પ્રવાહને નિયંત્રણ કરવા સમર્થ બનાવે છે તેને નિયંત્રણમાળખાં કે કન્ટ્રોલ-સ્ટ્રક્ચર (control structure) ગણવામાં આવે છે.

કન્ટ્રોલ સ્ટ્રક્ચર બે પ્રકારનાં હોય છે : લૂપ્સ (loops) અને બ્રાન્ચિસ (branches). લૂપનો ઉપયોગ અમુક શરત સંતોષાય ત્યાં સુધી વિધાનોની શ્રેણીનું વારંવાર પુનરાવર્તન કરવાનું હોય ત્યારે થાય છે. બ્રાન્ચનો ઉપયોગ જ્યારે બે કે વધુ શક્ય કાર્ય-દિશામાંથી પસંદગી કરવાની હોય, ત્યારે થાય છે, આથી તેને સિલેક્ટિવ સ્ટ્રક્ચર (selective structure) પણ કહેવામાં આવે છે. પ્રોગ્રામમાં સામાન્ય પ્રવાહના નિયંત્રણ માટે જાવામાં વપરાતા કન્ટ્રોલ-સ્ટ્રક્ચર નીચે મુજબ છે : if વિધાન, switch વિધાન, while લૂપ, do...while લૂપ અને for લૂપ. આ બધાં માળખાં એ એક વિધાન કે વિધાનોનો બ્લૉક હોઈ શકે છે.

બ્લૉક (Block)

બ્લૉક-સ્ટેટમેન્ટ એ કૌંસની જોડી "{" અને "}" વચ્ચે લખાયેલાં વિધાનોનું જૂથ છે.

બ્લૉકનું સ્વરૂપ નીચે મુજબ હોય છે :

```
{
    <statements>
}
```

બ્લૉકનો ઉપયોગ નીચે જણાવેલા અનેક હેતુઓ માટે કરી શકાય છે :

- સામાન્ય રીતે કન્ટ્રોલ-સ્ટ્રક્ચરમાં વિધાનોની શ્રેણીના જૂથને એક એકમ બનાવી એક વિધાન તરીકે ગણવા માટે. (ટૂંક સમયમાં ચર્ચા કરીશું.)
- તાર્કિક રીતે સંબંધિત વિધાનોનું જૂથ બનાવવા માટે.
- બ્લૉકની અંદર વિધાનોના સ્થાનિક અવકાશ (local scope) સાથે નવા ચલ બનાવવા.

ઉદાહરણ :

```
{ // This block exchanges the values of x and y
        int temp;       // temporary variable for use in this block only
        temp = x;       // save a copy of x in temp
        x = y;          // copy y into x
        y = temp;       // copy temp into y
}
```

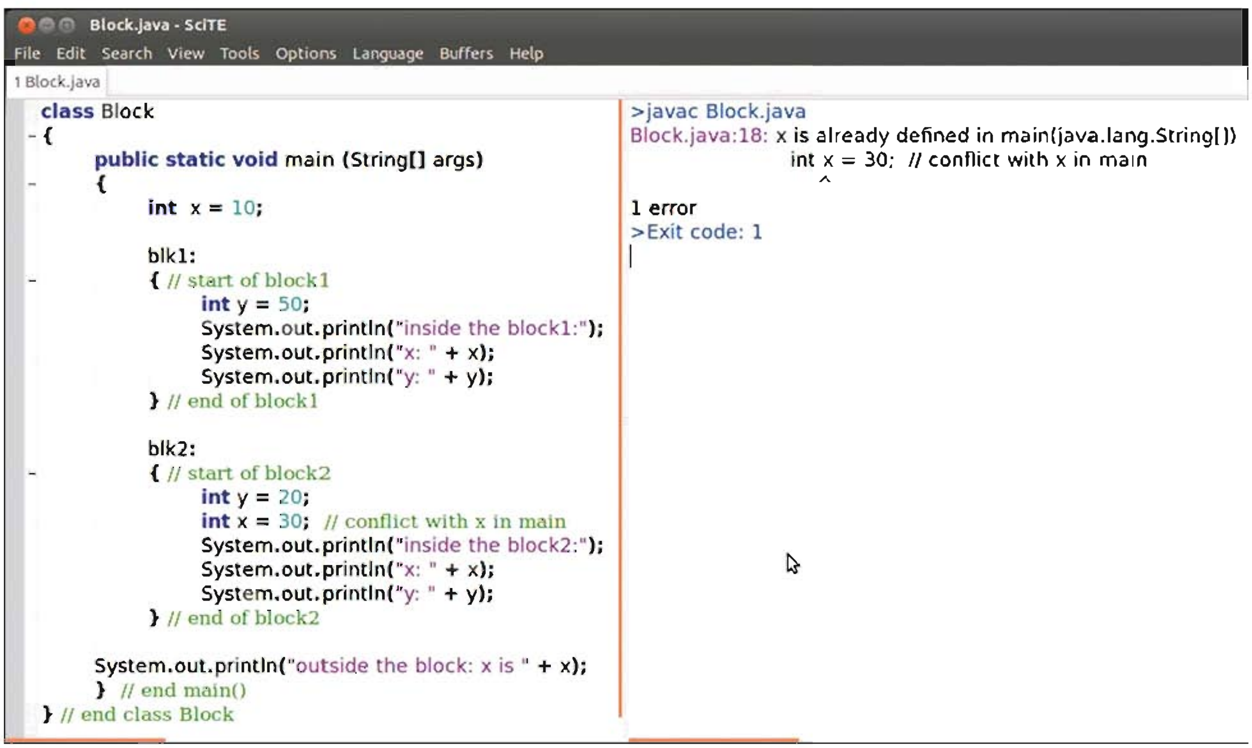
જ્યારે આપણે કોઈ બ્લૉકની અંદર ચલ ઘોષિત કરીએ છીએ, ત્યારે તે ચલ તે બ્લૉકમાં જ સ્થાનિક (local) રહે છે અને તે બ્લૉકની બહાર તેનું અસ્તિત્વ રહેતું નથી. આપણે જે બ્લૉકની અંદર ઘોષિત કરેલ હોય તેની બહાર તેનો ઉપયોગ કરી શકતા નથી.

આપણે જે બ્લૉકની અંદર ચલ ઘોષિત કરેલા હોય તે બ્લૉકની બહાર તે સંપૂર્ણપણે ન મેળવી શકાય તેવા (inaccessible) અને અદૃશ્ય (invisible) રહે છે. જ્યારે ચલ ઘોષિત કરવાના વિધાનનો અમલ કરવામાં આવે છે, ત્યારે ચલમાં કિંમત રાખવા માટે મેમરીની ફાળવણી કરવામાં આવે છે. જ્યારે બ્લૉકનો અંત આવે છે, ત્યારે તે મેમરી છૂટી કરવામાં આવે છે અને તેનો ફરી ઉપયોગ કરવા માટે ઉપલબ્ધ બને છે. ચલ બ્લૉક પૂરતો સ્થાનિક કહી શકાય.

એક સામાન્ય ખ્યાલ છે જેને ચલનો "સ્કૉપ (scope)" (કાર્યક્ષેત્રની મર્યાદા) કહેવામાં આવે છે. ચલની કાર્યક્ષેત્રની મર્યાદા (scope) પ્રોગ્રામનો એ ભાગ છે જેમાં તે ચલ માન્ય છે. જે બ્લૉકમાં ચલ વ્યાખ્યાયિત કરેલો હોય તે બ્લૉક પૂરતો જ ચલનો સ્કૉપ મર્યાદિત રહે છે.

જ્યારે આપણે કોઈ ચલના સ્કૉપની અંદર બીજો ચલ તે જ નામનો ઘોષિત કરવા પ્રયત્ન કરીએ, ત્યારે ભૂલ (error) મેળવીએ છીએ. આકૃતિ 7.10માં આપેલું કોડલિસ્ટિંગ અને પરિણામ જુઓ. અહીં આપણે ફક્ત સારી સમજણ માટે labelled બ્લૉકનો ઉપયોગ કર્યો છે. અહીં આપેલા ઉદાહરણમાં તે વાપરવું અનિવાર્ય નથી. આપણે labelled blockનો ઉપયોગ આ પ્રકરણમાં હવે પછી જોઈશું.

આકૃતિ 7.10માં આપેલા કોડલિસ્ટિંગમાં આપણે blk2 નામના બ્લૉકમાં ચલ x ઘોષિત કરવા પ્રયત્ન કર્યો છે. આ બ્લૉક blk2 એ main મેથડ બ્લૉકની અંદર છે. main મેથડની અંદર ઘોષિત કરેલા ચલ xનો સ્કૉપ blk2ની અંદર પણ છે. આથી, blk2 બ્લૉકની અંદર ચલ x ઘોષિત કરવાથી તે main મેથડ બ્લૉકમાંના ચલ x સાથે સંઘર્ષમય સ્થિતિ (conflicts) ઉદ્ભવશે અને કંપાઇલર ભૂલ દર્શાવશે.



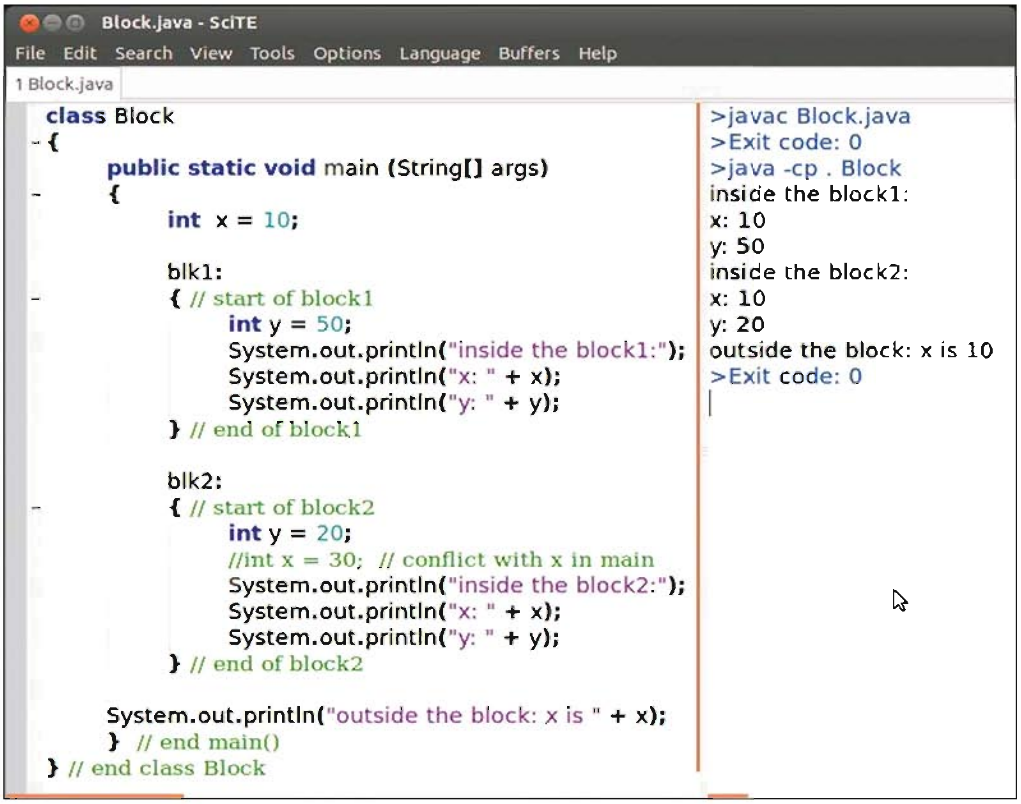

```
class Block
{
    public static void main (String[] args)
    {
        int  x = 10;

        blk1:
        { // start of block1
            int y = 50;
            System.out.println("inside the block1:");
            System.out.println("x: " + x);
            System.out.println("y: " + y);
        } // end of block1

        blk2:
        { // start of block2
            int y = 20;
            int x = 30;  // conflict with x in main
            System.out.println("inside the block2:");
            System.out.println("x: " + x);
            System.out.println("y: " + y);
        } // end of block2

    System.out.println("outside the block: x is " + x);
    } // end main()
} // end class Block
```

```
>javac Block.java
Block.java:18: x is already defined in main(java.lang.String[])
                int x = 30;  // conflict with x in main
                    ^
1 error
>Exit code: 1
```

આકૃતિ 7.10 : એક જ કાર્યક્ષેત્રની મર્યાદા (સ્કૉપ)માં સંઘર્ષમય સ્થિતિમાં ચલનાં નામો

આકૃતિ 7.11માં કોડમાં ફેરફાર કરીને દર્શાવ્યો છે. અહીં, main મેથડના પૂર્ણ સ્કોપમાં ચલ x ઘોષિત કરેલ છે. આ જ બ્લૉકમાં blk1 અને blk2 હોવાથી તેમાં પણ ચલ x ઉપલબ્ધ રહે છે. blk1 બ્લૉકમાં ચલ y ઘોષિત કરેલ છે જે બ્લૉકની બહાર અસ્તિત્વ ધરાવતો નથી. blk1ના અમલ પછી ચલ y કાઢી નાખવામાં આવે છે. જ્યારે blk2માં y ઘોષિત કરવામાં આવે છે, ત્યારે તે ઉપલબ્ધ મેમરીમાંથી કોઈ પણ મેમરીસ્થાન ઉપર કરે છે. જો આપણે blk2માં ચલ yને કોઈ કિંમત ન આપીએ.

અને જ્યારે blk2માં તેનો નિર્દેશ કરવામાં આવે ત્યારે તે ભૂલ (error) પરત કરે છે. આનો અર્થ એ થાય કે blk1ના ચલ yને blk2ના ચલ y સાથે કોઈ સંબંધ નથી. હકીકતમાં blk1નો ચલ y એ blk2માં મેળવી શકાતો નથી.



```
class Block
{
    public static void main (String[] args)
    {
        int  x = 10;

        blk1:
        { // start of block1
            int y = 50;
            System.out.println("inside the block1:");
            System.out.println("x: " + x);
            System.out.println("y: " + y);
        } // end of block1

        blk2:
        { // start of block2
            int y = 20;
            //int x = 30;  // conflict with x in main
            System.out.println("inside the block2:");
            System.out.println("x: " + x);
            System.out.println("y: " + y);
        } // end of block2

    System.out.println("outside the block: x is " + x);
    } // end main()
} // end class Block
```

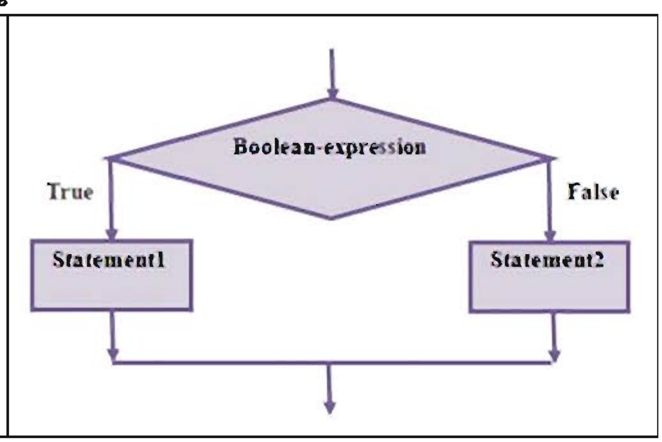
```
>javac Block.java
>Exit code: 0
>java -cp . Block
inside the block1:
x: 10
y: 50
inside the block2:
x: 10
y: 20
outside the block: x is 10
>Exit code: 0
```

આકૃતિ 7.11 : ચલનો સ્કોપ દર્શાવતો પ્રોગ્રામ

if વિધાન (if Statement)

જ્યારે પ્રોગ્રામમાં if વિધાન વાપરવામાં આવે છે ત્યારે બુલિયન પદાવલીની કિંમત true છે કે false તેના પ્રમાણે બેમાંથી એક વિકલ્પ પ્રમાણે કાર્ય કરવા સમર્થ બનાવે છે. તે 'branching' અથવા 'decision' અથવા 'selective' કન્ટ્રોલ સ્ટ્રક્ચરનું ઉદાહરણ છે. તેનું સ્વરૂપ આકૃતિ 7.2માં દર્શાવ્યું છે.

```
if ( <boolean-expression> )
{
        <statement1>
}
else
{
        <statement2 >
}
```



આકૃતિ 7.12 : if વિધાનનું બંધારણ

જ્યારે if વિધાનનો અમલ થાય છે, ત્યારે સૌપ્રથમ બુલિયન પદાવલીની કિંમત શોધવામાં આવે છે. જો કિંમત true હોય, તો તે statement1નો અમલ કરશે; નહિ તો ચાવીરૂપ શબ્દ else પછી લખેલા વિધાનોના બ્લૉકનો અમલ કરશે. if વિધાનમાં <statement> સામાન્ય રીતે બ્લૉકસ્ટેટમેન્ટ હોય છે. તે કોઈ એક વિધાન પણ હોઈ શકે પણ બ્લૉક વાપરવો સલાહભર્યું

છે, જેથી ભવિષ્યમાં જરૂર હોય, તો વધારાનાં વિધાન ઉમેરવાનું કાર્ય સરળ બને. નીચે આપેલો પ્રોગ્રામનો ખંડ જુઓ, જે પૂર્ણાંક સંખ્યા એકી છે કે બેકી તે નક્કી કરવા માટે if વિધાનનો ઉપયોગ કરે છે.

```
if ( x%2 == 0 ) // assume int x
  { // divisible by 2
        System.out.print(x);
        System.out.println (" is even");
  }
else
  { // not divisible by 2
System.out.println (x + " is odd");
  }
```

અહીં ચાવીરૂપ શબ્દ (keyword) else અને else પછી લખેલ બ્લૉક વૈકલ્પિક છે. આથી, else વિના if વિધાનનું સ્વરૂપ નીચે પ્રમાણે હોઈ શકે :

```
if ( <boolean-expression> )
{
        <statement1>
}
```

જ્યારે એક if વિધાનની અંદર બીજું if વિધાન વાપરવામાં આવે, તો તેને nested-if વિધાન કહેવામાં આવે છે. હવે નીચે આપેલું ઉદાહરણ જુઓ, જે મેળવેલા ગુણ પ્રમાણે ગ્રેડ નક્કી કરે છે.

```
if ( marks >= 70 ) // int marks
  { grade = 'A';      // char grade
  }
else
  {
   if (marks >= 60)
        grade = 'B';
   else if (marks >= 50)
               grade = 'C';
        else
         grade = 'F';
 }
```

આપણે nested-if વિધાનનું એક વધારે ઉદાહરણ લઈએ.

```
if ( x > 0 )
        if (y > 0)
               System.out.println("both x and y are greater than zero");
else
        System.out.println("x <= 0");
```

અહીં, એવું જણાય છે કે else ભાગ "if (x > 0)" વિધાનને સંગત છે, પણ હકીકતમાં તે "if (y > 0)" સાથે જોડાયેલ છે, જે તેની નજીક છે. આ રીતે ફક્ત "if (x>0)"ના true ભાગનો અમલ થાય છે.

જો આપણે else ભાગને "if (y>0)" સાથે જોડવા ઇચ્છતા હોય, તો આપણે nested-ifને નીચે જણાવ્યા પ્રમાણે બ્લૉકમાં રાખવું જોઈએ.

```
if ( x > 0 )
{
        if (y > 0)
                System.out.println("both x and y are greater than zero");
}
else
System.out.println("x <= 0");
```

સ્વિચ-સ્ટેટમેન્ટ (Switch Statement)

જ્યારે ચલ કે પદાવલીની કિંમત પ્રમાણે અલગ-અલગ ઘણાં કાર્યોના વિકલ્પ હોય, ત્યારે સ્વિચ-સ્ટેટમેન્ટ (switch statement) વાપરવામાં આવે છે. અહીં, તે પદાવલી ટેસ્ટ કરવાની છે તેનું પરિણામ એવા પ્રકારનું હોવું જોઈએ કે જે જુદી જુદી કિંમતો (discrete) હોય. સ્વિચ-સ્ટેટમેન્ટનું સ્વરૂપ નીચે પ્રમાણે હોય છે :

```
switch (<expression>)
{
        case <constant-1>:
                <statements-1>
                break;
        case <constant-2>:
                <statements-2>
                break;

        // more such cases

        case <constant-n>:
                <statements-n>
                break;

        default:
                <statements-n1>
}
```

સ્વિચ-સ્ટેટમેન્ટમાં ટેસ્ટ-પદાવલીનો પ્રકાર byte, char, short અથવા int હોવો જોઈએ. તે enum ડેટાપ્રકારની પણ હોઈ શકે (અહીં ચર્ચા કરેલી નથી.).

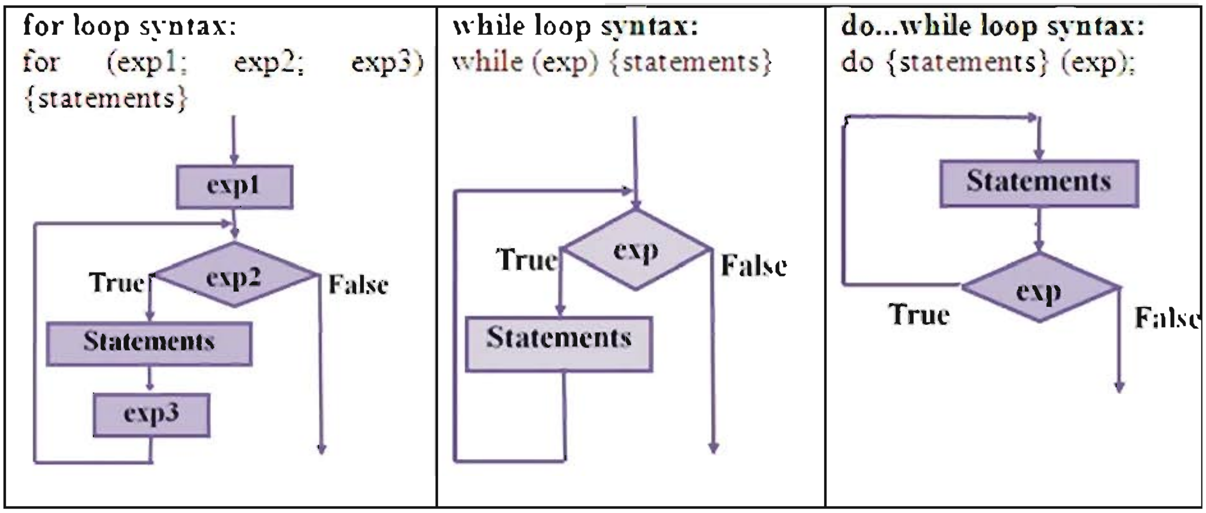
સ્વિચ-સ્ટેટમેન્ટનો અમલ કરતા સમયે case1 થી શરૂ કરી ટેસ્ટ-પદાવલીની કિંમત દરેક case કિંમત સાથે સરખાવવામાં આવે છે. જો બંને કિંમત સમાન મળે, તો તે caseનાં વિધાનો (બ્લૉક હોય તે જરૂરી નથી)નો અમલ થાય છે. જો સમાન

કિંમત ન મળે તો default વિધાનનો અમલ થાય છે. default વિધાન વૈકલ્પિક છે, આથી જો default ન આપેલું હોય અને કોઈ પણ case સાથેની કિંમત મૅચ ન થાય, તો કંઈ પણ કર્યા વિના સ્વિચ-સ્ટેટમેન્ટનો અંત આવે છે.

અહીં નોંધ કરો કે દરેક case વિધાન પછી break વિધાન ફરજિયાત નથી. break વિધાનનો ઉપયોગ સ્વિચ-સ્ટેટમૅન્ટનો અંત લાવવા માટે થાય છે, એટલે કે, સ્વિચના અંત પછીના પહેલા વિધાન ઉપર જવા માટે થાય છે.

રિપીટિટિવ કન્ટ્રોલ-સ્ટ્રક્ચર (Repetitive Control Structures)

જાવા ત્રણ પ્રકારનાં લૂપિંગની રચના પૂરી પાડે છે : for, while અને do...while. આકૃતિ 7.13માં આ બધાં લૂપની વાક્યરચના (syntax) અને માળખું (structure) દર્શાવેલું છે.



આકૃતિ 7.13 : રિપીટિટિવ કન્ટ્રોલ-સ્ટ્રક્ચર (પુનરાવર્તી નિયંત્રણમાળખા)

for અને while લૂપમાં સૌપ્રથમ ટેસ્ટપદાવલીની ગણતરી કરવામાં આવે છે અને જો શરત સાચી હોય, તો લૂપની અંદરનાં વિધાનોનો અમલ થાય છે. આ લૂપ પ્રવેશ-નિયંત્રિત (entry-controlled) અથવા પ્રી-ટેસ્ટ (pre-test) પ્રકારના લૂપ છે. અહીં, એ શક્ય છે કે લૂપની અંદરનાં વિધાનોનો બિલકુલ અમલ જ ન થાય.

જ્યારે પુનરાવૃત્તિ (iterations)ની સંખ્યા અગાઉથી નક્કી હોય, ત્યારે સામાન્ય રીતે for લૂપ વપરાય છે. આ લૂપની અંદર ત્રણે પદાવલીઓ વૈકલ્પિક હોય છે. પહેલી પદાવલી initializer (જે પ્રારંભિક કિંમત આપે છે.), બીજી પદાવલી condition (શરત) અને ત્રીજી પદાવલી iterator (પુનરાવૃત્તિ કરનાર) છે. આથી for(;;) એ માન્ય વિધાન છે, પણ લૂપમાંથી બહાર આવવા માટે કોઈ નિયંત્રણવિધાન (control-statement)ની જરૂર રહે છે. જો break વિધાનનો અમલ ન થાય, તો તે અનંત લૂપમાં પરિણમે છે.

do...while લૂપમાં પહેલાં પદાવલીઓનો અમલ થાય છે અને પછી પદાવલીની કિંમત ટેસ્ટ કરવામાં આવે છે. જો ટેસ્ટની શરતનું પરિણામ true મળે તો લૂપમાં પુનરાવર્તન થાય છે. આ રીતે, do...while લૂપ નિર્ગમન-નિયંત્રિત (exit-controlled) અથવા પોસ્ટ-ટેસ્ટ (post-test) પ્રકારનું લૂપ છે. અહીં, લૂપની અંદરનાં વિધાનો ઓછામાં ઓછા એક વખત અમલમાં આવે છે.

નીચેનાં ઉદાહરણો 0 થી 9ની પૂર્ણાંક સંખ્યાઓ ત્રણે પ્રકારનાં લૂપનો ઉપયોગ કરીને છાપે છે.

for લૂપનો ઉપયોગ :

```
for (int i = 0; i < 10; i++)
{
        System.out.println(i);
}
```

while લૂપનો ઉપયોગ

```
int i = 0;
while(i < 10)
{
        System.out.println(i++); //prints i before applying i++
}
```

do...while લૂપનો ઉપયોગ

```
int i = 0;
do
{
System.out.println(i++);
} while(i < 10);
```

break અને continue વિધાનનો ઉપયોગ (Use of Break and Continue Statement)

break વિધાનનો ઉપયોગ સ્વિચ કે લૂપ પ્રકારની રચનાની બહાર પ્રોગ્રામનું નિયંત્રણ ફેરવવા માટે થાય છે.

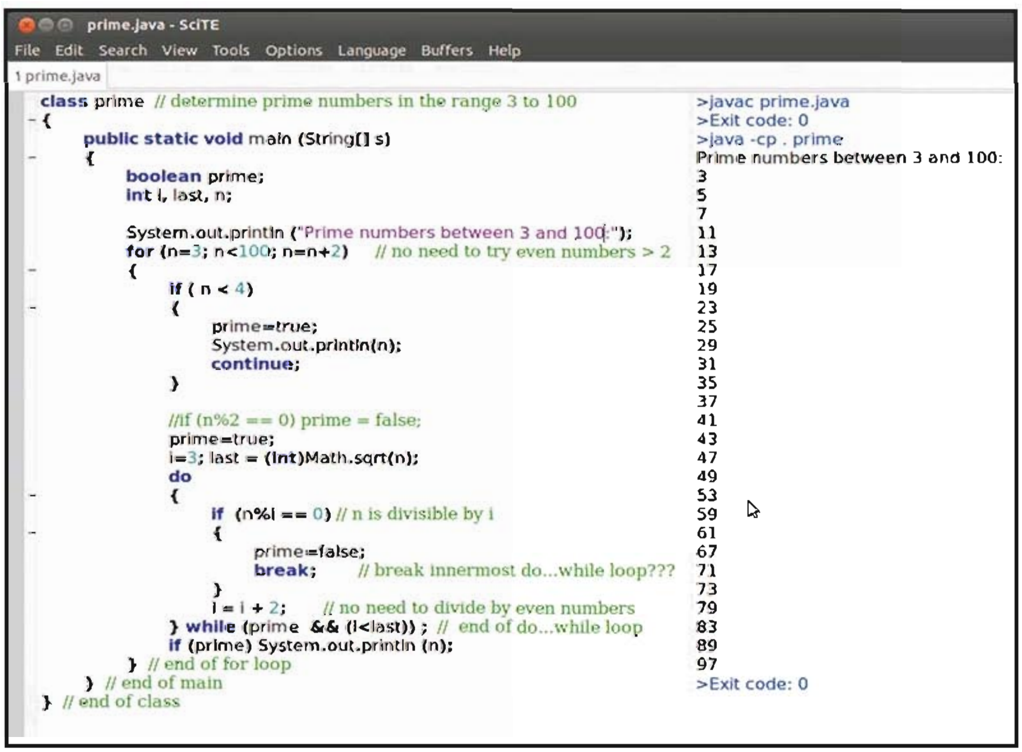
જ્યારે સ્વિચ (switch) સાથે break વાપરવામાં આવે છે, ત્યારે તે પછીનાં વિધાનોનો અમલ કર્યા વિના તેમને છોડી દઈને switch વિધાનના અંત પછીના પહેલા વિધાન ઉપર નિયંત્રણનું સ્થાનાંતર થાય છે.

લૂપ (loop)માં break વિધાન લૂપમાંથી નિર્ગમન (exit) માટે વપરાય છે. જ્યારે લૂપમાં break વિધાનનો અમલ થાય છે, ત્યારે લૂપમાં સમાયેલાં બધાં વિધાનો અમલ કર્યા વિના છોડી દેવામાં આવે છે અને વધારાનું કોઈ પુનરાવર્તન કરવામાં આવતું નથી. પ્રોગ્રામનું નિયંત્રણ લૂપના અંત પછીના પહેલા વિધાન ઉપર સ્થાનાંતર થાય છે. યાદ રાખો કે લૂપની બહાર અને સૌથી નજીકના આ વિધાન ઉપર break કૂદકો મારે છે.

continue વિધાનનો ઉપયોગ લૂપમાંનાં તેનાં પછીનાં વિધાનોને છોડી દેવા માટે અને તે પછીનાં પુનરાવર્તન માટે થાય છે. બંને વિધાનો break અને continueનો ઉપયોગ C ભાષાની રીતે જ થાય છે.

નેસ્ટેડલૂપ (Nested-loops)

જાવામાં સમાન પ્રકારના કે અલગ પ્રકારના લૂપનું નેસ્ટિંગ (Nesting) કરી શકાય છે. આકૃતિ 7.4માં નેસ્ટેડ લૂપ, break અને continue વિધાનોનો ઉપયોગ દર્શાવ્યો છે.



આકૃતિ 7.14 : નેસ્ટેડ લૂપ, break અને continue વિધાનનો ઉપયોગ

આકૃતિ 7.14માં આપણે એક પ્રોગ્રામ લખેલો છે જે 3 થી 100 વચ્ચેની અવિભાજ્ય સંખ્યા (prime numbers) છાપે છે. આ ઉપરાંત તે sqrt વિધેયનો ઉપયોગ પણ દર્શાવે છે. જાવાની ક્લાસ-લાઇબ્રેરીમાં Math નામના ક્લાસનો સમાવેશ થાય છે. Math ક્લાસમાં ગાણિતિક કાર્યોનો સંપૂર્ણ સેટ વ્યાખ્યાયિત કરેલો છે. sqrt() વિધેય Math ક્લાસની એક સ્ટેટિક મેથડ સભ્ય છે જે Math.sqrt() થી ઇન્વોક (invoke) કરી શકાય છે.

લેબલ કરેલા લૂપ અને લેબલ કરેલા break (Labelled Loops and Labelled Break)

જ્યારે આપણે નેસ્ટેડ લૂપ વાપરીએ છીએ, ત્યારે તેમાં રહેલું break વિધાન કે જે સૌથી નજીકના લૂપમાં સમાયેલું છે તેને અટકાવી લૂપની બહાર નિયંત્રણ સ્થાનાંતર કરે છે. એ જ પ્રમાણે continue વિધાન તે જે લૂપમાં સમાયેલું છે તે ફરી શરૂ કરે છે.

જો આપણે લૂપ અટકાવવું હોય અથવા ક્યા લૂપનું પુનરાવર્તન કરવું તેનું નિયંત્રણ કરવા ઇચ્છતા હોઈએ તો આપણે લેબલ કરેલા લૂપનો ઉપયોગ કરી શકીએ છીએ. લેબલ કરેલું લૂપ વાપરવા માટે લૂપની શરૂઆત પહેલાં લેબલ (label) અને તેના પછી ગુરુવિરામ (:) લખવું પડે. જે લૂપમાં હોય તેની બહાર નિયંત્રણ લઈ જવું હોય ત્યારે break અથવા continue કીવર્ડ (keyword) પછી લેબલનું નામ લખવામાં આવે છે. આકૃતિ 7.15માંનો પ્રોગ્રામ લેબલ સાથેના લૂપનો ઉપયોગ દર્શાવે છે.

જ્યારે આકૃતિ 7.15ના કોડનો અમલ થાય છે, ત્યારે i*x ની કિંમત 350 થાય છે, જ્યારે i=5 અને x = 70 હોય. જ્યારે while લૂપની અંદર રહેલાં ifની શરતનું પરિણામ true મળે ત્યારે 'break out;' વિધાનનો અમલ થાય છે. તે બહારનું for લૂપ કે જે 'out' નામથી લેબલ કરેલું છે તેનો અંત લાવશે. જો ફક્ત break વિધાન જ વાપર્યું હોય તો તે જે while લૂપની અંદર છે તેમાંથી જ બહિર્ગમન કરતા અને for લૂપના પછીના પુનરાવર્તનથી ચાલુ રહેશે. અહીં, બહારનું for લૂપ i=6 અને તેનાથી વધારે કિંમત માટે અમલમાં આવશે નહીં.

LblBlock.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 LblBlock.java

```
class LblBlock  // Use of labelled block to break the outer loop
{
    public static void main (String[] s)
    {
        int x=0;
        out: for (int i = 4; i <10; i++)
        {
            x=10;
            while (x < 100)
            {
                System.out.println ( "Inside while loop: i is "
                                     + i + ", x is " + x);
                if ( i * x == 350 )
                    break out;
                x = x+20;
            } // end while
            System.out.println ( "\nOutside while loop: i is "
                                 + i + ", x is " + x + "\n");
        } // end out for loop
        System.out.println ( "\nOutside for loop: x is " + x);
    } // end of main
} // end of class
```

```
>javac LblBlock.java
>Exit code: 0
>java -cp . LblBlock
Inside while loop: i is 4, x is 10
Inside while loop: i is 4, x is 30
Inside while loop: i is 4, x is 50
Inside while loop: i is 4, x is 70
Inside while loop: i is 4, x is 90

Outside while loop: i is 4, x is 110

Inside while loop: i is 5, x is 10
Inside while loop: i is 5, x is 30
Inside while loop: i is 5, x is 50
Inside while loop: i is 5, x is 70
Outside for loop: x is 70
>Exit code: 0
```

આકૃતિ 7.15 : લેબલ કરેલા લૂપ અને breakનો ઉપયોગ

હવે આપણે પ્રોગ્રામમાં લેબલ કરેલા લૂપ અને breakનું બીજું ઉદાહરણ લઈએ જે 40 થી 100 વચ્ચેની પ્રથમ એકી વિભાજ્ય (non-prime) સંખ્યા આવે ત્યાં સુધીની અવિભાજ્ય સંખ્યાઓ (prime numbers) છાપે. જુઓ આકૃતિ 7.16.

```
LblLoop.java - SciTE
File Edit Search View Tools Options Language Buffers Help
1 LblLoop.java

class LblLoop  // determine first non-prime odd number in the range 41 to 100
{
    public static void main (String[] s)
    {
        boolean prime=true;
        int i, last, n;

        // break the loops as soon as first non-prime odd number is found
        forLoop: for (n=41; n<100; n=n+2) // no need to try even numbers >2
        {
            if ( n < 4)
            {
                prime=true;
                System.out.println(n);
                continue;
            }

            prime=true;
            i=3; last = (int)Math.sqrt(n);
            do
            {
                if  (n%i == 0) // n is divisible by i
                {
                    prime=false;
                    break forLoop;      // break for loop labelled 'forLoop'
                }
                i = i + 2;      // no need to divide by even numbers
            } while (prime  && (i<last)) ; //  end of do...while loop
            if (prime) System.out.println (n + " is prime");
        } // end of for loop
        if (!prime) System.out.println (n + " is not prime");
    } // end of main
} // end of class
```

```
>javac LblLoop.java
>Exit code: 0
>java -cp . LblLoop
41 is prime
43 is prime
45 is not prime
>Exit code: 0
```

આકૃતિ 7.16 : 40થી 100 વચ્ચેની અભાજ્ય સંખ્યા છાપતો પ્રોગ્રામ

સારાંશ

આ પ્રકરણમાં આપણે જાવાની મૂળભૂત બાબતો વિશે ચર્ચા કરી. જાવામાં પ્રાથમિક ડેટા પ્રકારો આઠ પ્રકારના છે. જાવામાં character એ 2 બાઇટનો યુનિકોડ અક્ષર છે. જાવામાં અનેક પ્રકારનાં લિટરલ વપરાય છે. ન્યુમરિક લિટરલ પૂર્વનિર્ધારિત રીતે double પ્રકારનાં છે. C ભાષા જેવા જ કન્ટ્રોલ-સ્ટ્રક્ચર જેમકે if, switch, for લૂપ, while લૂપ અને do...while લૂપ ઉપલબ્ધ છે. છગડિયા કૌંસ { }ની વચ્ચે વિધાનો લખીને બ્લૉકનો નિર્દેશ કરી શકાય છે. ચલના કાર્યક્ષેત્રની મર્યાદા (scope) તે બ્લૉક પૂરતી જ છે જ્યાં તે વ્યાખ્યાયિત કરેલા છે.

સ્વાધ્યાય

1. જાવામાં character ડેટાપ્રકારનું કદ કેટલું હોય છે ?
2. જાવામાં int અને long ડેટાપ્રકાર કેટલી બાઈટ રોકે છે તે જણાવો.
3. જાવામાં ઉપલબ્ધ if અને switch વિધાન સમજાવો.
4. જાવામાં ઉપલબ્ધ વિવિધ પુનરાવર્તિત માળખાં વિશે ચર્ચા કરો.
5. જાવામાં બ્લૉક અથવા લૂપ માટે લેબલ કઈ રીતે વાપરી શકાય ?

6. નીચેનામાંથી ખરા વિકલ્પની પસંદગી કરો :

(1) જાવામાં કેટલી પ્રાથમિક (primitive) ડેટાટાઇપ હોય છે ?

(a) 2 (b) 4 (c) 8 (d) 16

(2) અપૂર્ણાંક લિટરલનો પૂર્વનિર્ધારિત ડેટાપ્રકાર ક્યો હોય છે ?

(a) int (b) long (c) float (d) double

(3) જાવામાં char ડેટાપ્રકાર માટે ક્યો કૅરેક્ટરસેટ (characterset) વપરાય છે ?

(a) Unicode (b) ASCII

(c) EBCIDIC (d) ઉપરના તમામ વિકલ્પ

(4) નીચેનામાંથી ક્યું વિધાન કોઈ પણ ભૂલ વિના કંપાઇલ થશે ?

(a) for(;;){int i=7}; (b) while (1){int i=7};

(c) while (True){int i=7}; (d) ઉપરના તમામ વિકલ્પ

(5) નીચેનામાંથી માન્ય ચલના નામનો પહેલો અક્ષર કોણ ન હોય શકે ?

(a) અન્ડરસ્કોર (_) (b) અંક (c) અક્ષર (d) ડૉલર ($)

(6) નીચેનામાંથી ક્યો પ્રાથમિક ડેટાપ્રકાર નથી ?

(a) char (b) long (c) byte (d) string

(7) બુલિયન ડેટાપ્રકારની પૂર્વનિર્ધારિત કિંમત શું હોય છે ?

(a) નલ (null) (b) true (c) false (d) 0

(8) અંકગણિતની પદાવલી 7/2નું શું પરિણામ મળશે ?

(a) 3 (b) 3.5 (c) 1 (d) 0

(9) અંકગણિતની પદાવલી -7%2નું શું પરિણામ મળશે ?

(a) -3 (b) -1 (c) 1 (d) -3.5

(10) અંકગણિતની પદાવલી -7.5%2નું શું પરિણામ મળશે ?

(a) -3 (b) -1.5 (c) 1.5 (d) ભૂલ (Error)

પ્રાયોગિક સ્વાધ્યાય

નીચે જણાવેલાં કાર્ય માટે જાવાપ્રોગ્રામ લખો :

1. સ્ટોરમાં સેલ સમયે ₹ 5000ની ખરીદી ઉપર 10% વળતર આપવામાં આવે છે. એક પ્રોગ્રામ લખો, જે 'purchase' નામના ચલને કોઈ કિંમત ઍસાઈન કરે અને પછી વળતરબાદની કિંમતની ગણતરી કરે. ખરીદકિંમત અને આપેલ વળતર પ્રદર્શિત કરો.

2. એક પ્રોગ્રામ લખો, જે ગ્રાહકની ઉંમર અને ચલચિત્રના શૉના સમય (મેટિની કે અન્ય શો) પ્રમાણે ચલચિત્રની ટિકિટની કિંમત શોધે. વયસ્ક માટે નૉર્મલ શૉ અને મેટિની-શૉની ટિકિટ અનુક્રમે ₹ 100 અને ₹ 50 છે. વયસ્ક એ છે, જેની ઉંમર 13 વર્ષ કરતાં વધારે છે. બાળકો માટે નૉર્મલ શૉ અને મેટિની-શૉની ટિકિટ અનુક્રમે ₹ 60 અને ₹ 40 છે. ઉંમર (age) અને શૉના સમય (show time) માટે યોગ્ય ડેટાપ્રકારના ચલ ઘોષિત કરો, આ ચલને કિંમત ઍસાઈન કરો અને ઉંમર, શૉ-ટાઈમ અને ટિકિટ-કિંમત છાપો.

3. switch વિધાનનો ઉપયોગ કરીને મેળવેલા ગુણ પ્રમાણે ગ્રેડ પ્રદર્શિત કરતો પ્રોગ્રામ લખો.

4. એક બૅન્ક ગ્રાહકોને વાર્ષિક 12%ના સાદા વ્યાજથી ધિરાણ આપે છે. પૂર્ણસમયનું વ્યાજ શરૂઆતમાં વસૂલવામાં આવે છે. માસિક હપ્તાની ગણતરી 36 માસના સમયગાળા અને ધિરાણ-રકમ ₹ 10000, 20000, 30000, 40000, 50000, 60000, 70000, 80000, 90000 અને 100000 માટે કરો.

5. એક પ્રોગ્રામ લખો, જે પૂર્ણાંક સંખ્યાઓ 5થી શરૂ કરી, જેનું વર્ગમૂળ 50 કે તેથી ઓછું હોય, તેવી સંખ્યાઓનું વર્ગમૂળ છાપે.

જાવામાં ક્લાસ અને ઑબ્જેક્ટ 8

આપણે અગાઉ શીખી ગયા છીએ કે ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગમાં ઑબ્જેક્ટ (object) અને ક્લાસ (class) એ પાયાના એકમો છે. જાવા ઑબ્જેક્ટ આધારિત ભાષા છે. આ પ્રકરણ જાવા-પ્રોગ્રામિંગમાં ક્લાસ અને ઑબ્જેક્ટ કઈ રીતે બનાવવા તે સમજાવે છે. આ પ્રકરણના અભ્યાસ પછી જાવામાં ઑબ્જેક્ટ શું છે અને ક્લાસ શું છે, તેનું સ્પષ્ટ ચિત્ર મેળવવા તમે સમર્થ હશો.

પરિચય (Introduction)

ક્લાસ (class) ડેટા (જેને ઍટ્રિબ્યૂટ કહે છે) અને પ્રોગ્રામનો કોડ (વિધેય, જેને મેથડ કહે છે) બન્ને ધરાવે છે.

અગાઉના પ્રકરણમાં આપણે ક્લાસનો ઉપયોગ કરીને 'main' નામની એક જ મેથડ ધરાવતો પ્રોગ્રામ લખ્યો. જ્યારે ક્લાસનો અમલ કરવામાં આવ્યો, ત્યારે main મેથડ વાપરીને એક સામાન્ય પ્રોગ્રામની જેમ વિનિયોગ ચાલ્યો. આપણે આ ઉદાહરણોમાં કોઈ પણ ડેટા મેમ્બર (data members)નો ઉપયોગ કર્યો ન હતો.

જાવા પ્રોજેક્ટમાં ફક્ત એક જ ક્લાસ વાપરવો શક્ય છે, પણ મોટા વિનિયોગ માટે તે રીત સારી નથી. સૉફ્ટવેર ડિઝાઇન કરતા સમયે સંપૂર્ણ વિનિયોગને સરળ ઘટકોમાં વિભાજિત કરી દેવા જોઈએ કે જે તાર્કિક રીતે સંબંધિત કાર્યો કરે. આ દરેક ઘટક કે ભાગનો આપણે એક ક્લાસ બનાવી શકીએ.

હવે, આપણે 'Room' નામના ઉદાહરણથી જાવા પ્રોગ્રામિંગમાં ક્લાસ અને ઑબ્જેક્ટના ખ્યાલ વિશે સમજીએ. ઘર, છાત્રાલય, હોટેલ અથવા નિશાળના ખંડ (રૂમ)ની લાક્ષણિકતાઓ એકસમાન હોય છે. દરેક ખંડ તેની પ્રૉપર્ટી (property) જેમ કે length, height, number of windows, number of doors અને directionથી અજોડ રીતે ઓળખાય છે. સરળતા માટે આપણે અહીં length, width, height અને number of windows લઈશું.

હવે આપણે કોડલિસ્ટિંગ 8.1માં દર્શાવ્યા પ્રમાણે જાવાપ્રોગ્રામ લખીશું, જે length, width, height અને nWindows ઍટ્રિબ્યૂટ સાથેનો 'Room' નામનો ક્લાસ અને ત્રણ મેથડ બનાવશે. પહેલી મેથડ ઍટ્રિબ્યૂટને કિંમત ઍસાઇન કરશે. બીજી મેથડ ક્ષેત્રફળની ગણતરી કરશે અને ત્રીજી મેથડ તેના ઍટ્રિબ્યૂટ પ્રદર્શિત કરશે. તે પછી આપણે આ ક્લાસનો ઉપયોગ કરવા માટે કોડ લખીશું.

```
/* Class Room */
class Room
{
        float length, width, height;
        byte nWindows;

        void setAttr (float l, float w, float h, byte n)
        {
                length = l; width = w; height = h;
                nWindows = n;
        }  // end setAttr () method

    double area ( )  // area =  length * width
```

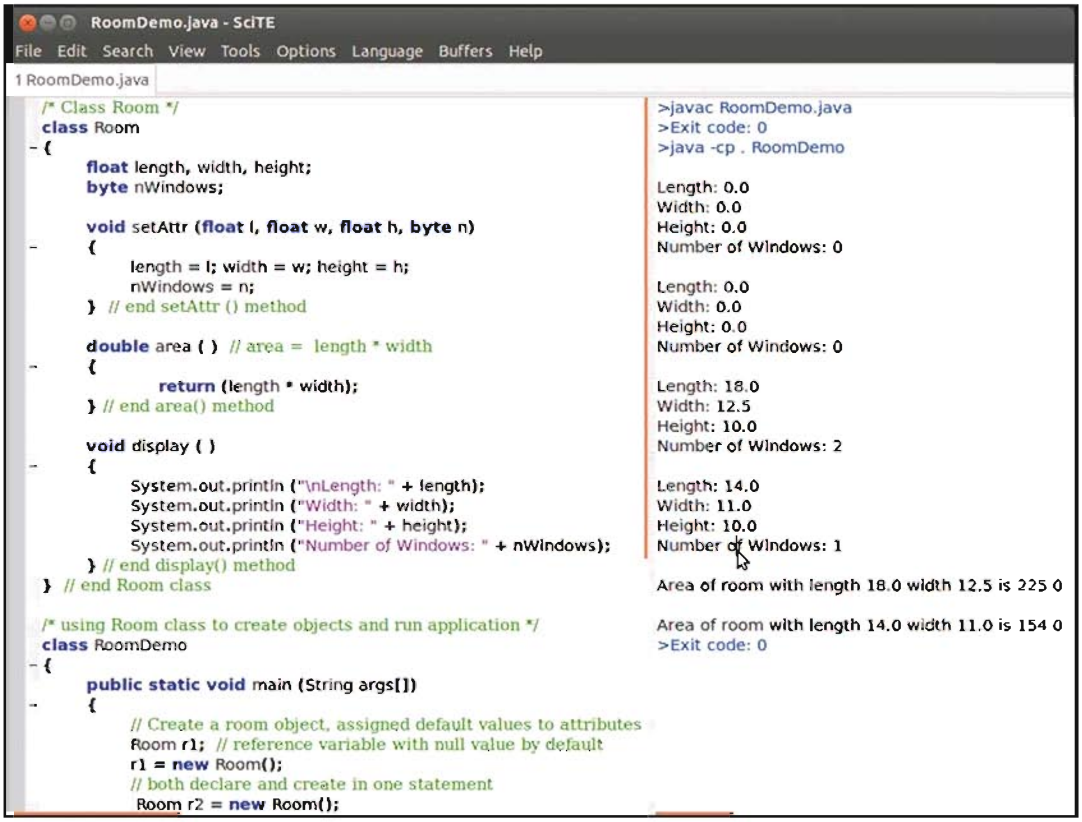
```
        {
                return (length * width);
        } // end area() method

        void display ( )
        {
                System.out.println ("\nLength: " + length);
                System.out.println ("Width: " + width);
                System.out.println ("Height: " + height);
                System.out.println ("Number of Windows: " + nWindows);
        } // end display() method
}  // end Room class

/* using Room class to create objects and run application */
class RoomDemo
{
        public static void main (String args[])
        {
                // Create a room object, assigned default values to attributes
                Room r1;  // reference variable with null value by default
                r1 = new Room();
                // both declare and create in one statement
                 Room r2 = new Room();

            // Display two room objects with initial default values
                r1.display();
                r2.display();

                // Assign values of attributes of objects
                r1.setAttr (18, 12.5f, 10, (byte)2);
                r2.setAttr (14, 11, 10, (byte)1);

                // Display updated contents
                r1.display();
                r2.display();

                // Display area
                System.out.println ("\nArea of room with length " + r1.length
                    + " width " + r1.width +  " is " + r1.area());
                System.out.println ("\nArea of room with length " + r2.length
                    + " width " + r2.width +  " is " + r2.area());
        }  // end main()
}  // end RoomDemo
```

કોડલિસ્ટિંગ 8.1 : ક્લાસ અને ઑબ્જેક્ટ બનાવવા અને તેનો ઉપયોગ કરવો

અહીં, સૌપ્રથમ, 'Room' નામનો ક્લાસ બનાવવા માટે કોડ લખેલો છે. તે પછી 'Room' ક્લાસના ઑબ્જેક્ટ બનાવવા માટેનો કોડ લખેલો છે અને તેથી મેથડ (method) વાપરેલી છે. આ કાર્ય કરવા માટે main( ) મેથડ ધરાવતો એક બીજો ક્લાસ 'RoomDemo' બનાવેલો છે. સૉર્સફાઇલમાં આ ક્લાસ કોઈ પણ ક્રમમાં હોઈ શકે.

અહીં, આપણે તાર્કિક રીતે સંબંધિત કાર્યો કરતા 'Room' ક્લાસ બનાવવા અને આ ક્લાસનો 'RoomDemo' નામના ક્લાસના વિનિયોગમાં ઉપયોગ એ બંને અલગ કરેલા છે.

જ્યારે એક પ્રોગ્રામમાં બે અથવા વધારે ક્લાસ હોય, ત્યારે ફક્ત એક જ ક્લાસ main( ) મેથડ ધરાવી શકે. આકૃતિ 8.1માં SciTE એડિટરમાં કોડનો અમલ દર્શાવ્યો છે.



આકૃતિ 8.1 : એક કરતાં વધારે ક્લાસના ઉપયોગનું પ્રદર્શન કરવા માટે જાવાપ્રોગ્રામ

આકૃતિ 8.1માં દર્શાવ્યા પ્રમાણે RoomDemo વિનિયોગના આપેલા ઉદાહરણમાં નીચે જણાવેલાં કાર્ય કરવામાં આવે છે.

- સૌપ્રથમ Room ક્લાસના બે ઑબ્જેક્ટ r1 અને r2 બનાવવામાં આવેલા છે. પૂર્વનિર્ધારિત કિંમત તેને આપવામાં આવે છે (initialized). (પૂર્વનિર્ધારિત રીતે આંકડાકીય કિંમત શૂન્ય છે.)
- display મેથડ વડે આ બંને ઑબ્જેક્ટની માહિતી પ્રદર્શિત કરવામાં આવે છે.
- ન્યુમરિક લિટરલ પ્રાચલો સાથે setAttr() મેથડનો અમલ (invoke) કરીને બંને ઑબ્જેક્ટ r1 અને r2 ના એટ્રિબ્યૂટમાં ફેરફાર થાય છે.
- બંને ઑબ્જેક્ટની નવી માહિતી પ્રદર્શિત કરવામાં આવે છે.

અંતમાં બંને Room ઑબ્જેક્ટનું ક્ષેત્રફળ પ્રદર્શિત કરવામાં આવે છે. અહીં, ક્ષેત્રફળ શોધવા માટે area( ) મેથડનો અમલ કરેલી છે.

જાવામાં ક્લાસ (Class in Java)

આપણે અત્યાર સુધીમાં ક્લાસનો ઉપયોગ કરતો એક પ્રોગ્રામ લખેલો છે. હવે આપણે વાક્યરચના (syntax) અને ક્લાસ તથા ઑબ્જેક્ટ બનાવવા માટેની અન્ય માહિતી અને વિનિયોગમાં તેના ઉપયોગ વિશે શીખીએ.

ક્લાસ એ એકસમાન લાક્ષણિકતાઓ ધરાવતા અનેક ઑબ્જેક્ટનું એક માળખું (template) છે. ક્લાસ કોઈ ચોક્કસ ઑબ્જેક્ટના જૂથની લાક્ષણિકતાઓને સમાવે છે. ઉદાહરણ તરીકે, 'Room' એ બધા ખંડની સામાન્ય લાક્ષણિકતાઓ સાથેનું એક ટેમ્પ્લેટ છે.

જાવામાં class ચાવીરૂપ શબ્દ(keyword)નો ઉપયોગ કરીને ક્લાસને નીચે પ્રમાણે વ્યાખ્યાયિત કરવામાં આવે છે :

```
class <ClassName >
{
        <Variables>
        <Methods>
}
```

આપણે જાવામાં દરેક ક્લાસ બનાવીએ છીએ, તે સામાન્ય રીતે બે ઘટકોનો બનેલો હોય છે : ઍટ્રિબ્યૂટ (attribute) અને બિહેવ્યર (behaviour). ઍટ્રિબ્યૂટને ક્લાસમાં ચલથી વ્યાખ્યાયિત કરવામાં આવે છે. બિહેવ્યરને ક્લાસમાં મેથડથી વ્યાખ્યાયિત કરવામાં આવે છે. મેથડનો ઉપયોગ ઍટ્રિબ્યૂટને મેળવવામાં અને તેમાં ફેરફાર કરવા માટે થાય છે.

કોડલિસ્ટિંગ 8.1માં આપણે 'Room' નામનો ક્લાસ વ્યાખ્યાયિત કરેલો છે, જેના ઍટ્રિબ્યૂટ length, width, height અને nWindows નામના ચલથી વ્યાખ્યાયિત કરેલાં છે અને બિહેવ્યર setAttr(), display() અને area() નામની મેથડથી વ્યાખ્યાયિત કરેલા છે. 'Room' ક્લાસનો ઉપયોગ કરીને વિનિયોગ બનાવવા માટે એક અન્ય ક્લાસ બનાવ્યો છે.

ઑબ્જેક્ટ બનાવવા (Creating Objects)

ક્લાસમાંથી ઑબ્જેક્ટ બનાવવા માટે નીચે જણાવેલાં પગલાંઓની જરૂર પડે છે :

- **Declaration :** <class name> <variable name> વાક્યરચના (syntax) સાથેનો ક્લાસ પ્રકારનો ચલ ઘોષિત કરવામાં આવે છે.
- **Instantiation :** (ઇન્સ્ટન્સિએશન - દૃષ્ટાંતીકરણ) મેમરી ફાળવીને ઑબ્જેક્ટ બનાવવા માટે **new** ચાવીરૂપ શબ્દ વપરાય છે.
- **Initialization :** નવા બનાવેલા ઑબ્જેક્ટને કિંમત આપવા માટે (initialize) કન્સ્ટ્રક્ટર (constructor) (એક વિશિષ્ટ પ્રકારની મેથડ) કૉલ કરવામાં આવે છે.

જાવામાં ક્લાસ એ int અને boolean જેવા સમાવિષ્ટ પ્રકારના જેવો જ એક પ્રકાર છે. આથી, ક્લાસના નામનો ઉપયોગ ઘોષિત વિધાનમાં ચલના પ્રકારનો ઉલ્લેખ કરવા, ઔપચારિક પ્રાચલનો પ્રકાર જણાવવા અને વિધેયની પરત કિંમતનો પ્રકાર જણાવવા થાય છે.

'Room' નામના ક્લાસનો ઑબ્જેક્ટ ઘોષિત કરવા માટે નીચેનું વિધાન વાપરો :

```
Room r1;
```

ચલને ઘોષિત કરવાથી ઑબ્જેક્ટ બનતો નથી. આ એક મહત્ત્વનો મુદ્દો યાદ રાખવો જોઈએ. જાવામાં કોઈ પણ ચલ ક્યારેય ઑબ્જેક્ટનો સંગ્રહ કરી શકતો નથી. ક્લાસ પ્રકારના ઉપયોગ વડે ઘોષિત કરેલો ચલ ફક્ત ઑબ્જેક્ટના નિર્દેશનો સંગ્રહ કરે છે. આથી, ક્લાસ પ્રકારના ચલને નિર્દેશિત ચલ (reference variable) પણ કહેવામાં આવે છે. અહીં, r1 એ નિર્દેશિત ચલ છે.

હવે પછીનું પગલું છે ઑબ્જેક્ટ બનાવવાનું. new ચાવીરૂપ શબ્દનો ઉપયોગ કરીને આપણે ઑબ્જેક્ટ બનાવી શકીએ. પ્રક્રિયક new ઑબ્જેક્ટ માટેની મેમરી ફાળવશે અને ભવિષ્યમાં તેનો ઉપયોગ કરી શકાય તે માટે તે ઑબ્જેક્ટનો સ્થાનાંક (address) પરત કરશે. રેફરન્સ (reference) એ મેમરીનો સ્થાનાંક છે, જ્યાં ઑબ્જેક્ટનો સંગ્રહ કરેલો છે. હકીકતમાં, મેમરીનો ખાસ ભાગ કે જેને હીપ (heap) કહેવામાં આવે છે, જ્યાં ઑબ્જેક્ટ રાખવામાં આવે છે.

જ્યારે ઑબ્જેક્ટ બનાવવામાં આવે છે ત્યારે, મેમરીની ફાળવણી ઉપરાંત 'constructor' નામની વિશિષ્ટ મેથડ શરૂઆતના કાર્ય કરવા માટે અમલમાં મૂકવામાં આવે છે. આપણે કન્સ્ટ્રક્ટર (constructor) વિશે આ પ્રકરણમાં હવે પછી શીખીશું.

હવે, આપણે Roomના પ્રકારનો એક ઑબ્જેક્ટ બનાવીએ અને તેનો સ્થાનાંક ચલ r1 ને સોંપીએ (ઍસાઇન કરીએ) :

```
r1 = new Room();
```

અહીં કૌંસ અગત્યના છે, તેને છોડી દેશો નહીં. ચલ (આર્ગ્યુમેન્ટ) વગરના ખાલી કૌંસ પૂર્વનિર્ધારિત constructorને કૉલ કરશે. તે પૂર્વનિર્ધારિત કિંમતો વડે ઑબ્જેક્ટના ઍટ્રિબ્યૂટની પ્રારંભિક કિંમત તેમાં મૂકશે (initializes). કૌંસમાં આર્ગ્યુમેન્ટની કિંમતો પણ હોઈ શકે કે જે ચલની પ્રારંભિક કિંમત નક્કી કરે. આ વપરાશકર્તા નિર્મિત કન્સ્ટ્રક્ટરનો ઉપયોગ કરીને શક્ય છે.

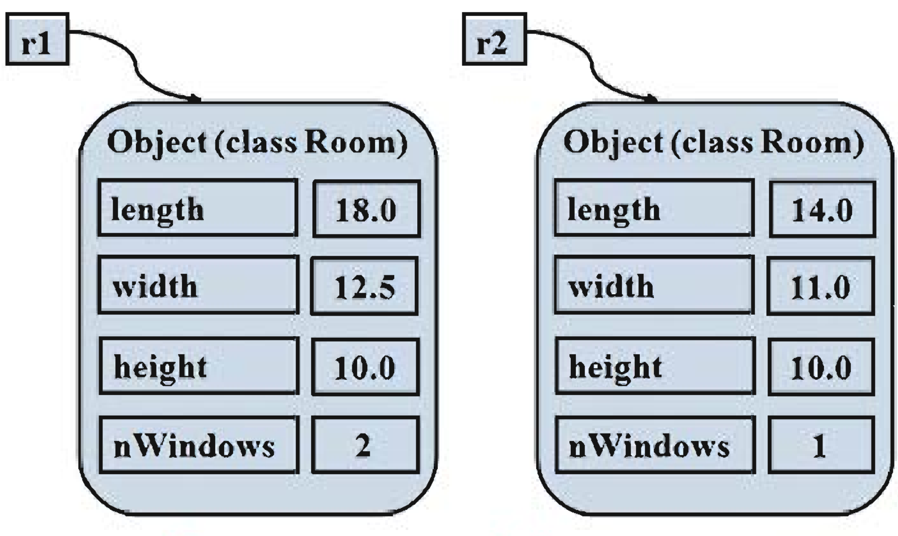
જ્યારે "r1=new Room();" વિધાનનો અમલ થાય છે, ત્યારે યાદ રાખો કે ઑબ્જેક્ટનો ચલ r1માં સંગ્રહ થતો નથી. ચલ r1 ઑબ્જેક્ટનો ફક્ત સ્થાનાંક (ઍડ્રેસ) જ ધરાવે છે.

ઑબ્જેક્ટ ઘોષિત કરવા અને બનાવવા માટે ઉપર જણાવેલાં બંને પગલાંને નીચે જણાવ્યા પ્રમાણે એક વિધાનમાં ભેગાં કરી શકાય છે :

```
Room r2 = new Room();
```

અહીં, ચલ r2 મેમરી સ્થાનાંક ધરાવે છે, જ્યાં નવો ઑબ્જેક્ટ બનાવેલો છે.

અહીં તે પણ નોંધવું જોઈએ કે ક્લાસ ફક્ત ચલનો પ્રકાર નક્કી કરે છે. ડેટા હકીકતમાં વ્યક્તિગત ઑબ્જેક્ટની અંદર હોય છે અને ક્લાસની અંદર નહીં. આ રીતે, દરેક ઑબ્જેક્ટને પોતાનો ડેટાનો સેટ હોય છે. કોડ લિસ્ટિંગ 8.1માં, આપણે new પ્રક્રિયકનો ઉપયોગ કરીને બે ઑબ્જેક્ટ r1 અને r2 બનાવેલા છે. આકૃતિ 8.2માં દર્શાવ્યા પ્રમાણે આ બંને ઑબ્જેક્ટને તેમની ડેટાની કિંમતને રાખવા માટે અલગ-અલગ મેમરીની ફાળવણી કરવામાં આવેલી છે.



આકૃતિ 8.2 : Room ક્લાસના બે ઈન્સ્ટન્સ (instance)

જાવામાં જ્યારે ઑબ્જેક્ટની પછી જરૂર રહેતી નથી, ત્યારે તે મેમરી ફરી વાપરવા માટે છૂટી કરવામાં આવે છે. જાવામાં ગાર્બેજ કલેક્ટર (garbage collector) હોય છે કે જે વણવપરાયેલા ઑબ્જેક્ટને શોધે છે અને તે આ ઑબ્જેક્ટ દ્વારા વપરાયેલી મેમરી પાછી મેળવે છે (એટલે કે છૂટી કરે છે). આપણે મેમરી મુક્ત કરવા માટે બાહ્ય રીતે કશું કરવાની જરૂર નથી.

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ (OOP) ભાષામાં ઑબ્જેક્ટ બનાવવાની ક્રિયાને **ઑબ્જેક્ટ ઈન્સ્ટન્સિએશન (object instantiation)** પણ કહેવામાં આવે છે. ઑબ્જેક્ટ માટે ડેટાનો સંગ્રહ કરવા માટે મેમરી ફાળવીને ઑબ્જેક્ટનો ઇન્સ્ટન્સ (instance) નિર્મિત કરવામાં આવે છે. કોઈ ક્લાસનો જે ઑબ્જેક્ટ ભાગ હોય, તેને ક્લાસનો **ઈન્સ્ટન્સ (instance)** કહેવામાં આવે છે.

આ રીતે, ક્લાસનો ઇન્સ્ટન્સ એ હકીકતમાં ઑબ્જેક્ટ માટેનો બીજો શબ્દ છે. ક્લાસ એ ઑબ્જેક્ટનું અમૂર્ત (abstract) સ્વરૂપ છે, જ્યારે ઇન્સ્ટન્સ એ મૂર્ત કે સાકાર સ્વરૂપ છે. હકીકતમાં, OOP ભાષામાં ઇન્સ્ટન્સ અને ઑબ્જેક્ટ શબ્દો ઘણી વખત એકબીજાની અદલાબદલીમાં વપરાય છે.

ક્લાસનો દરેક ઇન્સ્ટન્સ ક્લાસમાં ઘોષિત કરેલા ચલના ઍટ્રિબ્યૂટ માટે અલગ-અલગ કિંમત ધરાવી શકે છે. આ પ્રકારના ચલનો **ઇન્સ્ટન્સ વેરિયેબલ** (instance variable) તરીકે ઉલ્લેખ કરવામાં આવે છે. ઑબ્જેક્ટ બનાવતા સમયે ઇન્સ્ટન્સ વેરિયેબલનું નિર્માણ થાય છે અને ઑબ્જેક્ટના સંપૂર્ણ અસ્તિત્વ સુધી તે રહે છે.

ઇન્સ્ટન્સ વેરિયેબલ ઑબ્જેક્ટનાં ઍટ્રિબ્યૂટ વ્યાખ્યાયિત કરે છે. ક્લાસ ઍટ્રિબ્યૂટનો પ્રકાર વ્યાખ્યાયિત કરે છે. દરેક ઇન્સ્ટન્સ તે ઍટ્રિબ્યૂટની પોતાની કિંમતનો સંગ્રહ કરે છે.

ઑબ્જેક્ટના બિહેવ્યરને વ્યાખ્યાયિત કરવા માટે આપણે મેથડ બનાવીએ છીએ. જાવામાં મેથડ ફક્ત ક્લાસની અંદર જ વ્યાખ્યાયિત કરી શકાય છે. આ મેથડ ઇન્સ્ટન્સ વેરિયેબલની કિંમત મેળવવા અથવા બદલવા માટે ઑબ્જેક્ટનો ઉપયોગ કરીને ઇન્વૉક (invoke) કરી શકાય છે. આવી મેથડ **ઇન્સ્ટન્સ મેથડ** તરીકે ઓળખાય છે.

ઇન્સ્ટન્સ મેથડ ઑબ્જેક્ટનું બિહેવ્યર વ્યાખ્યાયિત કરવા માટે વપરાય છે. મેથડ ઇન્વૉક કરવી એ ઑબ્જેક્ટને કોઈ કાર્ય કરવા માટેનો હુકમ છે.

કોડલિસ્ટિંગ 8.1માં આપણે Room નામનો ક્લાસ length, width, height અને nWindows નામના ઇન્સ્ટન્સ વેરિયેબલ અને setAttr(), display() તથા area() નામની ઇન્સ્ટન્સ મેથડ સાથે વ્યાખ્યાયિત કરેલો છે.

સારાંશ રૂપે,

- new કી વર્ડથી ઑબ્જેક્ટ બનાવી શકાય છે.
- new કી વર્ડ ક્લાસના ઇન્સ્ટન્સને રજૂ કરતો ઑબ્જેક્ટનો નિર્દેશ પરત કરે છે.
- ક્લાસના બધા ઇન્સ્ટન્સની હીપ (heap) તરીકે ઓળખાતા ડેટા સ્ટ્રક્ચરમાં મેમરીની ફાળવણી થાય છે.
- દરેક ઑબ્જેક્ટ ઇન્સ્ટન્સનો પોતાનો ડેટા સેટ હોય છે.

ઇન્સ્ટન્સ વેરિયેબલ ઍક્સેસ કરવા (કાર્ય કરવા માટે મેળવવા) અને ઇન્સ્ટન્સ મૅથડ કોલ કરવી (Accessing instance variables and calling instance methods)

ઇન્સ્ટન્સ વેરિયેબલ અને ઇન્સ્ટન્સ મેથડ ઑબ્જેક્ટ દ્વારા મેળવવામાં (access) આવે છે. તેનો નિર્દેશ નીચે જણાવ્યા પ્રમાણે ડોટ પ્રક્રિયક (.) દ્વારા કરી શકાય છે :

<object refernce>.<instance variable or method>

ઉદાહરણ તરીકે, આપણે પ્રોગ્રામમાં Room r1ની lengthનો ઉલ્લેખ r1.length વાપરીને કરી શકીએ છીએ અને કોડલિસ્ટિંગ 8.1માં દર્શાવ્યા પ્રમાણે r1.display( ) મેથડને ઇન્વૉક કરી Room r1ની ઍટ્રિબ્યૂટની કિંમતોને પ્રદર્શિત કરી શકીએ છીએ. અહીં એ નોંધ કરશો કે ડોટ (.) એ એક પ્રક્રિયક છે અને ડોટ પ્રક્રિયકની સહચારિતા (એસોસિએટિવિટી) ડાબી બાજુથી જમણી બાજુ હોય છે.

અહીં એ યાદ રાખવું જોઈએ કે પ્રોગ્રામમાં કોઈ પણ જગ્યાએથી ડેટા આ રીતે સીધો મેળવવાથી સુરક્ષિત હોવો જોઈએ. આ જાતનું રક્ષણ ઍક્સેસ મોડીફાયર (access modifier) વડે શક્ય છે, જેની ચર્ચા આપણે હવે પછી કરીશું.

જ્યારે આપણે એક જ ક્લાસની મેથડની અંદર ઇન્સ્ટન્સ વેરિયેબલ વાપરીએ છીએ, ત્યારે ડોટ (.) વાપરવાની કોઈ જરૂર નથી. ઉદાહરણ તરીકે, કોડલિસ્ટિંગ 8.1માં display મેથડમાં ડોટપ્રક્રિયકના ઉપયોગ વિના ઇન્સ્ટન્સ વેરિયેબલને ઍક્સેસ કરેલાં છે. અહીં એ શક્ય છે, કારણકે રેફરન્સ વેરિયેબલ r1નો ઉપયોગ કરીને મેથડ ઇન્વૉક કરેલી છે અને આ રીતે r1 ઉપર સંગ્રહ કરેલા ઑબ્જેક્ટના અનુસંધાને ઉલ્લેખ કરેલાં ઇન્સ્ટન્સ વેરિયેબલ માનવામાં આવે છે.

આપણે અગાઉ શીખ્યા તે પ્રમાણે જ્યારે આપણે ક્લાસનો ઉપયોગ કરીને ફક્ત ઑબ્જેક્ટ ઘોષિત કરીએ છીએ, ત્યારે ઑબ્જેક્ટ

બનતો નથી. આ કિસ્સામાં રેફરન્સ વેરિયેબલ કોઈ ઑબ્જેક્ટનો નિર્દેશ કરતો નથી અને તેની પ્રારંભિક કિંમત પૂર્વનિર્ધારિત રીતે નલ (null) હોય છે. આ પ્રકારના નલ રેફરન્સ અથવા નલ પૉઇન્ટરનો ઉપયોગ અમાન્ય છે અને આથી નલ રેફરન્સ સાથેના ઇન્સ્ટન્સ વેરિયેબલ કે ઇન્વૉકિંગ મેથડ ભૂલ (error) આપશે. નીચે આપેલો કોડ વાપરી જુઓ :

```
Room r1; // null value assigned to reference variable by default
System.out.println (r1.length); // illegal
r1.display();  // illegal
```

ક્લાસ-વેરિયેબલ અને ક્લાસ-મેથડ (Class Variables and Class Methods)

આપણે અગાઉ ચર્ચા કરી તે પ્રમાણે જ્યારે new કી વર્ડનો ઉપયોગ કરીને ઑબ્જેક્ટ બનાવવામાં આવે છે ત્યારે તેના ઍટ્રિબ્યૂટની કિંમતોનો સંગ્રહ કરવા માટે હીપ (heap) વિસ્તારમાંથી મેમરીની ફાળવણી કરવામાં આવે છે. આ રીતે દરેક ઑબ્જેક્ટનાં પોતાના ઇન્સ્ટન્સ વેરિયેબલ હોય છે જે મેમરીમાં અલગ-અલગ જગ્યા રોકે છે.

હવે ધારો કે આપણે અત્યાર સુધીમાં બનાવેલા બધા Room ઑબ્જેક્ટની windowsની કુલ સંખ્યા આપણને જોઈએ છે. આ કિંમતનો સંગ્રહ કરવા માટે દરેક ક્લાસ દીઠ આપણે ફક્ત એક ચલની જરૂર પડશે. આ ચલને દરેક ઑબ્જેક્ટના ઍટ્રિબ્યૂટ તરીકે રાખવો એ અર્થહીન છે. હકીકતમાં તે ઑબ્જેક્ટનો ઍટ્રિબ્યૂટ નથી પણ તે ક્લાસનો ઍટ્રિબ્યૂટ છે.

આથી જ્યારે કેટલાક ચલની જરૂર હોય અને તે એક જ ક્લાસના બધા ઑબ્જેક્ટ ઇન્સ્ટન્સ વચ્ચે વાપરવાના હોય, તો દરેક ક્લાસ દીઠ ચલને મેમરી ફક્ત એક વાર જ ફાળવવી જોઈએ. આનો અર્થ એ થાય કે ચલ ક્લાસનો ભાગ છે અને ઑબ્જેક્ટનો નહીં. આવા ચલ ડેટાના પ્રકાર પહેલાં static કી વર્ડ વાપરી ને ક્લાસમાં ઘોષિત કરવામાં આવે છે અને આ સ્ટેટિક વેરિયેબલ (static variables) ક્લાસ વેરિયેબલ (class variable) કહેવાય છે.

ઇન્સ્ટન્સ વેરિયેબલની કિંમત ઇન્સ્ટન્સ (ઑબ્જેક્ટ માટે ફાળવવામાં આવેલી મેમરી)માં સંગ્રહ કરવામાં આવે છે, જ્યારે ક્લાસ-વેરિયેબલની કિંમત ક્લાસના પોતાનામાં જ સંગ્રહ કરે છે.

ઇન્સ્ટન્સ વેરિયેબલ સાથેના ક્લાસમાં નીચેનું વિધાન ઉમેરીને totWindows નામનો ક્લાસ-વેરિયેબલ ઘોષિત કરીએ :

```
static int totWindows;
```

સ્ટેટિક વેરિયેબલને ક્લાસના ઇન્સ્ટન્સ બનાવ્યા વિના મેળવી શકાય છે. (ઍક્સેસ કરી શકાય છે.) ઉદાહરણ તરીકે, 'Room' નામના ક્લાસમાં કોઈ પણ ઑબ્જેક્ટ બનાવ્યા વિના totwindowsની કિંમત પ્રદર્શિત કરવા આપણે પ્રયત્ન કરીશું, તો તે 0 પ્રદર્શિત કરશે.

ક્લાસ વેરિયેબલની ક્લાસમેથડ મેથડ ડેફિનેશન (method definition) પહેલાં static ચાવીરૂપ શબ્દ લખીને ઘોષિત કરી શકાય છે. કુલ બારીની સંખ્યા પ્રદર્શિત કરવા 'Room' ક્લાસની નીચેની મેથડ પ્રયત્ન કરી જુઓ.

```
static void displayTotalWindows()
{
    System.out.println("Total Windows: " +totWindows);
}
```

ક્લાસની બહાર <classname>.< class variable/method name>નો ઉપયોગ કરીને ક્લાસ વેરિયેબલ અને ક્લાસ મેથડ આપણે વાપરી શકીએ છીએ.

ઉદાહરણ તરીકે : Room.totWindows, Room.displayTotalWindows()

અહીં એ નોંધ કરશો કે આકૃતિ 8.3માં કોડલિસ્ટિંગમાં જણાવ્યા પ્રમાણે ક્લાસના નામ વિના એક જ ક્લાસની મેથડનો ઉપયોગ કરી ક્લાસના સભ્યો (વેરિયેબલ અને મેથડ)નો ઉલ્લેખ કરી શકાય છે.

```
RoomDemoStatic.java - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 RoomDemoStatic.java

/* Class Room */
class Room
{
      float length, width, height;
      byte nWindows;
      static int totWindows;  // class variable

      void setAttr (float l, float w, float h, byte n)
      {
            setWindows(n);
            length = l; width = w; height = h;
      } // end setAttr () method

      void setWindows ( byte n)
      {
            totWindows = totWindows - nWindows + n;
            nWindows = n;
      }
      double area ( ) // area =  length * width
      {     return (length * width);      } // end area() method

      void display ( )
      {
            System.out.println ("\nLength: " + length + "\nWidth: " + width);
            System.out.println ("Height: " + height);
            System.out.println ("Number of Windows: " + nWindows);
      } // end display() method
} // end Room class

/* using Room class to create objects and run application */
class RoomDemoStatic
{
      public static void main (String args[])
      {
            Room r1 = new Room();   Room r2 = new Room();

            r1.setAttr (18, 12.5f, 10, (byte)2);  r1.display();
            r2.setAttr (14, 11, 10, (byte)1);  r2.display();
            System.out.println("\nRoom2 Number of windows modified from 1 to 2");
            r2.setWindows((byte)2);
            System.out.println ("\nTotal number of Windows: " + Room.totWindows);
      } // end main()
} // end RoomDemo
```

આકૃતિ 8.3 : ક્લાસ-વેરિયેબલ totWindowsનો ઉપયોગ

આકૃતિ 8.3માં આપેલા કોડલિસ્ટિંગનો નિર્ગમ (આઉટપુટ) આકૃતિ 8.4માં આપેલો છે.

```
RoomDemoStatic.java - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 RoomDemoStatic.java

>javac RoomDemoStatic.java
>Exit code: 0
>java -cp . RoomDemoStatic

Length: 18.0
Width: 12.5
Height: 10.0
Number of Windows: 2

Length: 14.0
Width: 11.0
Height: 10.0
Number of Windows: 1

Room2 Number of windows modified from 1 to 2

Total number of Windows: 4
>Exit code: 0
```

આકૃતિ 8.4 : આકૃતિ 8.3માં આપેલા કોડલિસ્ટિંગનો નિર્ગમ

અહીં, આપણે એક વધારાની setWindows() નામની ઇન્સ્ટન્સ મેથડ લખી છે, જે બારીની કુલ સંખ્યામાં જૂની બારીની સંખ્યા બાદ કરીને અને નવી બારીની સંખ્યા ઉમેરીને સુધારો કરે છે. setWindows() મેથડ એ જ ક્લાસમાં વ્યાખ્યાયિત કરેલી છે. આથી ક્લાસના નામ વિના તે ક્લાસ વેરિયેબલને વાપરી શકે છે. main() મેથડમાં (જે RoomDemo સ્ટેટિક ક્લાસમાં વ્યાખ્યાયિત કરેલ છે) તે 'Room' ક્લાસની બહાર વ્યાખ્યાયિત કરેલી છે. આથી ક્લાસ વેરિયેબલ totWindowsને ક્લાસનું નામ વાપરીને Room.totWindowsથી ઍક્સેસ કરવું પડે.

ક્લાસ મેથડ એ જાતે ક્લાસથી વૈશ્વિક છે અને બીજા ક્લાસ કે ઑબ્જેક્ટને તે ઉપલબ્ધ છે. આથી, ક્લાસ મેથડ ક્લાસના ઇન્સ્ટન્સ અસ્તિત્વમાં છે કે નહીં તે ધ્યાનમાં લીધા વિના ગમે ત્યાં વાપરી શકાય છે.

હવે, પ્રશ્ન એ છે કે આપણે ક્લાસ મેથડ ક્યારે વાપરવી જોઈએ ? જે મેથડ કોઈ ચોક્કસ ઑબ્જેક્ટ ઉપર કાર્ય કરે અથવા ઑબ્જેક્ટને અસર કરે, તો તે ઇન્સ્ટન્સ મેથડ તરીકે વ્યાખ્યાયિત કરવી જોઈએ. એ મેથડ કે જે કેટલીક સામાન્ય યુટીલિટી પૂરી પાડે પણ ક્લાસના ઇન્સ્ટન્સ ઉપર સીધી અસર ન કરે, તેને ક્લાસ મેથડ તરીકે ઘોષિત કરવી વધુ સારી છે.

ઉદાહરણ તરીકે, આપેલી કોઈ સંખ્યા અવિભાજ્ય (prime) છે કે નહીં તે નક્કી કરતું વિધેય લો. આ વિધેયને ક્લાસ મેથડ તરીકે વ્યાખ્યાયિત કરવું જોઈએ. આકૃતિ 8.5માં કોડલિસ્ટિંગ અને પ્રોગ્રામનો આઉટપુટ દર્શાવ્યો છે.

primeClassMethod.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 primeClassMethod.java

```java
// Static method (class method)
// isPrime (int) returns true if given integer is prime
class Prime
{
    static boolean isPrime (int n)
    {   //n>1 is prime if it is not divisible by any number except 1 and itself
        int i, last;

        if (n <= 1) return false;
        if (n < 4) return true;
        //if (n%2==0) return false;   // divisible by 2, so not prime

        last = (int) Math.sqrt(n);
        i=3;
        do
        {
            if (n%i == 0) return false; // n is divisible by i
            i = i + 2;      // no need to divide by even numbers
        } while (i<last) ;  // end of do...while loop

        return true;
    } //end of method isPrime
} // end class primeFunc

class primeClassMethod
{
    public static void main (String[] s)
    {
        int i, n;

        System.out.println ("Prime numbers between 3 and 100:");
        for (n=3; n<100; n=n+2)
        {
            if (Prime.isPrime(n)) System.out.println(n);
        }
    } // end of main
} // end classs ClassMethodDemo
```

```
>javac primeClassMethod.java
>Exit code: 0
>java -cp . primeClassMethod
Prime numbers between 3 and 100:
3
5
7
11
13
17
19
23
25
29
31
35
37
41
43
47
49
53
59
61
67
71
73
79
83
89
97
>Exit code: 0
```

આકૃતિ 8.5 : સ્ટેટિક મેથડનો ઉપયોગ કરીને આપેલી પૂર્ણાંક સંખ્યા પ્રાઇમ છે કે નહીં તે નક્કી કરવું.

જાવાના રચયિતાઓએ આવી ક્લાસ મેથડ સાથેના પુષ્કળ પ્રમાણમાં સમાવિષ્ટ ક્લાસ પહેલેથી પૂરા પાડ્યા છે. પ્રકરણ 7માં આપણે Math ક્લાસનો કોઈ પણ ઑબ્જેક્ટ બનાવ્યા વિના sqrt() નામની સ્ટેટિક મેથડ વાપરેલી છે.

અહીં એ નોંધવુ જોઈએ કે આપણે અત્યાર સુધી જે main() મેથડ વ્યાખ્યાયિત કરેલી છે તે એક ક્લાસ મેથડ પણ છે. આ કારણને લીધે જ આપણે main() મેથડ વ્યાખ્યાયિત કરતા સમયે static ચાવીરૂપ શબ્દનો ઉપયોગ કરીએ છીએ.

અહીં જોવા મળે છે તે પ્રમાણે ક્લાસના સ્થિર અને ચલિત (static and non-static) ભાગો અલગ-અલગ હેતુઓ પૂરા પાડે છે. સૉર્સ કોડમાંની static વ્યાખ્યાઓ એ વસ્તુઓ જણાવે છે, જે ક્લાસનો જાતે એક ભાગ છે, જ્યારે non-static વ્યાખ્યાઓ એ વસ્તુઓ જણાવે છે, જે ક્લાસના દરેક ઑબ્જેક્ટ ઇન્સ્ટન્સના ભાગ બનશે.

યાદ રાખવા જેવા મુદ્દાઓ : (Points to Remember)

- ક્લાસ-વેરિયેબલ અને ક્લાસ-મૅથડને ક્લાસનેઇમ અથવા રેફરન્સ વેરિયેબલથી ઍક્સેસ કરી શકાય છે. તેની સુવાચ્યતા વધારવા માટે તેને ક્લાસનેઇમથી ઍક્સેસ કરવું વધારે સલાહભરેલું છે.
- ક્લાસ વેરિયેબલ અને ક્લાસ-મેથડને ઇન્સ્ટન્સ મેથડમાંથી પણ એક્સેસ કરી શકાય છે.
- ઇન્સ્ટન્સ વેરિયેબલ અને ઇન્સ્ટન્સ મૅથડને ક્લાસ મૅથડમાંથી ઍક્સેસ કરી ન શકાય કારણકે ક્લાસમૅથડ કોઈ ઑબ્જેક્ટની હોતી નથી.

ક્લાસમાં ઘોષિત કરેલા ચલ (વેરિયેબલ)નું વર્ગીકરણ (Classification of Variables Declared in a Class)

- **લોકલ વેરિયેબલ (Local Variables) :** જે વેરિયેબલ (ચલ) મેથડ કે બ્લૉકની અંદર વ્યાખ્યાયિત કરેલા હોય છે, તેને લોકલ વેરિયેબલ કહેવામાં આવે છે. મૅથડના નિયમાનુસારના પ્રાચલો પણ લોકલ વેરિયેબલ છે. જ્યારે બ્લૉકની શરૂઆત થાય છે, ત્યારે તેનું નિર્માણ થાય છે અને જ્યારે મૅથડ કે બ્લૉકનો અંત આવે છે, ત્યારે તેનો નાશ થાય છે. લોકલ વેરિયેબલની પૂર્વનિર્ધારિત કિંમતોથી શરૂઆત (initialized) થતી નથી.
- **ઇન્સ્ટન્સ વેરિયેબલ (Instance Variables) :** ઇન્સ્ટન્સ વેરિયેબલ એ કોઈ ક્લાસમાં પણ મેથડની બહાર વ્યાખ્યાયિત કરેલા ચલ (variable) છે. આ ચલ જ્યારે ઑબ્જેક્ટ નિર્મિત (instantiated) થાય છે, ત્યારે હીપ (heap) વિસ્તારમાંથી મેમરીની ફાળવણી થાય છે. ઇન્સ્ટન્સ વેરિયેબલને પૂર્વનિર્ધારિત કિંમતો આરંભમાં આપવામાં આવે છે (initialized).
- **ક્લાસ-વેરિયેબલ (Class Variables) :** ક્લાસ-વેરિયેબલને કોઈ ક્લાસની અંદર, મૅથડની બહાર અને static ચાવીરૂપ શબ્દ સાથે વ્યાખ્યાયિત કરવામાં આવે છે. આ ચલને ક્લાસ દીઠ ફક્ત એક વાર મેમરીની ફાળવણી થાય છે અને તેના બધા ઑબ્જેક્ટ વડે તે વાપરી શકાય છે. ક્લાસ-વેરિયેબલને પૂર્વનિર્ધારિત કિંમતો આરંભમાં આપવામાં આવે છે (initizalized).

પોલિમૉર્ફિઝમ (મૅથડ ઓવરલોડિંગ) - Polymorphism (Method Overloading)

આપણે ઑબ્જેક્ટ બનાવવાનાં પ્રથમ બે પગલાં Declaration અને Instantiation વિશે અભ્યાસ કર્યો. ત્રીજું પગલું પ્રારંભિક કિંમત આપવાનું (Initialization) છે. આ માટે કન્સ્ટ્રક્ટરના ઉપયોગની જરૂર પડે છે. આપણે કન્સ્ટ્રક્ટર વિશે શીખીએ તે પહેલાં આપણે જાવામાં પોલિમૉર્ફિઝમનું અમલીકરણ કઈ રીતે કરી શકાય તે બાબત જોઈએ.

પોલિમૉર્ફિઝમ શબ્દનો અર્થ 'અનેક સ્વરૂપ' કે 'બહુરૂપતા' થાય છે; એટલે કે એક જ નામ સાથે જુદી-જુદી મેથડ. જાવામાં એક જ નામ ધરાવતી પણ અલગ-અલગ સિગ્નેચર (signature) સાથેની વિવિધ મેથડ હોઈ શકે. આને 'મેથડ ઓવરલોડિંગ (method overloading)' કહેવામાં આવે છે. મેથડની સિગ્નેચર એ મેથડનું નામ, પરત કિંમતનો પ્રકાર (ઑબ્જેક્ટ કે બેઇઝ પ્રકાર) અને પ્રાચલોની યાદીનો સમૂહ છે.

ઉદાહરણ તરીકે, બે પૂર્ણાંક સંખ્યામાંથી મહત્તમ, ત્રણ પૂર્ણાંક સંખ્યામાંથી મહત્તમ, ત્રણ ડબલ પ્રિઝિશન (double precision) ધરાવતી અપૂર્ણાંક સંખ્યામાંથી મહત્તમ કિંમત શોધવા માટે કોઈ વ્યક્તિને એકસરખા પ્રકારનું પણ જુદી-જુદી સંખ્યાઓ ઉપર કાર્ય કરવું પડે છે. આ પરિસ્થિતિમાં, જાવા એક સમાન નામ પણ અલગ અલગ પ્રાચલો સાથે મેથડ બનાવવાની સગવડ પૂરી પાડે છે. મહત્તમ સંખ્યા શોધવામાં કોઈ ઑબ્જેક્ટ બનાવવાની જરૂર નથી, આથી તેને સ્ટેટિક મેથડ તરીકે વ્યાખ્યાયિત કરી શકાય છે. ઉદાહરણ તરીકે :

```
static int max(int x, int y) {...}
static int max(int x, int y, int z) {...}
static double max(double x, double y, double z) {...}
```

તમે આકૃતિ 8.6માં આપેલા કોડલિસ્ટિંગના ઉદાહરણને સમજ્યા પછી ઉપર જણાવેલી રચના પ્રમાણેની max મેથડ વ્યાખ્યાયિત કરવાનો અને તેનો વિનિયોગમાં ઉપયોગ કરવાનો પ્રયત્ન કરી શકો. અહીં તે જણાવેલા પ્રાચલો પ્રમાણે લીટી છાપવા માટે પોલિમૉર્ફિઝમનો ઉપયોગ કરે છે. કોઈ પણ પ્રાચલ વિના printline() 40 વાર '=' અક્ષર, printline(int n) n વાર '#' અક્ષર જ્યારે printline(int n, char ch) n વાર જણાવેલો અક્ષર ch છાપે છે.

polyDemo.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 polyDemo.java

```
// polymorphism: method printline
class PrintLine
{
    static void printline()
    {
        for (int i=0; i<40; i++)
            System.out.print('=');
        System.out.println();
    }

    static void printline(int n)
    {
        for (int i=0; i<n; i++)
            System.out.print('#');
        System.out.println();
    }

    static void printline(char ch, int n)
    {
        for (int i=0; i<n; i++)
            System.out.print(ch);
        System.out.println();
    }
} // end class PrintLine

public class polyDemo
{
    public static void main(String[] s)
    {
        PrintLine.printline();
        PrintLine.printline(30);
        PrintLine.printline('+',20);
    } // end main
} // end PolyDemo class
```

```
>javac polyDemo.java
>Exit code: 0
>java -cp . polyDemo
========================================
##############################
++++++++++++++++++++
>Exit code: 0
```

આકૃતિ 8.6 : મેથડ ઓવરલોડિંગ

કન્સ્ટ્રક્ટર્સ (Constructors)

કન્સ્ટ્રક્ટર એ એક વિશિષ્ટ પ્રકારની મેથડ છે, જે નવા ઑબ્જેક્ટ નિર્મિત કરતાં સમયે ઇન્વોક કરવામાં આવે છે. કન્સ્ટ્રક્ટર કોઈ પણ કાર્ય કરી શકે પણ તે મુખ્યત્વે પ્રારંભિક કિંમતો આપવા માટે (initializing actions) રચવામાં આવે છે. આપણે અત્યાર સુધી વપરાશકર્તા વ્યાખ્યાયિત કન્સ્ટ્રક્ટર વિના ક્લાસનો ઉપયોગ કર્યો છે. દરેક ક્લાસના **ડિફોલ્ટ કન્સ્ટ્રક્ટર** (default constructor) હોય છે; કોઈ વાર તેને ચલ વગરના કન્સ્ટ્રક્ટર તરીકે ઓળખવામાં આવે છે. ડિફોલ્ટ કન્સ્ટ્રક્ટર કોઈ ચલ (argument) લેતા નથી. તે નવા બનાવેલા ઑબ્જેક્ટના એટ્રિબ્યૂટને તેના ડેટાપ્રકાર પ્રમાણે પૂર્વનિર્ધારિત કિંમતો આપે છે (initializes).

કન્સ્ટ્રક્ટર અન્ય સામાન્ય મેથડથી નીચે જણાવ્યા પ્રમાણે અલગ પડે છે :

- કન્સ્ટ્રક્ટરનું નામ ક્લાસના નામ સમાન જ હોવું જોઈએ.
- કન્સ્ટ્રક્ટરને પરતપ્રકાર (return type) હોતો નથી.

- જ્યારે નવો પ્રક્રિયક વાપરીને ઑબ્જેક્ટ બનાવવામાં આવે છે, ત્યારે જ કન્સ્ટ્રક્ટર અંતર્ગત રીતે (implicitly) ઇન્વૉક કરવામાં આવે છે.
- કન્સ્ટ્રક્ટર પ્રોગ્રામમાં અન્ય કોઈ પણ જગ્યાએ સ્પષ્ટ રીતે (explicitly) ઇન્વૉક કરવામાં આવતો નથી.

અન્ય મેથડની જેમ કન્સ્ટ્રક્ટરને પણ ઓવરલોડ (overload) કરી શકાય છે. તે અલગ-અલગ પ્રાચલોની યાદી સાથે શક્ય છે. ઉદાહરણ તરીકે, આપણે નીચે જણાવ્યા પ્રમાણે 'Room' ઑબ્જેક્ટના ઍટ્રિબ્યૂટને અલગ-અલગ પ્રાચલોથી પ્રારંભિક કિંમતો આપવા (initialize કરવા) માટે આપણા પોતાના કન્સ્ટ્રક્ટર લખીએ :

Room(float l, float w, float h, byte n) : lengthને l, width ને w, heightને h અને nWindowsને n કિંમત આપવા માટે (initialize કરવા).

Room(float l, float w) : lengthને l, widthને w, heightને 10 અને nWindowsને 1 કિંમત આપવા માટે.

આકૃતિ 8.7માં આપેલો પ્રોગ્રામ કન્સ્ટ્રક્ટરનો ઉપયોગ દર્શાવે છે.

RoomConstructorDemoA.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 RoomConstructorDemoA.java

```
/* Using Constructors */
class Room
{
    float length, width, height;
    byte nWindows;
    static int totWindows;  // class variable

    Room (float l, float w, float h, byte n)
    {
        length = l; width = w; height = h;
        nWindows = n;  totWindows+=n;
    }
    Room (float l, float w)
    {
        length = l; width = w; height = 10;
        nWindows = 1;    totWindows++;
    }

    double area ( )  // area =  length * width
    {   return (length * width);    } // end area() method
    void display ( )
    {
        System.out.println ("\nLength: " + length + "\nWidth: " + width);
        System.out.println ("Height: " + height);
        System.out.println ( "Windows: " + nWindows);
    } // end display() method
} // end Room class

class RoomConstructorDemoA  /* Application using Room */
{
    public static void main (String args[])
    {
        Room r1 = new Room(16, 12.5f);
        Room r2 = new Room(20, 14, 12, (byte)2);
        r1.display();  r2.display();
        System.out.println ("\nTotal number of Windows: " + Room.totWindows);
    } // end main()
} // end RoomConstructorDemoA
```

```
>javac RoomConstructorDemoA.java
>Exit code: 0
>java -cp . RoomConstructorDemoA

Length: 16.0
Width: 12.5
Height: 10.0
Windows: 1

Length: 20.0
Width: 14.0
Height: 12.0
Windows: 2

Total number of Windows: 3
>Exit code: 0
```

આકૃતિ 8.7 : કન્સ્ટ્રક્ટરનો ઉપયોગ

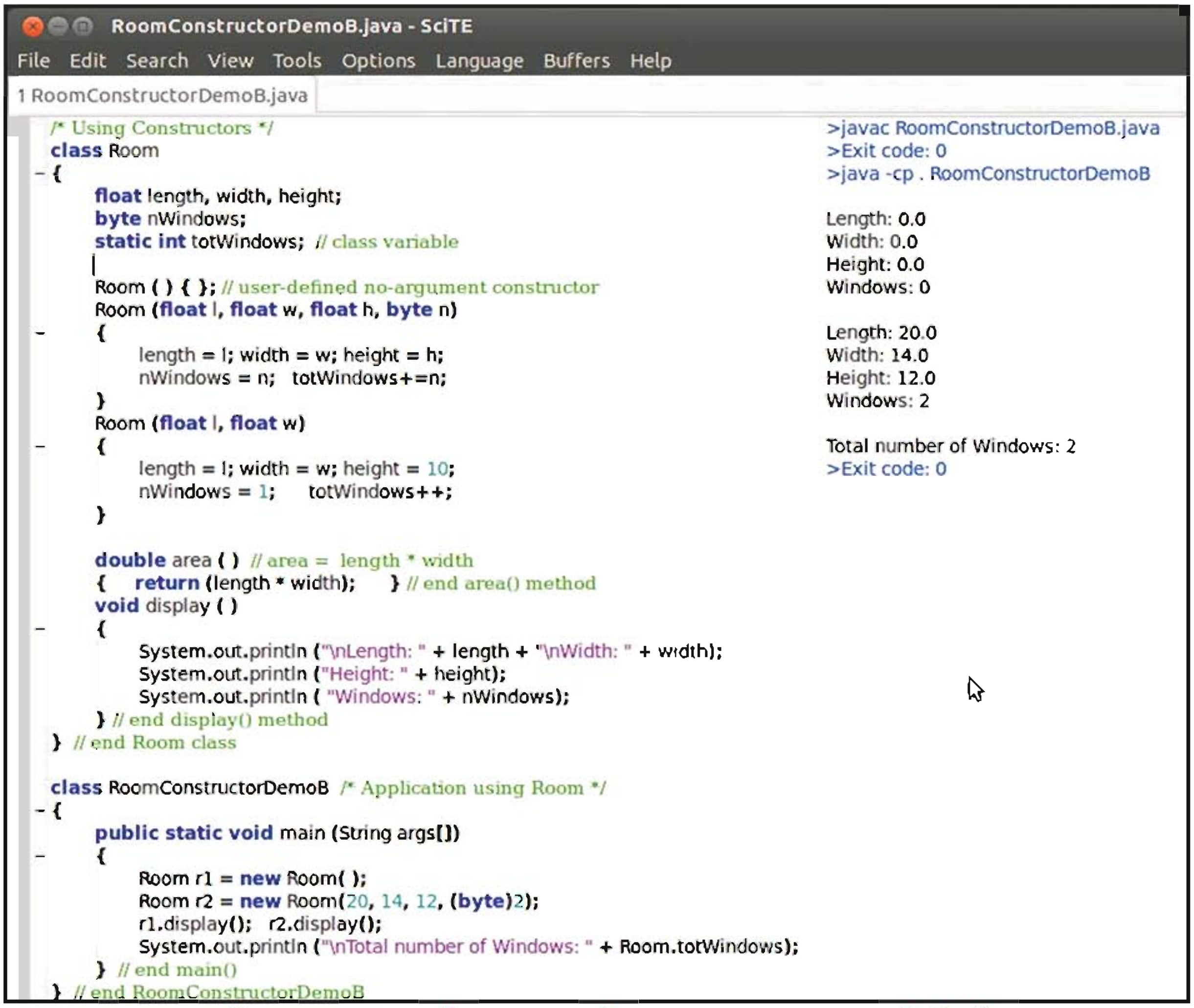
ક્લાસમાં વપરાશકર્તા વ્યાખ્યાયિત કન્સ્ટ્રક્ટરની ગેરહાજરીમાં ઑબ્જેક્ટ ચલ વગરના ડિફૉલ્ટ કન્સ્ટ્રક્ટરનો ઉપયોગ કરીને બનાવવામાં આવે છે. પૂર્વનિર્ધારિત કિંમત ઍટ્રિબ્યૂટને આપવામાં આવે છે.

ક્લાસમાં વપરાશકર્તા વ્યાખ્યાયિત કન્સ્ટ્રક્ટરની હાજરીમાં ડિફૉલ્ટ કન્સ્ટ્રક્ટર ઉપલબ્ધ હોતા નથી. ચલ વગરના કન્સ્ટ્રક્ટર સાથેનો ઑબ્જેક્ટ બનાવવાનો પ્રયત્ન કરવામાં આવે, તો કમ્પાઇલર એરર (error) પરત કરે છે. આ સમસ્યાનું નિરાકરણ લાવવા માટે આપણે નીચે જણાવ્યા પ્રમાણે વપરાશકર્તા દ્વારા વ્યાખ્યાયિત ચલ વગરનો કન્સ્ટ્રક્ટર પૂરો પાડવો જોઈએ :

<classname> ( ) { };

આકૃતિ 8.7માં આપેલા કોડ લિસ્ટિંગ પ્રમાણે નીચે જણાવ્યા પ્રમાણે main મેથડમાં Room ઑબ્જેક્ટ બનાવવાનો પ્રયત્ન કરી જુઓ અને કમ્પાઇલેશન દરમિયાન દર્શાવાતી errorનું નિરીક્ષણ કરો : Room r3 = new Room();

આ error દૂર કરવા માટે વપરાશકર્તા વ્યાખ્યાયિત ચલ વગરનો કન્સ્ટ્રક્ટર 'Room () {};' ઉમેરો અને પછી તે પ્રોગ્રામનો અમલ કરો. હવે આકૃતિ 8.8માં આપેલા પ્રમાણે પ્રોગ્રામનો સફળ અમલ જુઓ.



આકૃતિ 8.8 : વપરાશકર્તા દ્વારા વ્યાખ્યાયિત ચલ વગરના કન્સ્ટ્રક્ટરનો ઉપયોગ

એક્સેસ કન્ટ્રોલ માટેનાં વિઝિબિલિટી મોડિફાયર (Visibility Modifiers for Access Control)

એક્સેસ કન્ટ્રોલ એ દૃશ્યતાની નિયંત્રણ બાબત છે. આથી, એક્સેસ મોડિફાયર **વિઝિબિલિટી મોડિફાયર** તરીકે પણ ઓળખાય છે. જો મેથડ અથવા વેરિયેબલ બીજા ક્લાસમાં દૃશ્યમાન (visible) હોય, તો જ અન્ય ક્લાસમાં તેનો ઉલ્લેખ કરી શકાય છે. આ પ્રકારના નિર્દેશથી મેથડ અથવા વેરિયેબલને સુરક્ષિત રાખવા માટે આપણે ચાર સ્તરનાં વિઝિબિલિટીનો ઉપયોગ જરૂરી સુરક્ષા પૂરી પાડવા માટે કરીએ છીએ.

સુરક્ષા કે પ્રોટેક્શન (protection)નાં ચાર p's પબ્લિક (public), પૅકેજ (package) (પૂર્વનિર્ધારિત સુરક્ષા), પ્રોટેક્ટેડ (protected) અને પ્રાઇવેટ (private) છે. પબ્લિક, પ્રોટેક્ટેડ અને પ્રાઇવેટ એક્સેસ મોડિફાયર ચલ કે મેથડના પ્રકાર પહેલાં વપરાય છે. જ્યારે કોઈ પણ મોડિફાયર વાપરેલો ન હોય, ત્યારે તે પૂર્વનિર્ધારિત પ્રકારની વિઝિબિલિટી પૅકેજ છે, જેનો ક્લાસમાં સમાવેશ થાય છે.

પૅકેજ (package) વિવિધ ક્લાસને વ્યવસ્થિત રૂપ આપવા માટે વપરાય છે. આ માટે સૉર્સફાઇલમાં પૅકેજ વિધાન સૌથી પ્રથમ કોમેન્ટ ન હોય અને બ્લૅન્ક લીટી ન હોય તે રીતે ઉમેરવામાં આવે છે. જ્યારે ફાઇલમાં પૅકેજવિધાન ન હોય ત્યારે ફાઇલમાં વ્યાખ્યાયિત ક્લાસને ડિફૉલ્ટ પૅકેજમાં રાખવામાં આવે છે. પૅકેજવિધાનની વાક્યરચના નીચે પ્રમાણે છે :
package<packageName>;

આ પુસ્તકમાં આપણે ડિફોલ્ટ પૅકેજનો ઉપયોગ કરીશું. આપણા બધા પ્રોગ્રામમાં અત્યાર સુધી આપણે ઍક્સેસ મોડિફાયરનો ઉપયોગ કર્યો છે. કોષ્ટક 8.1માં ઍક્સેસ મોડિફાયર અને તેની વિઝિબિલિટી દર્શાવી છે.

		Type		
Access Modifier	public	(default: package)	protected	private
Visibility	widest	→ → →	→ →	narrowest

કોષ્ટક 8.1 : ઍક્સેસ મોડિફાયરના પ્રકારો અને તેની વિઝિબિલિટી

હવે આપણે ડિફોલ્ટ અને પ્રાઈવેટ મોડિફાયરનાં ઉદાહરણો જોઈશું.

પબ્લિક (Public)

કોઈ પણ મેથડ કે ચલ જે ક્લાસમાં વ્યાખ્યાયિત કરેલા હોય તેમાં જ તે ઉપલબ્ધ (વિઝિબલ) હોય છે. જો આપણે એ ક્લાસની બહાર બધા ક્લાસમાં ઉપલબ્ધ બનાવવા ઈચ્છતા હોઈએ તો તે મેથડ અથવા ચલને પબ્લિક ઍક્સેસ માટે ઘોષિત કરો. આ સૌથી વધારે શક્ય ઍક્સેસ (access) છે. તે અન્ય પૅકેજમાં વ્યાખ્યાયિત ક્લાસને પણ વિઝિબિલિટી પૂરી પાડે છે. પબ્લિક ઍક્સેસ આપવા માટે ચલ કે મેથડના પ્રકાર પહેલાં public ઍક્સેસ મોડિફાયર વાપરો. ઉદાહરણ તરીકે,

```
public float length
public double area ( )
```

અહીં એ નોંધ કરશો કે public ચલ અને મેથડ બધી જ જગ્યાએ વિઝિબલ હોય છે અને આથી તે અન્ય સૉર્સફાઈલ અને પૅકેજમાંથી પણ ઍક્સેસ કરી શકાય છે.

આપણે દરેક વ્યક્તિને ઉપલબ્ધ બનાવવા માટે main( ) મેથડ સાથે public ચાવીરૂપ શબ્દનો ઉપયોગ કર્યો છે.

```
public static void main(String[] args) { ... }
```

પૅકેજ (કોઈ પણ મોડિફાયર વિના) Package (Without any Modifier)

આ બીજા સ્તરનો ઍક્સેસ (access) છે જેને કોઈ નિશ્ચિત નામ નથી. તે ઘોષિત કરવાના વિધાનમાં કોઈ પણ પ્રકારના ઍક્સેસ મોડિફાયરની ગેરહાજરી વડે જણાવવામાં આવે છે. આ ડિફોલ્ટ સ્તરનું પ્રોટેક્શન (રક્ષણ) છે. તેનો વિસ્તાર પબ્લિક ચલ કરતાં ઓછો છે. પૅકેજમાં બધી જગ્યાએથી ચલ કે મેથડને ઍક્સેસ કરી શકાય છે કે જ્યાં ક્લાસનો સમાવેશ થયેલો છે, પણ તે પૅકેજની બહાર નહીં. અહીં એ નોંધ કરશો કે જે સૉર્સફાઈલ package વિધાન વિનાની હશે, તે package સાથેની પૂર્વનિર્ધારિત ગણવામાં આવશે. આથી, અત્યાર સુધીના આપણા પ્રોગ્રામમાં તે public બરાબર જ છે.

આકૃતિ 8.9માં દર્શાવેલા પ્રોગ્રામનો સંદર્ભ લો. અહીં 'Rectangle' ક્લાસનાં બે ઍટ્રિબ્યૂટ છે : length અને width. તેને અનેક મેથડ પણ છે. આપણે કોઈ પણ મોડિફાયર ચાવીરૂપ શબ્દ વાપર્યો નથી, આથી તેનું પૂર્વનિર્ધારિત રીતે package પ્રોટેક્શન રહેશે. આ કારણને લીધે આ જ સૉર્સફાઈલમાં (પૂર્વનિર્ધારિત package) વ્યાખ્યાયિત અન્ય ક્લાસ 'Rectangle Demo'માં તે સીધેસીધા મેળવી શકાય છે (accessible).

RectangleDemo.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 RectangleDemo.java

```java
class Rectangle
{
    double length, width;

    void setAttributes(double x, double y)
    {
        length = x;  width = y;
    }

    double area ()
    {
        return length * width;
    }

    void display()
    {
        System.out.println ("Rectangle with length = " + length
                            + " width = " + width );
    }
}// end class Rectangle

class RectangleDemo
{
    public static void main (String[] s)
    {
        Rectangle rect1;
        rect1 = new Rectangle();
        Rectangle rect2 = new Rectangle();

        rect1.setAttributes (10.5, 20);
        rect1.display();
        System.out.println ("Area of rectangle is " + rect1.area());
        rect2.setAttributes(10,15);
        System.out.println ("Area of rectangle with length " +
                            rect2.length + ", width = " + rect2.width
                            + " is " + rect2.area());
    } //end main()
} // end class RectangleDemo
```

```
>javac RectangleDemo.java
>Exit code: 0
>java -cp . RectangleDemo
Rectangle with length = 10.5 width = 20.0
Area of rectangle is 210.0
Area of rectangle with length 10.0, width = 15.0 is 150.0
>Exit code: 0
```

આકૃતિ 8.9 : પૂર્વનિર્ધારિત વિઝિબિલિટી મોડિફાયર, પૅકેજમાં સર્વત્ર ઉપલબ્ધ

પ્રોટેક્ટેડ (Protected)

આ સ્તરના પ્રોટેક્શનનો ઉપયોગ ફક્ત સબક્લાસને એક્સેસ કરવા માટે અથવા 'friend' તરીકે ઘોષિત કરેલી મેથડ સાથે સહિયારા ઉપયોગ માટે થાય છે. આથી, અગાઉ જણાવેલા બંને સ્તર કરતાં વિઝિબિલિટી ઓછી છે, પણ ચોથા સ્તરની 'private'માં પૂરી પાડવામાં આવેલી પૂર્ણ ગુપ્તતા (privacy) કરતાં વધુ છે.

પ્રોટેક્ટેડ પ્રોટેક્શનનો ઉપયોગ આપણે જ્યારે ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગનો ખ્યાલ ઇનહેરિટન્સ (inheritance) વાપરીશું, ત્યારે વધારે સુસંગત જણાશે.

પ્રાઇવેટ (Private)

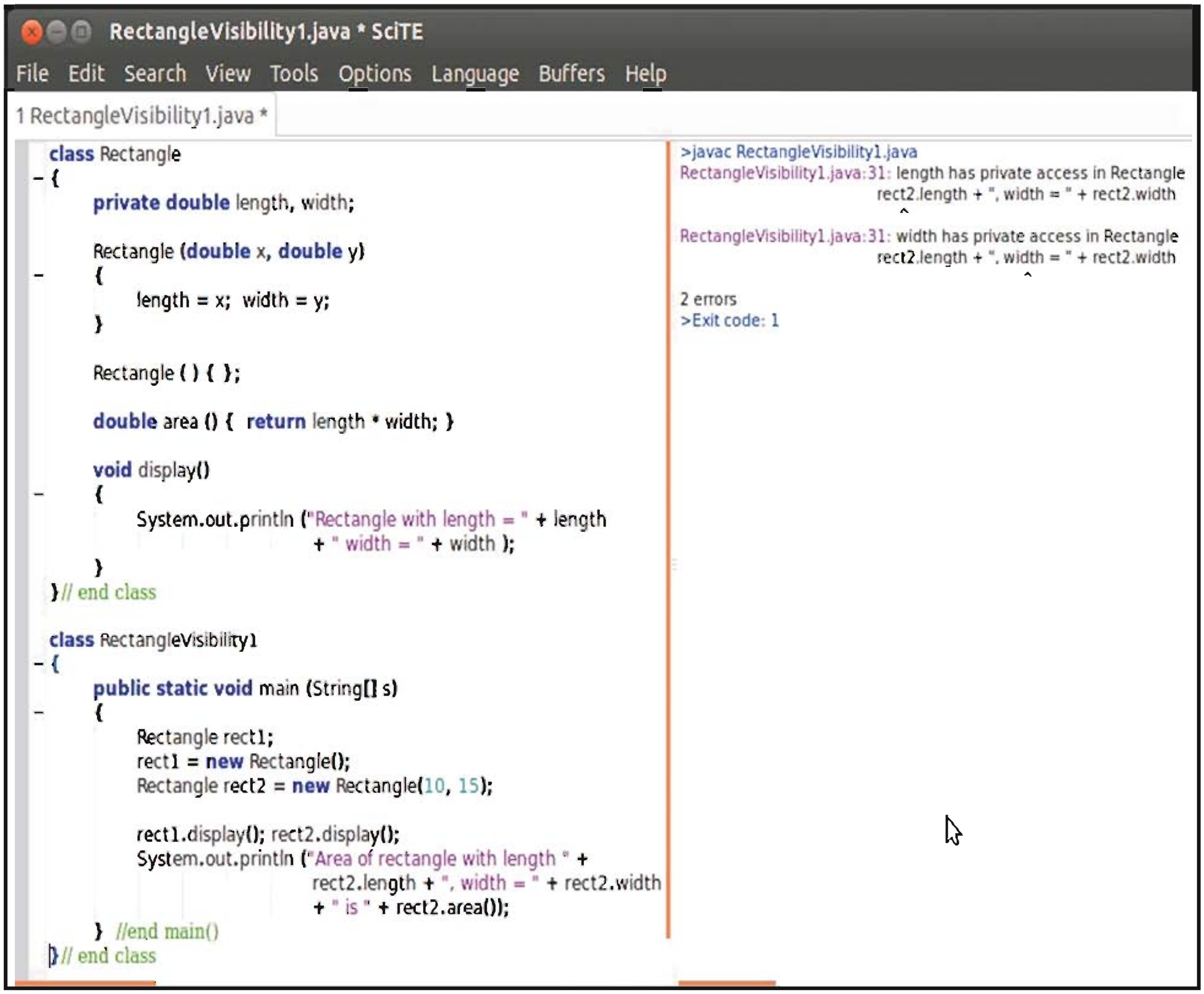
"private" કક્ષાનું પ્રોટેક્શન વાપરવાથી સૌથી ઉચ્ચ કક્ષાનું પ્રોટેક્શન મેળવી શકાય છે. તે સૌથી ઓછી દૃશ્યતા પૂરી પાડે છે. પ્રાઇવેટ મેથડ અને ચલને ક્લાસની અંદર જ વ્યાખ્યાયિત મેથડ દ્વારા સીધેસીધા ઍક્સેસ કરી શકાય છે. તે અન્ય કોઈ પણ ક્લાસ દ્વારા જોઈ શકાતા નથી.

આ બહુ જ પ્રતિબંધક જણાય, પણ હકીકતમાં તે સામાન્ય રીતે વપરાતા પ્રોટેક્શનનું સ્તર છે. તે ડેટા એનકૅપ્સ્યુલેશન (data encapsulation) પૂરું પાડે છે; દુનિયાની નજરમાંથી ડેટા છુપાવે છે અને તેની ખોટી રીતની ગણતરીને મર્યાદામાં રાખે છે. જે ઘટક સીધા કોઈ પણ સાથે તેના સબક્લાસના સમાવેશ સાથે સહિયારું ન હોય તે private છે.

ડેટા ઇનકૅપ્સ્યુલેશન પૂરો પાડવાનો સૌથી શ્રેષ્ઠ માર્ગ શક્ય એટલો વધારેમાં વધારે ડેટા private કરવાનો છે. તે તેની રચનાને તેના અમલીકરણની ક્રિયાથી અલગ કરે છે, અને તેનું કાર્ય કરવા માટે કોઈ ક્લાસને અન્ય ક્લાસની માહિતી જાણવાની જરૂરિયાતને ઓછામાં ઓછી કરે છે.

હવે, આપણે આકૃતિ 8.9માં આપેલા કોડલિસ્ટિંગમાં ફેરફાર કરીએ અને length તથા widthને private ઘોષિત કરીએ. private સભ્યો તે જ ક્લાસમાં સીધેસીધા ઉપલબ્ધ હોવાથી જ્યારે આપણે area() અને display() મેથડમાં તેનો ઉપયોગ કરીશું, ત્યારે તે કોઈ પણ પ્રકારની error નહીં બતાવે. જ્યારે તે RectangleDemo ક્લાસની main() મેથડમાં ઍક્સેસ કરવામાં આવશે, ત્યારે તે error દર્શાવશે.

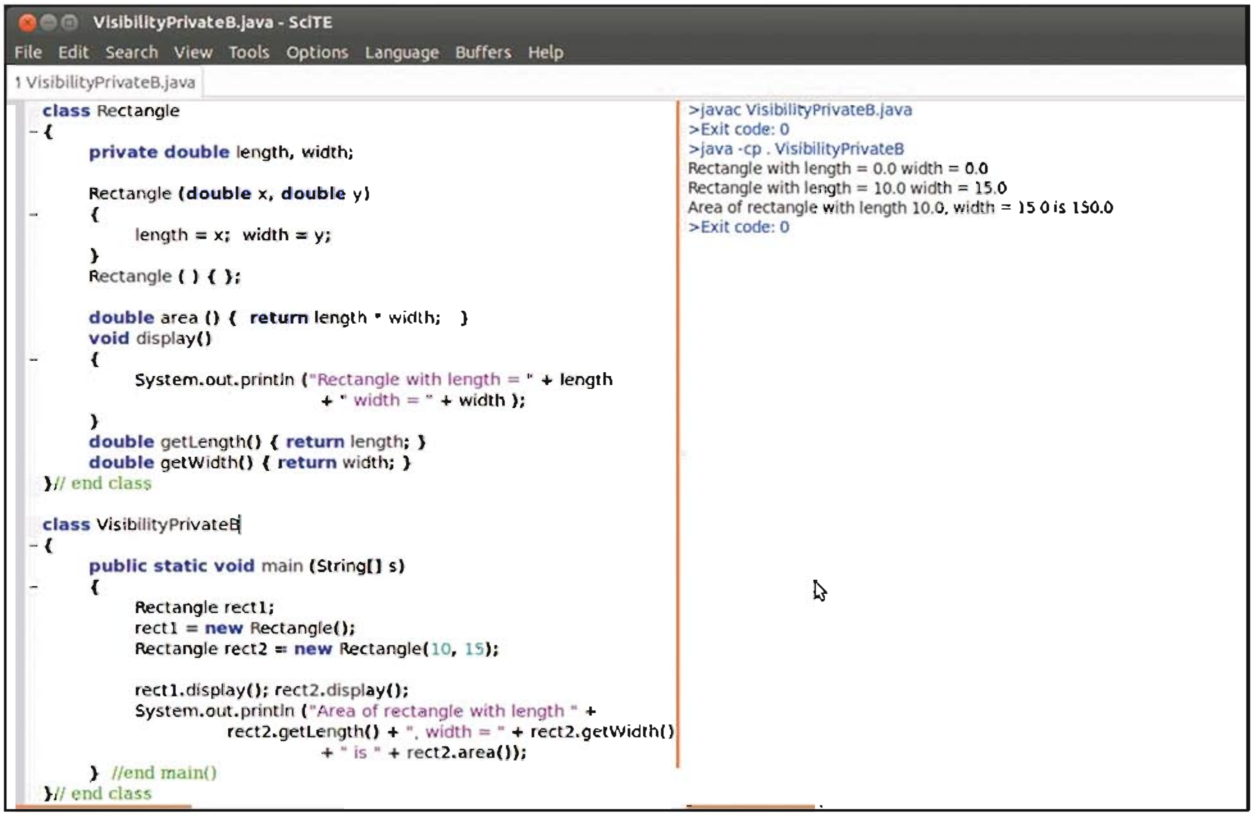
સુધારેલો કોડ આકૃતિ 8.10માં દર્શાવ્યો છે. અહીં, આપણે private ઇન્સ્ટન્સ વેરિયેબલ ઉપરાંત કન્સ્ટ્રક્ટર વાપરેલા છે. નિર્ગમ વિભાગમાં error દર્શાવે છે તેની નોંધ કરો જે જણાવે છે કે 'length has private access in Rectangle'. આનો અર્થ એ થાય કે તે 'RectangleVisibility1' ક્લાસમાં ઍક્સેસ કરી ન શકાય.



આકૃતિ 8.10 : બીજા ક્લાસમાંથી private ઇન્સ્ટન્સ વેરિયેબલ ઍક્સેસ કરવાથી error મળે છે

હવે સમસ્યા એ છે કે બીજા ક્લાસમાંથી private ચલ કઈ રીતે મેળવી શકાય ? અન્ય ક્લાસની મેથડ વડે ઍક્સેસિબલ (accessible) છે, તેના દ્વારા તે પરોક્ષ રીતે ઉપલબ્ધ બનાવી શકાય છે. આકૃતિ 8.11માં આપેલું કોડ લિસ્ટિંગ જુઓ.

અહીં આપણે બે મેથડ getLength() અને getWidth() ઉમેરેલી છે. અહીં કોઈ પણ મોડિફાયર વાપરેલો ન હોવાથી તેની package વિઝિબિલિટી છે અને આથી તે અન્ય ક્લાસ 'Visibility PrivateB' માં ઉપલબ્ધ છે. આ મેથડ દ્વારા આપણે private ડેટાફિલ્ડ (ઇન્સ્ટન્સ વેરિયેબલ) 'length' અને 'width'ની કિંમત મેળવી શકીએ છીએ. 'Visibility PrivateB' ક્લાસની main() મેથડના છેલ્લા નિર્ગમ વિધાનમાં getLength() અને getWidth() મેથડકૉલનો ઉપયોગ જુઓ.



VisibilityPrivateB.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 VisibilityPrivateB.java

```
class Rectangle
{
	private double length, width;

	Rectangle (double x, double y)
	{
		length = x;  width = y;
	}
	Rectangle ( ) { };

	double area () {  return length * width;   }
	void display()
	{
		System.out.println ("Rectangle with length = " + length
				+ " width = " + width );
	}
	double getLength() { return length; }
	double getWidth() { return width; }
}// end class

class VisibilityPrivateB
{
	public static void main (String[] s)
	{
		Rectangle rect1;
		rect1 = new Rectangle();
		Rectangle rect2 = new Rectangle(10, 15);

		rect1.display(); rect2.display();
		System.out.println ("Area of rectangle with length " +
			rect2.getLength() + ", width = " + rect2.getWidth()
				+ " is " + rect2.area());
	} //end main()
}// end class
```

```
>javac VisibilityPrivateB.java
>Exit code: 0
>java -cp . VisibilityPrivateB
Rectangle with length = 0.0 width = 0.0
Rectangle with length = 10.0 width = 15.0
Area of rectangle with length 10.0, width = 15.0 is 150.0
>Exit code: 0
```

આકૃતિ 8.11 : public અને package મેથડ દ્વારા private વેરિયેબલને ઍક્સેસ કરવા

ઍક્સેસર અને મ્યુટેટર મેથડ (Accessor and Mutator Methods)

જ્યારે આપણે ડેટાને private ઘોષિત કરીને તેનો ઍક્સેસ મર્યાદિત કરીએ છીએ, ત્યારે આપણો હેતુ અન્ય ક્લાસની મેથડ વડે તે સીધેસીધા ઍક્સેસ કરવાથી અથવા તેમાં ફેરફાર કરવાથી તેનું રક્ષણ કરવાનો છે. જો આપણે આ ડેટાનો ઉપયોગ અન્ય (વ્યક્તિઓ) દ્વારા માન્ય રાખતા હોઈએ તો આપણે ઍક્સેસર (accessor) મેથડ લખીએ. જો આપણે આ ડેટામાં ફેરફાર અન્ય (વ્યક્તિઓ) દ્વારા માન્ય રાખતા હોઈએ તો આપણે મ્યુટેટર (mutator) મેથડ લખીએ.

માન્ય પ્રણાલિકા પ્રમાણે ઍક્સેસર અને મ્યુટેટર મેથડના નામ ચલના નામનો પ્રથમ અક્ષર કેપિટલ બનાવીને તેના પૂર્વગ તરીકે અનુક્રમે get અને set લખવાથી બને છે. આ પ્રણાલીને કારણે ઍક્સેસર મેથડ "getter" તરીકે અને મ્યુટેટર મેથડ "setter" તરીકે પણ ઓળખાય છે.

આકૃતિ 8.11માં દર્શાવેલા કોડલિસ્ટિંગનું નિરીક્ષણ કરો, જેમાં આપણે "getter" મેથડ getLength() અને getWidth() મેથડનો ઉપયોગ કર્યો છે. જો આપણે ચલ 'length'નો પ્રકાર બદલીશું, તો તે અન્ય વપરાશકર્તાથી છૂપું રહેશે. તે ઍક્સેસર મેથડ 'getLength()'ના અમલીકરણમાં જ ફક્ત અસર કરશે.

જો આપણે બીજી મેથડને ફક્ત ડેટાકિંમત વાંચવાની જ પરવાનગી આપવા ઇચ્છતા હોય, તો આપણે "getter" મેથડ વાપરવી જોઈએ.

જો આપણે અન્ય મેથડને ડેટાની કિંમતમાં ફેરફાર કરવાની પરવાનગી આપવા ઇચ્છતા હોય, તો આપણે "setter" મેથડ વાપરવી જોઈએ. ઉદાહરણ તરીકે, setLength() મેથડને 'length' ઍટ્રિબ્યૂટની કિંમત નીચે જણાવ્યા પ્રમાણે ચલ પસાર કરીને સેટ કરવા માટે વ્યાખ્યાયિત કરી શકાય છે :

```
void setLength(float l) { length = l; }
```

ઍક્સેસર અને મ્યુટેટર મેથડનો ઉપયોગ અન્ય ક્લાસના વપરાશકર્તા દ્વારા ચલને સીધેસીધા ઍક્સેસ કરવા અને તેમાં ફેરફાર કરવાથી અટકાવે છે. આ બાબતની આદત કેળવવી જરા મુશ્કેલ જણાય છે, કારણકે આપણી જરૂરિયાત પ્રમાણે દરેક ઇન્સ્ટન્સ વેરિયેબલ માટે આપણે get અને set મેથડ લખવી પડે. આ નાની અગવડ આપણને સરળતાથી કોડને ફરી વાપરવાની અને તેની જાળવણીની ભેટ આપશે.

મેથડમાં પ્રાચલ તરીકે ઑબ્જેક્ટ પસાર કરવો (Passing Object as a Parameter in a Method)

ચલના પ્રાથમિક ડેટાપ્રકાર અને ઉપલબ્ધ સમાવિષ્ટ (built-in) ડેટાપ્રકારોની જેમ જ ઑબ્જેક્ટ પણ મેથડમાં પ્રાચલ તરીકે પસાર કરી શકાય છે.

ઉદાહરણ તરીકે, આપણે rectangle ઑબ્જેક્ટ ઇન્વૉક કરતી મેથડનું ક્ષેત્રફળ અન્ય લંબચોરસ કરતાં વધુ છે કે નહીં તે નક્કી કરવા ઇચ્છીએ છીએ. આ માટે આપણે એક મેથડ લખીએ, જેમાં અન્ય rectangle ઑબ્જેક્ટ એક ચલ કે પ્રાચલ તરીકે પસાર કરીએ. ચાલો, આપણે આકૃતિ 8.12માં દર્શાવ્યા પ્રમાણે (Rectangle) ક્લાસમાં 'isLarge' નામની મેથડ ઉમેરીએ અને તેને main() મેથડમાં વાપરીએ.

```
boolean isLarge (Rectangle rect)
{ if (area() > rect.area() ) return true; else return false; }
```

આકૃતિ 8.12માં દર્શાવ્યા પ્રમાણે main() મેથડમાં if વિધાનની અંદરનો મેથડ કૉલ જુઓ : rect1.isLarge(rect2). અહીં, 'rect1' એ કૉલિંગ અથવા ઇન્વૉકિંગ ઑબ્જેક્ટ છે જ્યારે 'rect2' એ પ્રાચલ તરીકે પસાર કરેલો ઑબ્જેક્ટ છે. isLarge() મેથડમાં area() એ 'rect1' ઑબ્જેક્ટનું ક્ષેત્રફળ છે અને rect.area() ચલ તરીકે પસાર કરેલા ઑબ્જેક્ટના ક્ષેત્રફળનો ઉલ્લેખ કરે છે.

ObjectParameter.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 ObjectParameter.java

```
class Rectangle
{
    private double length, width;

    Rectangle (double x, double y)
        { length = x; width = y; }
    Rectangle ( ) { };

    double area () { return length * width;   }
    void display()
    {
        System.out.println ("Rectangle with length = " + length
                            + " width = " + width );
    }
    double getLength() { return length; }
    double getWidth() { return width; }
    boolean isLarge (Rectangle rect)
    {
        if (area() > rect.area()) return true;
            else return false;
    }
}// end class

class ObjectParameter
{
    public static void main (String[] s)
    {
        Rectangle rect1 = new Rectangle(8, 20);
        Rectangle rect2 = new Rectangle(10, 15);

        rect1.display();
        System.out.println ("Area of rectangle 1 is "+ rect1.area() + "\n");
        rect2.display();
        System.out.println ("Area of rectangle 2 is "+ rect2.area() + "\n");
        if (rect1.isLarge(rect2))
            System.out.println ("Area of rectangle 1 is larger than " +
                                "Area of rectangle 2");
    } //end main()
}// end class ObjectParameter
```

```
>javac ObjectParameter.java
>Exit code: 0
>java -cp . ObjectParameter
Rectangle with length = 8.0 width = 20 0
Area of rectangle 1 is 160.0

Rectangle with length = 10.0 width = 15.0
Area of rectangle 2 is 150.0

Area of rectangle 1 is larger than Area of rectangle 2
>Exit code: 0
```

આકૃતિ 8.12 : પ્રાચલ તરીકે પસાર કરેલ ઑબ્જેક્ટ

અહીં એ યાદ રાખો કે પ્રાથમિક (primitive) પ્રકારના પ્રાચલો કિંમત તરીકે પસાર કરેલા છે. વાસ્તવિક (actual) પ્રાચલોની કિંમત નિયમાનુસાર (formal) પ્રાચલોમાં નકલ કરી પછી વિધેયનો અમલ થાય છે. નિયમાનુસાર પ્રાચલોમાં કરેલો ફેરફાર વાસ્તવિક પ્રાચલોને અસર કરતો નથી.

અહીં એ નોંધ કરશો કે ઑબ્જેક્ટના પ્રાચલો રેફરન્સ (Reference) થી પસાર કરેલા છે. આથી, મેથડની અંદરના ઑબ્જેક્ટમાં જે કંઈ ફેરફાર કરવામાં આવે છે, તેથી મૂળ ઑબ્જેક્ટમાં પણ અસર થાય છે. અહીં, વાસ્તવિક પ્રાચલના સ્થાનાંક (અને કિંમત નહીં)ની નકલ નિયમાનુસાર પ્રાચલમાં કરવામાં આવે છે.

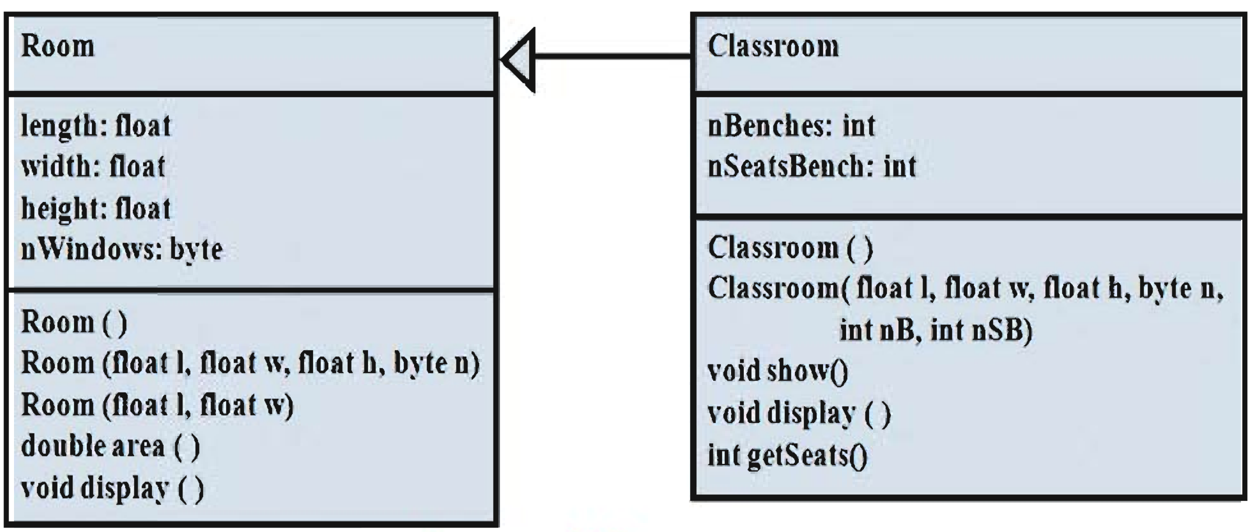
ઈનહેરિટન્સ (Inheritance)

ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ ભાષા ઈનહેરિટન્સના ઉપયોગ વડે ફરી વાપરવાની લાક્ષણિક્તા (reusability feature) પૂરી પાડે છે. ઈનહેરિટન્સ આપણને હયાત ક્લાસને વિસ્તૃત કરીને વધારાના સામર્થ્ય સાથેનો નવો ક્લાસ બનાવવાની સગવડ આપે છે.

ઈનહેરિટન્સ બે ક્લાસ 'is-a' પ્રકારનો સંબંધ ધરાવતું મૉડેલ છે. ઉદાહરણ તરીકે classroom એ એક room છે અને student એ એક person છે. અહીં, room અને personને પેરન્ટક્લાસ (parentclass) કહેવામાં આવે છે અને classroom તથા studentને ચાઈલ્ડક્લાસ (child class) કહેવામાં આવે છે. પેરન્ટક્લાસને સુપર ક્લાસ (superclass) અથવા બેઈઝ ક્લાસ (base class) પણ કહેવામાં આવે છે. એ જ પ્રકારે ચાઈલ્ડક્લાસને સબક્લાસ (subclass) અથવા ઍક્સટેન્ડેડ ક્લાસ (extended class) પણ કહેવામાં આવે છે.

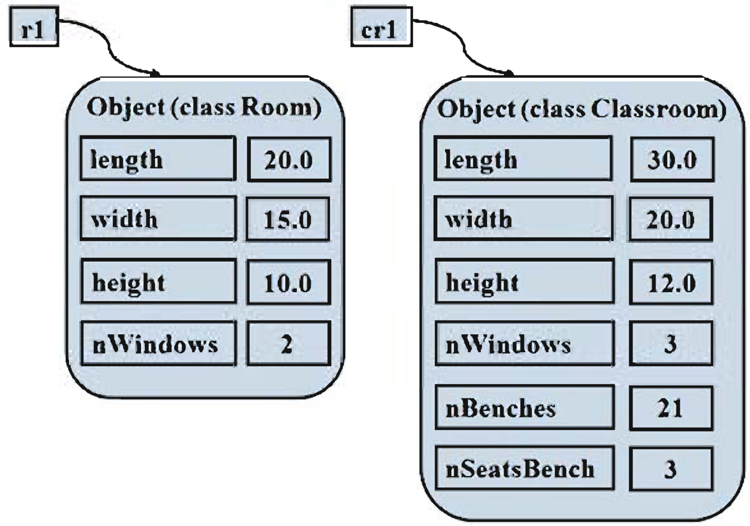
જ્યારે બે ક્લાસ વચ્ચે 'is-a' પ્રકારનો સંબંધ હોય છે, ત્યારે આપણે ઈનહેરિટન્સનો ઉપયોગ કરીએ છીએ. સામાન્ય ગુણધર્મો સુપર ક્લાસમાં રાખવામાં આવે છે. સબક્લાસ સુપર ક્લાસમાંથી બધા ઈન્સ્ટન્સ વેરિયેબલ અને મેથડ વારસામાં મેળવે છે અને તેના પોતાના વધારાના વેરિયેબલ અને મેથડ હોઈ શકે. અહીં નોંધ કરશો કે સબક્લાસમાં કન્સ્ટ્રક્ટર વારસામાં (inherit) મળતા નથી. ઉદાહરણ તરીકે, roomની જેમ classroomને પણ length, width, height અને number of windows ચલ હોય છે. આ ઉપરાંત તેમાં benchesની સંખ્યા અને દરેક benchની વિદ્યાર્થીઓ સમાવવાની ક્ષમતા પણ હોય છે. આ જ રીતે, સબક્લાસ સુપર ક્લાસની બધી જ મેથડ વારસામાં મેળવે છે અને તેની પોતાની વધારાની મેથડ પણ હોઈ શકે. અહીં, સબક્લાસ classroomની વધારાની મેથડ show(), display(), getSeats() તેના કન્સ્ટ્રક્ટરની મેથડ છે.

આકૃતિ 8.13માં ક્લાસ ડાયાગ્રામ દર્શાવ્યો છે, જેમાં 'classroom' નામનો ક્લાસ છે, જે તેના 'Room' નામના પેરન્ટક્લાસમાંથી આવેલો છે (derived). તીર સબક્લાસથી સુપરક્લાસ તરફ નિર્દેશ કરે છે તે જુઓ. સબક્લાસમાં ફક્ત વધારાના ઍટ્રિબ્યૂટ અને મેથડ જ દર્શાવવાની હોય છે. અહીં નોંધ કરશો કે સબક્લાસ એ સુપર ક્લાસનો સબસેટ નથી. હકીકતમાં, સબક્લાસ હંમેશાં તેના સુપર ક્લાસ કરતાં વધારે માહિતી અને મેથડ ધરાવે છે.



આકૃતિ 8.13 : ઈનહેરિટન્સ ક્લાસ ડાયાગ્રામ

જ્યારે સબક્લાસનો ઑબ્જેક્ટ ઈન્સ્ટન્સિયેટ (instantiate) થાય છે, ત્યારે ઈનહેરિટ થયેલાં સાથે તેના બધા ઍટ્રિબ્યૂટને મેમરી ફાળવવામાં આવે છે. આકૃતિ 8.14માં સુપર ક્લાસ Room અને સબક્લાસ classroomના ઈન્સ્ટન્સ દર્શાવ્યા છે.



આકૃતિ 8.14 : સુપર ક્લાસ અને સબક્લાસના ઈન્સ્ટન્સ

જાવામાં સબક્લાસ બનાવવા માટે ક્લાસની વ્યાખ્યામાં ચાવીરૂપ શબ્દ 'extends' વાપરવામાં આવે છે. હવે આપણે હયાત ક્લાસ 'Room'નો ઉપયોગ કરીને સબક્લાસ 'Classroom' બનાવીએ. હયાત ક્લાસ 'Room' કોડલિસ્ટિંગ 8.2માં દર્શાવ્યો છે.

```
class Room
{
        float length, width, height;
        byte nWindows;
        static int totWindows;  // class variable

        Room ( ) { }; // user-defined no-argument constructor
        Room (float l, float w, float h, byte n)
        {
                length = l; width = w; height = h;
                nWindows = n;    totWindows+=n;
        }
        Room (float l, float w)
        {
                length = l; width = w; height = 10;
                nWindows = 1;       totWindows++;
        }

    double area ( )  // area =  length * width
        {      return (length * width);        } // end area() method

          void display ( )
        {
                System.out.println ("\nLength: " + length + "\nWidth: " + width);
                System.out.println ("Height: " + height);
                System.out.println ( "Windows: " + nWindows);
        } // end display() method
}  // end Room class
```

કોડલિસ્ટિંગ 8.2 : ઈનહેરિટન્સના ઉપયોગનું ઉદાહરણ

હવે, આપણે કોડ લિસ્ટિંગ 8.3માં દર્શાવ્યા પ્રમાણે સુપર ક્લાસ 'Room'ને વિસ્તૃત કરવા કોડ ઉમેરી સબક્લાસ 'Classroom' બનાવીએ. સબક્લાસમાં બે વધારાના nBenches અને nSeatsBench ઇન્સ્ટન્સ વેરિયેબલ છે. અહીં nSeatsBench ચલ એક બેન્ચ ઉપર કેટલાં વિદ્યાર્થીઓ બેસી શકે તે સંખ્યાનો નિર્દેશ કરે છે. તેના પોતાના વધારાના કન્સ્ટ્રક્ટર અને ત્રણ મેથડ show(), display() અને getSeats() છે.

```
class Classroom extends Room
{
        int nBenches, nSeatsBench;
        Classroom( ) {};
        Classroom( float l, float w, float h, byte n, int nB, int nSB)
        {
                super (l,w,h,n);
                nBenches = nB; nSeatsBench = nSB;
        }
        void show()
        {
                super.display();
                System.out.println ("Benches: " + nBenches );
                System.out.println ("Seats per Bench: " + nSeatsBench );
                System.out.println ("Total Seats in a class: " + getSeats() );
        }
        void display()
        {
                System.out.println("\nClassroom with length " + length + " feet, width "
                    + width + " feet\nhas " + nBenches + " Benches, each to accomodate "
                    + nSeatsBench + " Students\nSo, Total seats in a class is  "
                    + getSeats());
        }
        int getSeats() {return nBenches * nSeatsBench; }
} // end class Classroom
```

કોડલિસ્ટિંગ 8.3 : Classroom નામના સબક્લાસનો કોડ

'Classroom' સબક્લાસમાં વપરાશકર્તા વ્યાખ્યાયિત કન્સ્ટ્રક્ટર અને મેથડ show()માં સુપરક્લાસ 'Room'માં લખાયેલા કોડનો ફરી ઉપયોગ થયેલો છે.

સુપર ક્લાસના કન્સ્ટ્રક્ટર સબક્લાસમાં વારસામાં આવતા ન હોવાથી સુપર ક્લાસના કન્સ્ટ્રક્ટરને કૉલ કરવા માટે સબક્લાસના કન્સ્ટ્રક્ટરમાં ચાવીરૂપ શબ્દ 'super' વાપરેલો છે. કન્સ્ટ્રક્ટરમાં આ કૉલ પહેલું વિધાન હોવું જોઈએ. જ્યારે સુપરક્લાસના કન્સ્ટ્રક્ટરનો સ્પષ્ટ રીતે (explicit) કૉલ ન હોય, ત્યારે પહેલા વિધાન તરીકે 'super()'નો કૉલ ગર્ભિત રીતે (implicitly) સુપરક્લાસના ચલ વગરના કન્સ્ટ્રક્ટરનો છે.

Classroom ઑબ્જેક્ટના ઍટ્રિબ્યૂટ પ્રદર્શિત કરવા માટે show() મેથડનો ઉપયોગ કર્યો છે. સુપરક્લાસ 'Room'માંથી ઇનહેરિટ (inherit) થયેલા પ્રથમ ચાર ઍટ્રિબ્યૂટર પ્રદર્શિત કરવા માટે આપણે સુપરક્લાસની display() મેથડના હયાત કોડનો ઉપયોગ કરવા ઇચ્છીએ છીએ. આપણે જાણીએ છીએ કે display() મેથડ સબક્લાસમાં પણ ઉપલબ્ધ છે.

show() મેથડની અંદર આપણો ઇરાદો સુપરક્લાસની display()ને કૉલ કરવાનો છે. આ માટે આપણે super.display()નો ઉપયોગ કર્યો છે.

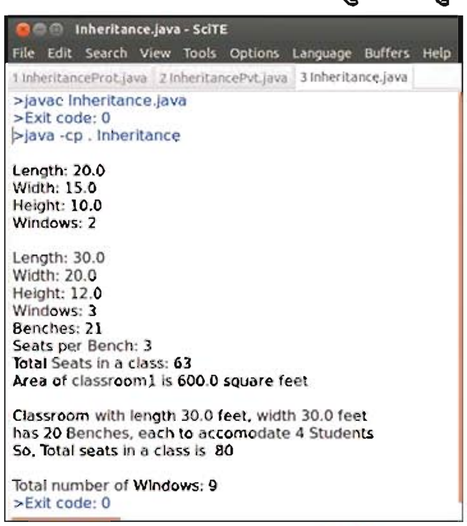
જ્યારે સુપર ક્લાસ અને સબક્લાસમાં એકસમાન સિગ્નેચર (signature) સાથેની મેથડ હોય, ત્યારે સુપર ક્લાસ મેથડની સબક્લાસમાં ઉપેક્ષા કરવામાં આવે છે. સબક્લાસની display() મેથડ સુપરક્લાસની display() મેથડની ઉપેક્ષા કરે છે. એનો અર્થ એ થાય કે આપણે સુપર ક્લાસની display() મેથડનો ફરી ઉપયોગ કર્યા વિના માહિતી જુદી રીતે પ્રદર્શિત કરવા ઇચ્છીએ છીએ. જ્યારે આપણે સુપર ક્લાસની આવી મેથડનો ઉલ્લેખ કરીએ, ત્યારે આપણે ચાવીરૂપ શબ્દ 'super' સાથે ડોટ પ્રક્રિયક અને મેથડનું નામ વાપરવું જોઈએ. અહીં આપણે સુપર ક્લાસની display() મેથડને ઇન્વૉક કરવા માટે show() મેથડની અંદર super.display()નો ઉપયોગ કર્યો છે. getSeats() મેથડ ક્લાસરૂપની બેઠકની ક્ષમતાની ગણતરી કરવા માટે વાપરેલી છે.

કોડલિસ્ટિંગ 8.4માં દર્શાવ્યા પ્રમાણે કોડ ઉમેરીને ચાલો, આપણે આ ક્લાસનો વિનિયોગમાં ઉપયોગ કરીએ. અહીં આપણે સુપર ક્લાસ અને સબક્લાસ બંનેના ઑબ્જેક્ટ બનાવેલા છે.

```
class Inheritance  /* Application using Room, Classroom */
{
        public static void main (String args[])
        {
                Room r1 = new Room(20, 15, 10, (byte)2);
                r1.display();
                Classroom cr1 = new Classroom (30, 20, 12, (byte)3, 21, 3);
                cr1.show();
                System.out.println("Area of classroom1 is " + cr1.area() + " square feet");
                Classroom cr2 = new Classroom (30,30,10, (byte)4, 20, 4);
                cr2.display();
                System.out.println ("\nTotal number of Windows: " + Room.totWindows);
        }  // end main()
}  // end Inheritance
```

કોડલિસ્ટિંગ 8.4 : Room અને Classroomના ઉપયોગ સાથેનો વિનિયોગ

બધા જ ઇન્સ્ટન્સ વેરિયેબલ અને મેથડ સુપર ક્લાસમાંથી સબક્લાસમાં ઇનહેરિટ થાય છે. આથી, સુપર ક્લાસની area() મેથડને સબક્લાસના ઑબ્જેક્ટ દ્વારા પણ cr1.area()નો ઉપયોગ કરી ઇન્વૉક કરી શકાય છે. જ્યારે સબક્લાસના ઑબ્જેક્ટનો ઉપયોગ કરી વિનિયોગમાં ઉપેક્ષા કરેલી મેથડનો ઉલ્લેખ કરવામાં આવે છે, ત્યારે તે સબક્લાસની મેથડ cr2.display()ને કૉલ કરે છે. કોડલિસ્ટિંગ 8.2, 8.3, 8.4ને ભેગાં કરીને બનાવેલા કોડનું આઉટપુટ આકૃતિ 8.15માં દર્શાવ્યું છે.



આકૃતિ 8.15 : Classroom અને Room ક્લાસના ઉપયોગ બાદ ઇનહેરિટન્સનું આઉટપુટ

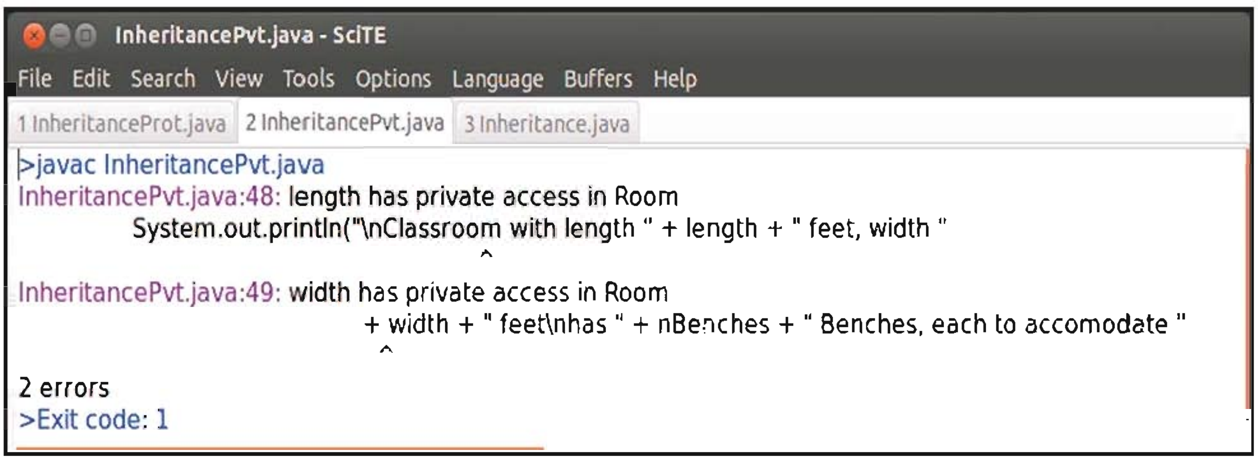
સુપર ક્લાસના પ્રાઈવેટ મેમ્બર (Private Members of Superclass)

અગાઉના ઉદાહરણમાં બધા જ ઇન્સ્ટન્સની પેકેજ વિઝિબિલિટી હતી. આથી, તે આખા પેકેજમાં સીધેસીધા ઉપયોગ માટે ઉપલબ્ધ હતા. આથી, જો આપણે main() મેથડમાં સીધા r1.length અથવા cr1.width એક્સેસ કરવા પ્રયત્ન કરીએ તો તેમાં કોઈ ભૂલ નહીં હોય.

જો આપણે સુપર ક્લાસના ઇન્સ્ટન્સ વેરિયેબલની વિઝિબિલિટી બદલીને private કરીશું, તો આ વેરિયેબલ ક્લાસની બહાર સીધા ઉપલબ્ધ નહીં રહે. યાદ રાખો કે આ ઍટ્રિબ્યૂટ સબક્લાસના પણ છે, છતાં પ્રાઈવેટ ઇન્સ્ટન્સ વેરિયેબલ અથવા મેથડ તેના સબક્લાસમાં પણ વિઝિબલ નથી.

નીચે જણાવ્યા પ્રમાણે કોડલિસ્ટિંગ 8.2માં ઇન્સ્ટન્સ વેરિયેબલને ઘોષિત કરવાના વિધાનમાં તેને private બનાવો અને આકૃતિ 8.16માં દર્શાવ્યા પ્રમાણે તેના આઉટપુટમાં error આવી તેનું નિરીક્ષણ કરો.

```
private float length, width, height;

private byte nWindows;
```



આકૃતિ 8.16 : ક્લાસના પ્રાઈવેટ મેમ્બર તે ક્લાસની બહાર ઍક્સેસિબલ નથી

યાદ રાખો કે પ્રાઈવેટ મેમ્બર તે જે ક્લાસમાં વ્યાખ્યાયિત કરેલા હોય, તેમાં જ ફક્ત સીધા ઉપલબ્ધ હોય છે અને બીજે ક્યાંય પણ નહીં. તેને અન્ય જગ્યાએ ઉપલબ્ધ કરવા માટે પબ્લિક ઍક્સેસર અને મ્યુટેટર મેથડ 'getter' અને 'setter' મેથડનો ઉપયોગ કરો.

સુપરક્લાસના પ્રોટેક્ટેડ મેમ્બર (Protected Members of Superclass)

અગાઉ 'Visibility Modifiers'ના વિષય ઉપર આપણે શીખ્યા કે ઇનહેરિટેડ સબક્લાસમાં 'protected' મેમ્બર 'private' મેમ્બર તરીકે ઉપલબ્ધ હોય છે. નીચે જણાવ્યા પ્રમાણે 'Room' ક્લાસના ઇન્સ્ટન્સ વેરિયેબલને બદલીને 'protected' કરો.

```
protected float length, width, height;

protected byte nWindows;
```

આ બદલાયેલો કોડ જ્યારે અમલમાં મૂકવામાં આવે છે, ત્યારે આકૃતિ 8.15માં દર્શાવ્યા પ્રમાણે આપણે એ જ પરિણામ મેળવીએ છીએ. અહીં એ નોંધવું જોઈએ કે જાવામાં બહુવિધ ઇનહેરિટન્સની સગવડ નથી. સબક્લાસ ફક્ત એક જ સુપરક્લાસમાંથી આવે છે.

કૉમ્પોઝિશન અને એગ્રિગેશન (Composition and Aggregation)

કૉમ્પોઝિશન અને એગ્રિગેશન ક્લાસની રચના છે, જે અન્ય ઑબ્જેક્ટનો સમાવેશ કરે છે. તે અલગ-અલગ ક્લાસ વચ્ચે "has-a" પ્રકારનો સંબંધ રચે છે.

ઉદાહરણ તરીકે, જો આપણે 'Library' નામનો ક્લાસ વ્યાખ્યાયિત કરીએ, તો તેને એક reading room (વાંચનખંડ) હોય છે, અહીં reading room એ 'Room' નામના ક્લાસનો એક ઑબ્જેક્ટ છે. આ રીતે Libraryમાં Room હોય છે. જ્યારે કોઈ ક્લાસ અન્ય ક્લાસના ઑબ્જેક્ટનો સમાવેશ કરે, ત્યારે તે કન્ટેઈનર ક્લાસ (container class) તરીકે ઓળખાય છે.

અન્ય ઉદાહરણ :

- દરેક વ્યક્તિનું name અને address હોય છે. અહીં name ક્લાસ Nameમાં છે, જેને first name, middle name અને last name એમ ત્રણ એટ્રિબ્યૂટ છે. address ક્લાસ Addressમાં છે, જેના એટ્રિબ્યૂટ house number, apartment / society name, area, city, state, country અને pincode છે.
- કારને steering, wheels અને engine હોય છે. અહીં steering ક્લાસ Steeringમાં છે, wheel ક્લાસ Wheelમાં છે અને engine ક્લાસ Engineમાં છે. ક્લાસ Carના ત્રણે એટ્રિબ્યૂટ કારના ભાગ છે.

હવે આપણે નીચે જણાવેલા એટ્રિબ્યૂટ સાથેનો ક્લાસ 'Library' બનાવીએ :

- nBooks: int, પુસ્તકાલયમાં પુસ્તકની સંખ્યા
- nMagazines: int, પુસ્તકાલયમાં લવાજમ ભરેલાં સામયિકની સંખ્યા
- nNewspapers: int, પુસ્તકાલયમાં લવાજમ ભરેલાં વર્તમાનપત્રકની સંખ્યા
- readingRoom: Room

અહીં readingRoom એ બેઝિક ડેટાટાઈપનો નથી તે જુઓ. તેનો પ્રકાર 'Room' ક્લાસનો છે.

કોડ લિસ્ટિંગ 8.5માં Room અને Library નામના ક્લાસ બનાવવાનો કોડ અને તેનો 'Container.java' નામના વિનિયોગમાં ઉપયોગ દર્શાવ્યો છે.

```
/* Using objects as data members in a container class */
class Room
{
        protected float length, width, height;
        protected byte nWindows;
        static int totWindows;  // class variable

        Room ( ) { }; // user-defined no-argument constructor
        Room (float l, float w, float h, byte n)
        {
                length = l; width = w; height = h;
                nWindows = n;    totWindows+=n;
        }
        Room (float l, float w)
        {
                length = l; width = w; height = 10;
                nWindows = 1;        totWindows++;
        }
```

```
            double area ( )  // area =  length * width
           {      return (length * width);        } // end area() method
            void display ( )
           {
                  System.out.println ("Length: " + length + "\nWidth: " + width);
                  System.out.println ("Height: " + height);
                  System.out.println ( "Windows: " + nWindows);
           } // end display() method
}  // end Room class

class Library
{
           int nBooks, nMagazines, nNewspapers;
           Room readingRoom;
           Library( ) {};
           Library( int nB, int nM, int nN, Room r)
           {
                  nBooks = nB; nMagazines=nM; nNewspapers=nN;
                  readingRoom=r;
           }
           void display()
           {
                  System.out.println ("\nLibrary Details:\nNumber of books: " + nBooks );
                  System.out.println ("Number of subscribed magazines: " + nMagazines );
                  System.out.println ("Number of subscribed newspapers : " + nNewspapers);
                  System.out.println ("Reading Room:");
                  readingRoom.display();
           }
}  // end class Library

class Container  /* Application using Room, Library */
{
           public static void main (String args[])
           {
                  Room r1 = new Room(20, 15, 10, (byte)2);
                  r1.display();
                  Library lib = new Library (300, 20, 5, r1);
                  lib.display();
           }  // end main()
}  // end class Container
```

કોડલિસ્ટિંગ 8.5 : કોમ્પોઝિશન અને એગ્રિગેશનનું ઉદાહરણ

કોડલિસ્ટિંગ 8.5માં નીચેના મુદ્દાઓનું નિરીક્ષણ કરો :

- ક્લાસ 'container'ની main() મેથડમાં :
  - ક્લાસ 'Library'નો ઑબ્જેક્ટ lib ચાર ચલ (arguments) સાથે કન્સ્ટ્રક્ટર વાપરીને બનાવવામાં આવ્યો છે. Room ક્લાસનો r1 એ છેલ્લો ચલ છે.
  - ક્લાસ 'Library'ની મેથડ display() ઑબ્જેક્ટ libનો ઉપયોગ કરીને ઇન્વૉક કરેલી છે.
- ક્લાસ 'Library'માં :
  - ઑબ્જેક્ટ readingRoom ક્લાસ 'Room'નો ઍટ્રિબ્યૂટ છે.
  - ચાર ચલ (arguments) સાથેના કન્ટ્રક્ટરમાં ક્લાસ 'Room'ની છેલ્લી ચલ 'r'નો ઉપયોગ કરે છે અને ઍસાઇન્મેન્ટ વિધાન 'readingRoom = r;' વાપરીને Roomના ઍટ્રિબ્યૂટને કિંમતો ઍસાઇન (assign) કરે છે.
  - display() મેથડ વ્યાખ્યાયિત કરેલી છે 'readingRoom.display();' દ્વારા 'Room' ક્લાસની display() મેથડ ઇન્વૉક કરે છે. અહીં નોંધ કરશો કે display() મેથડની ઉપેક્ષા કરી નથી, આપણે ઇનહેરિટન્સનો ઉપયોગ કર્યો નથી. વાચક આ મેથડ માટે અન્ય નામ વાપરી શકે. જેમકે, 'lib.display()'ને બદલે 'Library' ક્લાસમાં show()નો ઉપયોગ main() મેથડમાં 'lib.show()' વડે.

આવા સંજોગોમાં વાચક એવું વિચારે કે - ઇનહેરિટન્સ શા માટે ન વાપરવું ? શા માટે 'Library' ક્લાસને 'Room' ક્લાસમાંથી ઇનહેરિટ ન કરવો અને તેમાં ત્રણ વધારાના ઍટ્રિબ્યૂટ ઉમેરવા ?

અહીં નોંધ કરશો કે 'Library' એ 'Room' પ્રકારનો નથી. 'Library' અને 'Room' વચ્ચે 'is-a' સંબંધ નથી, પણ 'has-a' સંબંધ છે. 'Library'માં 'Room' હોય, જેનો રીડિંગરૂમ તરીકે ઉપયોગ થાય છે. આથી, readingRoomને આપણે 'Room'ના પ્રકારનો જણાવ્યો છે અને તે 'Library' ક્લાસનો ઍટ્રિબ્યૂટ છે.

સારાંશ

આપણે ક્લાસ અને ઑબ્જેક્ટ કઈ રીતે બનાવવા, ક્લાસના ઇન્સ્ટન્સ વેરિયેબલ કેવી રીતે ઍક્સેસ કરવા અને ઑબ્જેક્ટ વડે ઇન્સ્ટન્સ મેથડ ઇન્વૉક કરવી તે બાબત અહીં શીખ્યા. કન્સ્ટ્રક્ટર એ ક્લાસના નામની જ, ચલ વગરની અને પરત પ્રકાર વગરની વિશિષ્ટ મેથડ છે. જ્યારે ઑબ્જેક્ટ નવા પ્રક્રિયક સાથે ઇન્સ્ટેન્સિયેટ કરવામાં આવે છે, ત્યારે ગર્ભિત રીતે કન્સ્ટ્રક્ટર કૉલ કરવામાં આવે છે, ઇન્સ્ટન્સ વેરિયેબલ દરેક ઑબ્જેક્ટના હોય છે. ક્લાસ વેરિયેબલ અને મેથડ 'static' ચાવીરૂપ શબ્દ વાપરીને વ્યાખ્યાયિત કરવામાં આવે છે. સ્ટેટિક મેમ્બરને ક્લાસ દીઠ ફક્ત એક જ વાર મેમરી ફાળવવામાં આવે છે અને ક્લાસના બધા ઑબ્જેક્ટ દ્વારા તેનો સહિયારો ઉપયોગ કરે છે, તે ક્લાસના હોય છે; ઑબ્જેક્ટના નહીં. આપણે ઍક્સેસ મોડિફાયર વિશે પણ અભ્યાસ કર્યો, જે ઇન્સ્ટન્સ મેમ્બરની વિઝિબિલિટી નક્કી કરે છે. પ્રાઇવેટ મેમ્બર ફક્ત ક્લાસની અંદર જ વિઝિબલ હોય છે કે જ્યાં તે વ્યાખ્યાયિત કરેલ હોય છે, પ્રોટેક્ટેડ મેમ્બર ફક્ત ઇનહેરિટ સબક્લાસમાં જ વિઝિબલ હોય છે, પૅકેજમેમ્બર પૅકેજમાં સર્વત્ર વિઝિબલ હોય છે અને પબ્લિક મેમ્બર બધી જ જગ્યાએ વિઝિબલ હોય છે. સુરક્ષાના હેતુ માટે પ્રાઇવેટ ઇન્સ્ટન્સ વેરિયેબલનો ઉપયોગ અને 'getters' અને 'setters' મેથડનો public તરીકે ઉપયોગ કરવો સલાહભર્યું છે. અંતમાં આપણે હયાત ક્લાસમાંથી નવો ક્લાસ ઇનહેરિટ કરવો, મેથડનો ફરી ઉપયોગ, તેને વિસ્તૃત કરવી અને તેની ઉપેક્ષા કરવી વિશે શીખ્યા. આપણે ક્લાસમાં ઇન્સ્ટન્સ વેરિયેબલ તરીકે ઑબ્જેક્ટનો ઉપયોગ પણ જોયો. તે ક્લાસને ઑબ્જેક્ટના કૉમ્પોઝિશન અને ઍગ્રિગેશન બનવાનું સામર્થ્ય પૂરું પાડે છે.

સ્વાધ્યાય

1. ઑબ્જેક્ટનું ઇન્સ્ટેન્સિએશન એટલે શું ?
2. ક્લાસ-વેરિયેબલની જરૂરિયાત બાબત એક ઉદાહરણ આપો.
3. મેથડ અને કન્સ્ટ્રક્ટર વચ્ચેનો તફાવત જણાવો.

4. ઍક્સેસર અને મ્યુટેટર મેથડ વિશે જણાવો.

5. સબક્લાસમાંથી સુપરક્લાસનો કન્સ્ટ્રક્ટર કેવી રીતે ઇન્વૉક કરી શકાય ?

6. સુપરક્લાસની ઉપેક્ષા કરેલી મેથડ સબક્લાસમાંથી કેવી રીતે ઇન્વૉક કરી શકાય ?

7. 'ઍક્સેસ મૉડિફાયર' વિશે ટૂંક નોંધ લખો.

8. ઇનહેરિટન્સનો ઉપયોગ અને અલગ-અલગ ક્લાસ વચ્ચે સંબંધના પ્રકાર પ્રમાણે કૉમ્પોઝિશન અથવા ઍગ્રિગેશનનો ઉપયોગ સમજાવો.

9. મેથડ ઓવરલોડિંગ અને મેથડ ઓવરરાઇડિંગ વચ્ચેનો તફાવત સમજાવો.

10. નીચે આપેલા વિકલ્પોમાંથી સાચો વિકલ્પ પસંદ કરો :

(1) નીચેનામાંથી ઍટ્રિબ્યૂટ અને મેથડને કોણ વ્યાખ્યાયિત કરે છે ?

(a) ક્લાસ (b) ઑબ્જેક્ટ (c) ઇન્સ્ટન્સ (d) વેરિયેબલ

(2) નીચેનામાંથી ક્યો ચાવીરૂપ શબ્દ ક્લાસ વેરિયેબલ અને ક્લાસમેથડને ઘોષિત કરવા માટે વપરાય છે ?

(a) static (b) private (c) public (d) package

(3) નીચેનામાંથી ક્યો પ્રક્રિયક ઑબ્જેક્ટ બનાવે છે અને તેનો રેફરન્સ પરત કરે છે ?

(a) ડોટ (.) (b) new (c) કોલન (:) (d) ઍસાઇનમેન્ટ (=)

(4) ક્લાસનો ઇન્સ્ટન્સ બનાવ્યા વિના નીચેનામાંથી કઈ મેથડ કૉલ કરી શકાય ?

(a) ઇન્સ્ટન્સ મેથડ (b) ક્લાસ મેથડ

(c) કન્સ્ટ્રક્ટર મેથડ (d) ઉપરના બધા વિકલ્પ

(5) નીચેનામાંથી એકસમાન નામ ધરાવતી પણ અલગ-અલગ પ્રાચલો સાથેની એક કરતાં વધારે મેથડનો ઉલ્લેખ કોણ કરે છે ?

(a) ઓવરલોડેડ મેથડ્સ (b) ઓવરરીડન મેથડ્સ

(c) ડુપ્લિકેટ મેથડ્સ (d) ઉપરના બધા વિકલ્પ

(6) નીચેનામાંથી કઈ મેથડ ઑબ્જેક્ટ બનાવતા સમયે આપોઆપ ઇન્વૉક થાય છે ?

(a) ઇન્સ્ટન્સ મેથડ (b) કન્સ્ટ્રક્ટર

(c) ક્લાસ મેથડ (d) ઉપરના બધા વિકલ્પ

(7) નીચેનામાંથી ક્યો ચાવીરૂપ શબ્દ સબક્લાસ કન્સ્ટ્રક્ટરમાં સુપર ક્લાસ કન્સ્ટ્રક્ટરનો ઉલ્લેખ કરે છે ?

(a) extends (b) super

(c) સુપર ક્લાસનું નામ (d) new

(8) નીચેનામાંથી જાવામાં ઇન્સ્ટન્સ મેથડ ઇન્વૉક કરવા માટે વપરાય શું છે ?

(a) ઑબ્જેક્ટનું નામ, કોલોન(:) અને મેથડનું નામ

(b) ઑબ્જેક્ટનું નામ, ડોટ(.) અને મેથડનું નામ

(c) ક્લાસનું નામ, કોલોન(:) અને મેથડનું નામ

(d) ક્લાસનું નામ, ડોટ(.) અને મેથડનું નામ

(9) નીચેનામાંથી ઇન્સ્ટન્સ મેથડ વડે શું ઍક્સેસિબલ છે ?

(a) ફક્ત ઇન્સ્ટન્સ વેરિયેબલ (b) ફક્ત ક્લાસ વેરિયેબલ

(c) બંને ક્લાસ વેરિયેબલ અને ઇન્સ્ટન્સ વેરિયેબલ (d) ઉપરના બધા વિકલ્પ

(10) જ્યારે સુપરક્લાસમાં મેથડનું નામ અને સિગ્નેચર સમાન હોય ત્યારે તેને શું કહેવામાં આવે છે ?

(a) ઓવરલોડેડ મેથડ્સ (b) ઓવરરીડન મેથડ્સ

(c) ઇનહેરિટેડ મેથડ્સ (d) ઉપરના બધા વિકલ્પ

પ્રાયોગિક સ્વાધ્યાય

નીચેનાં કાર્ય માટે જાવાપ્રોગ્રામ લખો :

1. 'FixedDeposit' નામનો ક્લાસ બનાવો, જેના ત્રણ ઍટ્રિબ્યૂટ (મુદલ રકમ, વાર્ષિક વ્યાજનો દર અને વર્ષમાં જમા રકમનો સમયગાળો) અને મેથડ હોય કે જે ચક્રવૃદ્ધિ વ્યાજની રીતે પાકતી મુદતની રકમ પરત કરે. main() મેથડ સાથેનો અન્ય 'FixedDepositDemo' નામનો ક્લાસ બનાવો. main() મેથડમાં બે ઑબ્જેક્ટ બનાવો, તેના ઍટ્રિબ્યૂટને કિંમતો ઍસાઇન કરો અને તેને પાકતી મુદતની રકમ સાથે પ્રદર્શિત કરો.

2. 'FixedDeposit' ક્લાસમાં નીચેના કન્સ્ટ્રક્ટર ઉમેરો અને તેનો ઑબ્જેક્ટ બનાવવામાં ઉપયોગ કરો.

   a. ચલ વગરના કન્સ્ટ્રક્ટર કે જે મુદલ રકમને 1000, વાર્ષિક વ્યાજના દરને 5% અને જમા રકમના સમયગાળાને 3 વર્ષ પ્રારંભિક કિંમત (initialize) આપે.

   b. ત્રણ ઍટ્રિબ્યૂટને પ્રારંભિક કિંમત આપવા 3 ચલ (arguments) સાથેનો પ્રાચલોવાળો કન્સ્ટ્રક્ટર.

3. પ્રાયોગિક સ્વાધ્યાય-1માં 'FixedDeposit' ક્લાસના ઇન્સ્ટન્સ વેરિયેબલની વિઝિબિલિટી બદલીને 'private' કરો અને તેનો અમલ કરવા પ્રયત્ન કરો. તે error આપશે ? જો હા હોય તો, ઇન્સ્ટન્સ વેરિયેબલની કિંમત પ્રદર્શિત કરવા માટે ક્લાસમાં display() મેથડ ઉમેરો.

4. 'FixedDeposit' ક્લાસનાં ત્રણ ઍટ્રિબ્યૂટને get અને set કરવા માટે ઍક્સેસર અને મ્યુટેટર મેથડ ઉમેરો. આ મેથડનો ઉપયોગ કરીને main() મેથડના પ્રાઇવેટ ઇન્સ્ટન્સ વેરિયેબલની કિંમત પ્રદર્શિત કરો.

5. કુલ મુદલ જમા રકમ ધરાવતા ક્લાસમાં 'totDeposit' ક્લાસ વેરિયેબલ ઉમેરો. કન્સ્ટ્રક્ટર અને સેટર મેથડમાં ફેરફાર કરો, જેથી કુલ જમા રકમ મેળવી શકાય. 'totDeposit' ચલની કિંમત પ્રદર્શિત કરવા મેથડ લખો અને તેનો ઉપયોગ બતાવો.

6. 'Rectangle' ક્લાસનો ઉપયોગ કરી 'Box' નામનો સબક્લાસ વધારાના 'height' ઍટ્રિબ્યૂટ અને 'volume' મેથડ સાથે ઉત્પન્ન કરો (નિર્માણ કરો - derive). (ઘનફળ = ઊંચાઈ x પહોળાઈ x લંબાઈ = ઊંચાઈ x ક્ષેત્રફળ)

ઍરે અને સ્ટ્રિંગનો ઉપયોગ 9

પ્રકરણ 7માં આપણે જાવામાં ઉપલબ્ધ ડેટાના મૂળભૂત પ્રકારોનો અભ્યાસ કર્યો. મૂળભૂત પ્રકારના ચલ (variable)માં એક સમયે ફક્ત એક જ કિંમત રાખી શકાય છે. આ પ્રકારના ચલને અદિશ ચલ કે સ્કેલર વેરિયેબલ (scalar variable) કહેવામાં આવે છે. આ પ્રકરણમાં આપણે કેટલાક સંયોજિત કે કૉમ્પોઝિટ (composite) ડેટાપ્રકારો વિશે અભ્યાસ કરીશું જેનો ઉપયોગ એક કરતાં વધારે ડેટાકિંમતોના સમૂહનો સંગ્રહ કરવા માટે થાય છે, ઉદાહરણ તરીકે, ઍરે (array) અને સ્ટ્રિંગ (string).

ઍરેનો પરિચય (Introduction to Array)

ઍરે (array) એ એક જ પ્રકારના ઘટકોના સંગ્રહને રજૂ કરતો ચલ (variable) છે. ઍરે સદિશ (vector), શ્રેણિક (matrix) અને અન્ય બહુ-પરિમાણીય (multi-dimensional) માહિતી રજૂ કરવા માટે ઉપયોગી છે. વૅક્ટર (vector) એ એક-પરિમાણીય (1-D) ડેટા-સંરચના (data structure) છે, જે અક્ષરો, સંખ્યાઓ જેવી વસ્તુઓની યાદી સંગ્રહવા માટે વાપરી શકાય છે. રો (આડી હરોળ) અને કૉલમ (ઊભી હરોળ) વડે બનેલી કોષ્ટક જેવી દ્વિ-પરિમાણીય (2-D) ડેટા-સંરચના રજૂ કરવા માટે મૅટ્રિક્સ (matrix) વપરાય છે.

જ્યારે સમાન પ્રકારના અનેક ઘટકો ઉપર એકસમાન કાર્ય કરવાનું હોય, ત્યારે ઍરે ઉપયોગી બને છે. ઍરેના તમામ ઘટકો મેમરીમાં એકસાથે લગોલગ સંગ્રહાયેલા હોય છે. ઍરેના દરેક ઘટકને ઍરે વેરિયેબલ સાથે સંકળાયેલા સૂચક સ્થાનાંક (index position) વડે ઓળખવામાં આવે છે.

જાવામાં વસ્તુઓની યાદીનું સંચાલન કરવા માટે વપરાતો ઑબ્જેક્ટ એ ઍરે છે. ઍરે બનાવવા માટે બે પગલાંની ક્રિયા છે :

1. ઍરે-ઑબ્જેક્ટ ઘોષિત કરો.
2. ઍરે-ઑબ્જેક્ટની રચના કરો.

ઍરે-ઑબ્જેક્ટ બે રીતથી બનાવી શકાય છે :

1. new પ્રક્રિયક વાપરીને અને તેનું કદ જણાવીને.
2. સીધેસીધા ઍરેના ઘટકોને પ્રારંભિક કિંમત આપીને. (initializing)

એક-પરિમાણીય ઍરે (1-D Array)

એક-પરિમાણવાળા ઍરે 1-D ઍરે તરીકે ઓળખાય છે. ઉદાહરણ તરીકે, વૅક્ટર (vector) કે જેને એક અથવા વધારે અદિશ ચલ (scalar variables)ના સમૂહ તરીકે ગણી શકાય. marks1, marks2, marks3, marks4, marks5ની રીતે અલગ-અલગ ચલ ઘોષિત કરવાના બદલે આપણે ઍરે marks[5] ઘોષિત કરી શકીએ અને તેના દરેક ઘટકને marks[0], marks[1], marks[2], marks[3], marks[4] વાપરી આપણે તેને મેળવી શકીએ છીએ.

1-D ઍરે ઘોષિત કરવા માટે ચોરસ કૌંસની જોડી [ ] ઍરેના નામ પછી અથવા ડેટાના પ્રકાર પછી આપણે વાપરીએ છીએ. ઍરે ઘોષિત કરવાની વાક્યરચના નીચે પ્રમાણે છે :

<data type> <array name> []; અથવા <data type> [] <array name>;

ઉદાહરણ તરીકે, એક વિદ્યાર્થીના ગણિત વિષયની પાંચ કસોટીમાં મેળવેલા ગુણનો સંગ્રહ કરવા માટે આપણે પાંચ પૂર્ણાંક સંખ્યા ધરાવતા ઘટકોનો ઍરે વાપરી શકીએ. 'marks' નામનો પાંચ ઘટકો (elements) ધરાવતો ઍરે-ઑબ્જેક્ટ નીચે મુજબ બનાવી શકાય છે :

```
int marks[];                // declare array object
marks = new int [5];        // create array object
```

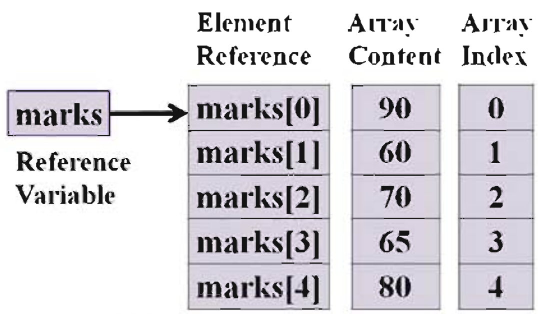
ઉપરનાં બંને વિધાનોને ભેગાં કરીને નીચે મુજબ એક જ વિધાનથી પણ આપણે ઍરે બનાવી શકીએ :

int marks[] = new int [5];   અથવા   int [] marks = new int [5];

અહીં ઍરેનું નામ 'marks' છે. પાંચ પૂર્ણાંક સંખ્યાઓને સંગ્રહ જ્યાં થશે, તે મેમરી સ્થાનાંકનો તે નિર્દેશ કરે છે. int ડેટા પ્રકારની પૂર્ણાંક સંખ્યાના સંગ્રહ માટે 4 બાઇટ વપરાય છે. આથી, 'marks' ઍરે માટે મેમરીમાં સળંગ 5 x 4 = 20 બાઇટની જરૂર પડે છે.

ઍરેના કોઈ ઘટકનો નિર્દેશ કરવા માટે આપણે આકૃતિ 9.1માં દર્શાવ્યા પ્રમાણે ઍરેના નામ પછી મોટા કૌંસ [ ]ની અંદર ઇન્ડેક્સ (index) અથવા સબસ્ક્રિપ્ટ (subscript)નો ઉપયોગ કરીએ છીએ. ઇન્ડેક્સ કે સબસ્ક્રિપ્ટ ઍરેમાં ઘટકનું સ્થાન દર્શાવે છે. ઉદાહરણ તરીકે, marks[2]. ઇન્ડેક્સની કિંમત શૂન્ય (0)થી શરૂ થાય છે.

```
int marks[] = {90, 60, 70, 65, 80};
```

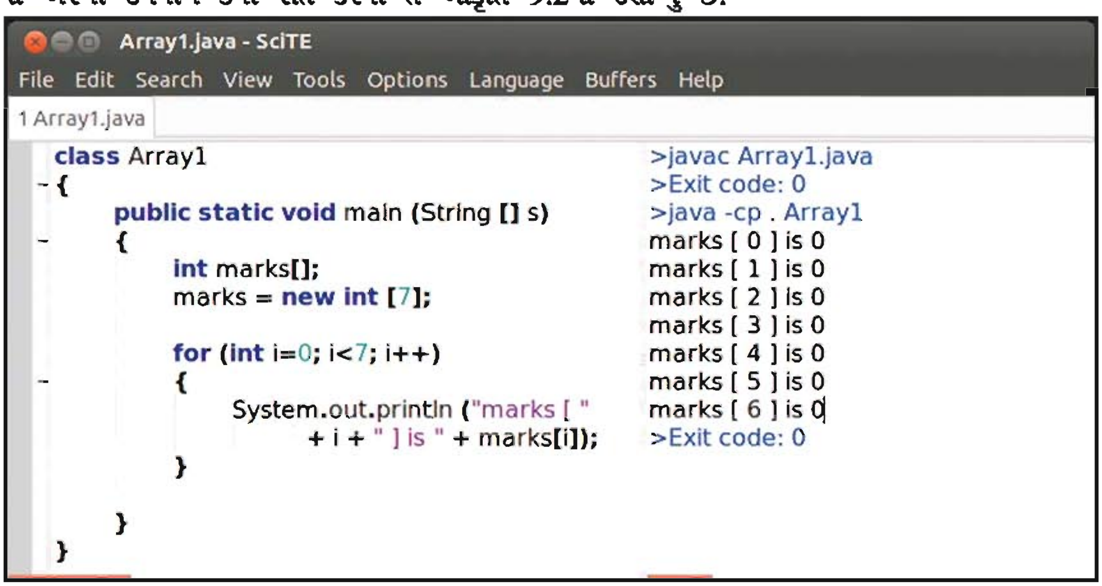


આકૃતિ 9.1 : એક-પરિમાણીય ઍરે

આકૃતિ 9.1માં ઍરે ચલ marksની ઇન્ડેક્સકિંમત 0થી 4 છે. અહીં, marks[0] ઍરેના પ્રથમ ઘટકનો નિર્દેશ કરે છે અને marks[4] એ છેલ્લા ઘટકનો નિર્દેશ કરે છે.

પ્રકરણ 8માં સમજૂતી આપ્યા પ્રમાણે ઑબ્જેક્ટ એક રેફરન્સ ચલ છે. ઍરે જાવામાં એક ઑબ્જેક્ટ હોવાથી ઍરેનું નામ પણ એક રેફરન્સ ચલ છે. તે મેમરીના એ સ્થાનાંકનો રેફરન્સ ધરાવે છે, જ્યાં ઍરેના ઘટકોનો સંગ્રહ થયેલો છે. આકૃતિ 9.1 જુઓ.

ઍરે એક ઑબ્જેક્ટ છે, આથી તેના ઘટકોને પૂર્વનિર્ધારિત (default) કિંમતો આપવામાં આવે છે (initialized). જાવા-પ્રોગ્રામમાં ઍરેનો ઉપયોગ કેવી રીતે કરવો તે આકૃતિ 9.2માં દર્શાવ્યું છે.



આકૃતિ 9.2 : ઍરે-ઑબ્જેક્ટના ઘટકોને પૂર્વનિર્ધારિત કિંમતો પ્રારંભિક કિંમત તરીકે આપવી

જ્યારે ઍરે ઘોષિત કરવામાં આવે છે, તે સમયે પણ તેના ડેટા ઘટકોની પ્રારંભિક કિંમત (initial values) જણાવી શકાય છે. 1-D ઍરેના ઘટકોની પ્રારંભિક કિંમતો છગડિયા કૌંસ { }માં અલ્પવિરામ વડે અલગ પાડેલી કિંમતો વડે આપી શકાય છે. ઉદાહરણ તરીકે,

int marks[ ] = {90, 70, 77}; અથવા **int [] marks = {90, 70, 77};**

અહીં નોંધ કરો કે આપણે ઍરે બનાવતા સમયે જો તેના ઘટકોની પ્રારંભિક કિંમતો આપીએ, તો new પ્રક્રિયકના ઉપયોગની જરૂર રહેતી નથી. ઍરેનું કદ કૌંસમાં આપેલી કિંમતોની સંખ્યા બરાબર છે.

આકૃતિ 9.3માં આપેલો જાવાકોડ ઍરે બનાવવા માટે તેમજ તેના ઘટકોની પ્રારંભિક કિંમતો આપવા માટેની અલગ-અલગ રીતો દર્શાવે છે. ઍરેના ઘટકોની કિંમતો પ્રદર્શિત કરવા માટે વાપરેલી display() ક્લાસ મેથડની નોંધ લો. ક્લાસમેથડને static ઘોષિત કરાય છે અને તે ક્લાસના કોઈ પણ ઑબ્જેક્ટ વિના વાપરી શકાય છે.

```
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 Array2.java

/* creating and intializing 1-D array  in different ways */
class Array2
{
    public static void main (String [] s)
    {
        int marks1[];
        marks1 = new int [3];

        int marks2[] = new int[3];
        int [] marks3 = new int [3];
        int marks4[] = {50, 60, 70};
        int [] marks5 = {70, 80, 90};

        System.out.print ("Array marks1:\t");
        display(marks1,3);
        System.out.print ("Array marks2:\t");
        display(marks2,3);
        System.out.print ("Array marks3:\t");
        display(marks3,3);
        System.out.print ("Array marks4:\t");
        display(marks4,3);
        System.out.print ("Array marks5:\t");
        display(marks5,3);
    }

    static void display(int arr[], int size)
    {
        for (int i=0; i<size; i++)
        {
            System.out.print (arr[i] + "\t");
        }
        System.out.println();
    }
}
```

```
>javac Array2.java
>Exit code: 0
>java -cp . Array2
Array marks1:  0   0   0
Array marks2:  0   0   0
Array marks3:  0   0   0
Array marks4:  50  60  70
Array marks5:  70  80  90
>Exit code: 0
```

આકૃતિ 9.3 : ઍરે-ઑબ્જેક્ટ બનાવવા તથા તેના ઘટકોને પ્રારંભિક કિંમતો આપવાની વિવિધ રીત

જાવામાં આપણે જ્યારે ઍરે ઘોષિત કરીએ, તે સમયે પરિમાણનું માપ (dimensions) અને તેના ઘટકોની પ્રારંભિક કિંમતો (initial values) બંને એકસાથે આપી શકતા નથી. આકૃતિ 9.4માં આ વિધાનને સમજાવતો કોડ દર્શાવ્યો છે. જો આપણે ઍરે ચલ પ્રારંભિક કિંમતો વગર (without initialization) ઘોષિત કરીએ, તો આકૃતિ 9.1માં દર્શાવ્યા પ્રમાણે સંદર્ભ રાખવા માટેનો માત્ર ચલ બને છે. એ સમયે ઍરે-ઑબ્જેક્ટ બનતો નથી, એટલે કે ઍરેના ઘટકો માટે મેમરી ફાળવવામાં આવતી નથી. જો આપણે ઍરેના ઘટકો વાપરવા માટે પ્રયત્ન કરીએ તો તે ભૂલમાં પરિણમે છે.

```
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 Array3.java

/* creating and intializing 1-D array
   specifiyng size with array variable*/

class Array3
{
    public static void main (String [] s)
    {
        int marks4[3] = {50, 60, 70};
        int [5] marks5 = {70, 80, 90};

        System.out.print ("Array marks4:\t");
        display(marks4,3);
        System.out.print ("Array marks5:\t");
        display(marks5,3);
    }

    static void display(int arr[], int size)
    {
        for (int i=0; i<size; i++)
        {
            System.out.print (arr[i] + "\t");
        }
        System.out.println();
    }
}
```

```
>javac Array3.java
Array3.java:8: ']' expected
        int marks4[3] = {50, 60, 70};
                   ^
Array3.java:8: illegal start of expression
        int marks4[3] = {50, 60, 70};
                    ^
Array3.java:8: illegal start of expression
        int marks4[3] = {50, 60, 70};
                      ^
Array3.java:8: not a statement
        int marks4[3] = {50, 60, 70};
                       ^
Array3.java:8: ';' expected
        int marks4[3] = {50, 60, 70};
                         ^
Array3.java:9: ']' expected
        int [5] marks5 = {70, 80, 90};
             ^
Array3.java:9: ';' expected
        int [5] marks5 = {70, 80, 90};
              ^
Array3.java:9: <identifier> expected
        int [5] marks5 = {70, 80, 90};
                     ^
Array3.java:11: <identifier> expected
        System.out.print ("Array marks4:\t");
                         ^
Array3.java:11: illegal start of type
        System.out.print ("Array marks4:\t");
```

આકૃતિ 9.4 : ઍરે-ઑબ્જેક્ટના ઘટકોને પ્રારંભિક કિંમતો આપતા સમયે તેનું કદ જણાવવું અમાન્ય છે.

ઍરેનો દરેક ઘટક એક અલગ ચલ છે, જેનો ઇન્ડેક્સ (index) વાપરીને નિર્દેશ કરી શકાય છે. ઍરેના ઘટકની કિંમત બદલવા માટે અન્ય ચલની જેમ આપણે ઘટક ચલને ઍસાઇન્મેન્ટ વિધાનમાં ડાબી બાજુએ વાપરી શકીએ છીએ. ઉદાહરણ તરીકે, **marks[3] = 56;** ચાલો, આપણે 10 અપૂર્ણાંક સંખ્યાઓની સરેરાશ (મધ્યક) શોધવા માટેનો પ્રોગ્રામ લખીએ અને તેનો અમલ કરીએ. આકૃતિ 9.5માં આપેલું કોડલિસ્ટિંગ અને તેના આઉટપુટનો અભ્યાસ કરો.

```
ArrayAvg.java - SciTE
File Edit Search View Tools Options Language Buffers Help
1 ArrayAvg.java

/* compute average of 10 numbers */
class ArrayAvg
{
    public static void main (String [] s)
    {
        double numbers [] = { 10.5, 20.6, 30.8, 15.5,
                              17.3, 25.5, 27.2,
                              20, 30, 18.5};

        byte ctr;
        double sum=0, avg;

        System.out.println ("list of numbers is");
        for (ctr=0; ctr<10; ctr++)
        {
            System.out.println (numbers[ctr]);
            sum = sum + numbers[ctr];
        }
        avg = sum/10;
        System.out.println ("\nAverage of above numbers is "
                            + avg);
    } // main
} // class
```

```
>javac ArrayAvg.java
>Exit code: 0
>java -cp . ArrayAvg
list of numbers is
10.5
20.6
30.8
15.5
17.3
25.5
27.2
20.0
30.0
18.5

Average of above numbers is 21.59
>Exit code: 0
```

આકૃતિ 9.5 : 1-D ઍરે અને લૂપનો ઉપયોગ કરીને 10 સંખ્યાની સરેરાશ શોધવાનો પ્રોગ્રામ

અહીં, આપણને સમાન પ્રકારના અલગ-અલગ ઘટકો ઉપર એક જ જાતની ક્રિયા 'સંખ્યાને સરવાળામાં ઉમેરો', વારંવાર કરવાની જરૂર પડે છે. એકસાથે 10 ચલને ઘોષિત કરવા અને લૂપનો ઉપયોગ કરી ઍરેના જુદા-જુદા ઘટકો ઉપર એકસમાન પ્રક્રિયા કરવામાં ઍરેનો ઉપયોગ મદદરૂપ થાય છે.

ઍરે ઉપર આપણે અન્ય અનેક વિવિધ પ્રક્રિયાઓ કરી શકીએ છીએ. ઉદાહરણ તરીકે, બે ઍરેની સરખામણી કરવી, એક ઍરેના બધા ઘટકોની અન્ય ઍરેમાં નકલ કરવી, કોઈ ચોક્કસ ઘટક ઍરેમાં શોધવો, ઍરેના ઘટકોને ક્રમાનુસાર ગોઠવવા વગેરે. આ પ્રકારનાં બધાં કાર્યો માટે જેમ આપણે ઘટકોનું સરેરાશ શોધવા માટે લખી હતી તે પ્રમાણે પ્રોસીજર (procedures) લખી શકીએ. આ માટેનો કોડ આપણે જાતે લખવાને બદલે જાવા દ્વારા પૂરી પાડવામાં આવેલી java.util.Arrays ક્લાસની વિવિધ static methodsનો આપણે ઉપયોગ કરી શકીએ છીએ. જાવામાં ઍરેને Arrays classની મેથડ (methods)નો ઉપયોગ કરવાની સગવડ આપે છે. આકૃતિ 9.6માં દર્શાવેલું કોડલિસ્ટિંગ અને તેનો આઉટપુટ java.util.Arrays ક્લાસની sort() અને fill() મેથડનો ઉપયોગ કઈ રીતે કરી શકાય તે દર્શાવે છે.

આપણે સમગ્ર ઍરે કે તેના અમુક ભાગને ક્રમાનુસાર ગોઠવવા માટે (શૉર્ટ કરવા) sort() મેથડ વાપરી શકીએ છીએ. આપણે જ્યારે મેથડના આર્ગ્યુમૅન્ટ (argument) તરીકે ફક્ત ઍરે આપીએ, ત્યારે તે સમગ્ર ઍરેને (ઍરેના બધા ઘટકોને) સૉર્ટ કરે છે. જો આપણે ઍરે, આરંભસ્થાન, અંતિમસ્થાન એમ પણ આર્ગ્યુમૅન્ટ આપીને મેથડ વાપરીએ, તો તે આરંભસ્થાનના ઘટકથી અંતિમસ્થાનથી આગળના ઘટક સુધીના આંશિક ઍરેને શોટ (sort) કરશે. ઉદાહરણ તરીકે, java.util.Arrays.sort (list, 1, 5)નો ઉપયોગ કરવાથી તે list[1] થી list[5-1] સુધીના ઍરેના ઘટકોને શોટ કરશે. જુઓ આકૃતિ 9.6.

સમગ્ર કે આંશિક ઍરેને કોઈ ચોક્કસ કિંમતથી ભરવા માટે 'fill' મેથડનો ઉપયોગ થાય છે. જ્યારે મેથડ ઍરે અને કિંમત એમ બે આર્ગ્યુમૅન્ટ સાથે ઇન્વોક કરવામાં આવે, ત્યારે ઍરેના બધા ઘટકોમાં તે જણાવેલી કિંમત ભરવામાં આવે છે. જ્યારે આ મેથડ ઍરે, આરંભસ્થાન, અંતિમ સ્થાન અને કિંમત એમ ચાર આર્ગ્યુમૅન્ટ સાથે ઇન્વોક કરવામાં આવે છે, ત્યારે તે આરંભસ્થાનથી શરૂ કરી અંતિમ સ્થાનથી આગળના ઘટક સુધી આપેલી કિંમત વડે ભરી દે છે. ઉદાહરણ તરીકે, fill (list, 7) ઍરેના બધા ઘટકો કિંમત 7 વડે ભરવામાં આવે છે, જ્યારે fill(list, 2, 6, 5) ઍરેના ઘટકો list[2] થી list[6-1]માં કિંમત 5 ભરવામાં આવે છે. જુઓ આકૃતિ 9.6.

ArraysClassSortFill.java - SciTE

```java
class ArraysClassSortFill
{ // sort and fill methods on whole or partial array
    public static void main (String [] s)
    {
        double list[] = {  6.4, 8, 7.8, 9.8, 9.5,
                           6, 7, 8, 8.5, 5.9      };
        int indx;

        System.out.println ("Initial Elements:");
        display(list);
        java.util.Arrays.sort (list, 3, 9);  //sort partial array
        System.out.println ("\nsort partial array: list[3] to list[8]:");
        display(list);
        java.util.Arrays.sort (list);  //sort whole array
        System.out.println ("\nsort whole array:");
        display(list);

        java.util.Arrays.fill (list,7);  //fill whole array
        System.out.println ("\nFill whole array:");
        display(list);
        java.util.Arrays.fill (list, 2, 6, 5);  //fill partial array
        System.out.println ("\nFill partial array: list[2] to list[5]");
        display(list);
    } // end main

    static void display(double ary[])
    {
        for (int i=0; i<ary.length; i++)
        {
            System.out.print (ary[i] + "\t");
        }
        System.out.println();
    } // end display
} // end class
```

```
>javac ArraysClassSortFill.java
>Exit code: 0
>java -cp . ArraysClassSortFill
Initial Elements:
6.4  8.0  7.8  9.8  9.5  6.0  7.0  8.0  8.5  5.9

sort partial array: list[3] to list[8]:
6.4  8.0  7.8  6.0  7.0  8.0  8.5  9.5  9.8  5.9

sort whole array:
5.9  6.0  6.4  7.0  7.8  8.0  8.0  8.5  9.5  9.8

Fill whole array:
7.0  7.0  7.0  7.0  7.0  7.0  7.0  7.0  7.0  7.0

Fill partial array: list[2] to list[5]
7.0  7.0  5.0  5.0  5.0  5.0  7.0  7.0  7.0  7.0
>Exit code: 0
```

આકૃતિ 9.6 : ઍરેક્લાસની sort() અને fill() મૅથડનો ઉપયોગ

આપેલી કિંમતવાળો ઘટક એરેમાં શોધવા માટે એરે ક્લાસમાં binarySearch() મેથડ ઉપલબ્ધ છે. સુરેખશોધ (linear search) પદ્ધતિથી આપેલી કિંમતવાળો ઘટક એરેમાં શોધવા માટે આપણે આપણી પોતાની મેથડ લખીશું. સુરેખશોધ પદ્ધતિ ઍરેના એક પછી એક ઘટકોને આપેલી કિંમત સાથે ક્રમાનુસાર સરખાવે છે. આકૃતિ 9.7માં આપેલા કોડલિસ્ટિંગ અને તેના આઉટપુટનો અભ્યાસ કરો. આ પદ્ધતિ મુજબ જો એરેમાં આપેલી કિંમતવાળો ઘટક મળે, તો તેનું સૂચક સ્થાન (index position) પરત કરે છે, નહિ તો -1 કિંમત પરત કરે છે.

LinearSrch.java - SciTE

```java
// Search element in array using linear search
class LinearSrch
{
    public static void main (String [] s)
    {
        double list[] = {6, 5, 7, 9, 9.5, 6.5, 7.5, 8 };
        int indx;

        System.out.println ("Given Array elements are:");
        display(list);

        indx=search(list, 8);  // search 8
        if (indx < 0)
            System.out.println("\nelement 8 is not found in array");
        else
            System.out.println("\nelement 8 is found at position " + indx);
        indx=search(list,5.5);  // search 5.5
        if (indx < 0)
            System.out.println("\nelement 5.5 is not found in array");
        else
            System.out.println("\nelement 5.5 is found at position " + indx);
    } // end main
    static void display(double ary[])
    {
        for (int i=0; i<ary.length; i++)
        {
            System.out.println ( ary[i] );
        }
        System.out.println();
    } // end display
    static int search(double ary[], double x)
    {
        for (int i=0; i<ary.length; i++)
        {
            if (ary[i] == x ) return i;
        }
        return -1;
    } // end search
} // end class
```

```
>javac LinearSrch.java
>Exit code: 0
>java -cp . LinearSrch
Given Array elements are:
6.0
5.0
7.0
9.0
9.5
6.5
7.5
8.0

element 8 is found at position 7

element 5.5 is not found in array
>Exit code: 0
```

આકૃતિ 9.7 : એરેમાં ચોક્કસ કિંમતવાળો ઘટક શોધવો

2-D ઍરે (2-D Array)

દ્વિ-પરિમાણીય (2-D) ઍરે હાર અને સ્તંભ સ્વરૂપે કોષ્ટકીય માહિતીનો સંગ્રહ કરવા માટે વપરાય છે. ઉદાહરણ તરીકે, પાંચ વિદ્યાર્થીઓના ત્રણ કસોટીના ગુણ દર્શાવવા માટે આપણે આકૃતિ 9.8માં દર્શાવ્યા મુજબ પાંચ હાર અને ત્રણ સ્તંભવાળી કોષ્ટકીય ગોઠવણીનો ઉપયોગ કરી શકીએ. ગણિતમાં આપણે તેને પાંચ હાર અને ત્રણ સ્તંભવાળું શ્રેણિક (matrix) કહીએ છીએ, જેમાં દરેક ઘટકમાં ગુણ હોય છે.

Students	Test1	Test2	Test3
1	50	60	70
2	35	30	50
3	70	75	80
4	80	85	90
5	40	50	55

આકૃતિ 9.8 : 2-D ઍરેની કોષ્ટકીય રજૂઆત

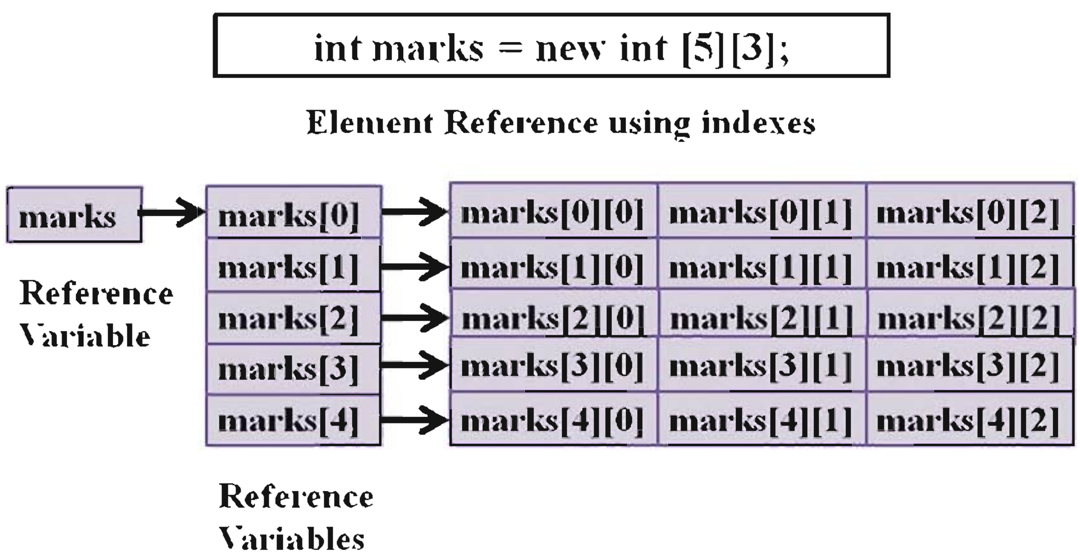
જાવામાં 2-D ઍરે ઘોષિત કરવા માટે ઍરેનું નામ અને મોટા કૌંસની બે જોડી [ ] [ ] કે જે અનુક્રમે પરિમાણો હાર (row) અને સ્તંભ (column)ના કદનો ઉલ્લેખ કરે છે, તેનો ઉપયોગ થાય છે. ઉદાહરણ તરીકે, સળંગ મેમરીમાં 5 x 3 = 15 પૂર્ણાંક કિંમતોનો સંગ્રહ કરવા માટે marks નામનો ઍરે ઘોષિત કરવા માટે તેમજ બનાવવા માટે નીચે આપેલું વિધાન વપરાય છે :

```
int marks [][] = new int [5][3];
```

અહીં, તાર્કિક દૃષ્ટિએ ઍરે એ 5 હાર અને 3 સ્તંભ સાથેનું એક કોષ્ટક છે. ભૌતિક દૃષ્ટિએ તે સળંગ મેમરીમાં સંગ્રહાયેલી 15 પૂર્ણાંક કિંમતો છે, જે 60 બાઇટ જગ્યા મેમરીમાં રોકે છે.

જાવા સીધી રીતે બહુ-પરિમાણીય ઍરે (multi-dimensional arrays)ની સગવડ પૂરી પાડતી નથી. 2-D ઍરે બનાવવા માટે આપણે ઍરેનો ઍરે (array of array) બનાવવો પડે. પરિમાણની સંખ્યા ઉપર કોઈ મર્યાદા નથી.

marks [5][3]માં એક 5 ઘટકોનો 1-D ઍરે-ઑબ્જેક્ટ બનાવે છે અને આકૃતિ 9.9માં દર્શાવ્યા પ્રમાણે આ દરેક ઘટક એ 3 પૂર્ણાંક સંખ્યાનો એક 1-D ઍરે-ઑબ્જેક્ટ છે.



આકૃતિ 9.9 : 2-D ઍરે એ એક 1-D ઍરેના ઍરે તરીકે

2-D એરેને ઘોષિત કરવાની અને પ્રારંભિક કિંમતો આપવાની રીત સંપૂર્ણપણે 1-D જેવી જ છે, સિવાય કે પરિમાણની સંખ્યા. 2-D એરેમાં દરેક હાર (row) એ 1-D એરે ઘટક તરીકે ગણાય છે. આથી, 1-D એરેની પ્રારંભિક કિંમતો આપવાની રીતે જ 2-D એરેની દરેક હારને પ્રારંભિક કિંમતો આપવા માટે છગડિયા કૌંસ { }ની અંદર અલ્પવિરામથી અલગ પાડેલી તેના ઘટકોની કિંમતો આપવામાં આવે છે. 2-D એરેને પ્રારંભિક કિંમતો આપવા માટે તેની દરેક પ્રારંભિક કિંમતો આપેલી હાર (initialized rows)ને અલ્પવિરામ (,)થી અલગ પાડીને છગડિયા કૌંસ { }માં રાખવામાં આવે છે. 1-D એરેની જેમ જ 2-D એરે પણ આકૃતિ 9.10ના કોડલિસ્ટિંગમાં દર્શાવ્યા પ્રમાણે વિવિધ રીતોથી ઘોષિત કરી શકાય છે. આકૃતિ 9.11 આ કોડનો આઉટપુટ આપ્યો છે.

```
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 Array2D.java *

/* creating and intializing 2-D array */
class Array2D
{
    public static void main (String [] s)
    {
        int marks1[] [ ];
        marks1 = new int [ 5] [3];
        int marks2[][] = new int [5][3];
        int [][] marks3 = new int [5][3];
        int marks4[] []= { {50, 60, 70}, {35, 30, 50},
                           {70, 75, 80}, {80, 85, 90},
                           {50, 50, 55}  };
        int [] []marks5 = {{50, 60, 70}, {35, 30, 50},
                           {70, 75, 80}, {80, 85, 90},
                           {50, 50, 55}  };

        System.out.print ("2-D Array marks1\n:");
        display(marks1,5,3);
        System.out.print ("2-D Array marks2:\n");
        display(marks2,5,3);
        System.out.print ("2-D Array marks3:\n");
        display(marks3,5,3);
        System.out.print ("2-D Array marks4:\n");
        display(marks4,5,3);
        System.out.print ("2-D Array marks5:\n");
        display(marks5,5,3);
    } // main
        static void display(int arr[][], int rows, int cols)
    {
        for (int i=0; i<rows; i++)
        {
            for (int j=0; j<cols; j++)
            {
                System.out.print (arr[i] [j] + "\t");
            }
            System.out.println();
        }
    } // display
} // class
```

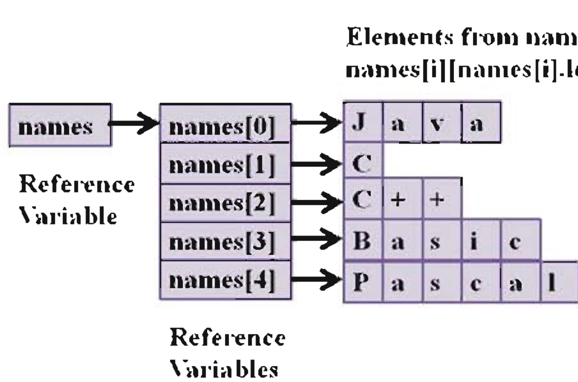
આકૃતિ 9.10 : 2-D એરેનું ઉદાહરણ

```
Array2D.java - SciTE
File  Edit  Search  View  Tools  Options  Language  Buffers  Help
1 Array2D.java
>javac Array2D.java
>Exit code: 0
>java -cp . Array2D
2-D Array marks1
:0    0    0
0     0    0
0     0    0
0     0    0
0     0    0
2-D Array marks2:
0     0    0
0     0    0
0     0    0
0     0    0
0     0    0
2-D Array marks3
0     0    0
0     0    0
0     0    0
0     0    0
0     0    0
2-D Array marks4:
50    60   70
35    30   50
70    75   80
80    85   90
50    50   55
2-D Array marks5.
50    60   70
35    30   50
70    75   80
80    85   90
50    50   55
>Exit code: 0
```

આકૃતિ 9.11 : આઉટપુટ

જાવામાં 2-D એરેને એરેનો એરે ગણવામાં આવે છે. આથી, 2-D એરેની હારમાં ગમે તેટલી સંખ્યામાં (પરિવર્તનશીલ - variable) સ્તંભ હોઈ શકે. આકૃતિ 9.12માં કમ્પ્યૂટર-પ્રોગ્રામિંગની પાંચ ભાષાઓનાં નામનો સંગ્રહ કરવા માટે અક્ષરોના 2-D એરેનો ઉપયોગ દર્શાવ્યો છે.

```
char names [ ][ ] = {   {'J', 'a', 'v', 'a'}, {'C'}, {'C', '+', '+'},
        {'B', 'a', 's', 'i', 'c'}, {'P', 'a', 's', 'c', 'a', 'l'}   }
```



આકૃતિ 9.12 : અલગ-અલગ સંખ્યામાં સ્તંભ સાથેનો 2-D એરે

દરેક નામ અલગ-અલગ હારમાં સંગ્રહવામાં આવ્યાં છે અને દરેક નામમાં અક્ષરોની સંખ્યા અલગ-અલગ છે. આ રીતે દરેક હારનું કદ અલગ-અલગ છે. દરેક હારનું કદ 1-D ઍરેની 'length' નામની પ્રૉપર્ટી વડે જાણી શકાય છે.

આકૃતિ 9.13માં અલગ-અલગ માપના સ્તંભવાળા 2-D ઍરેનો ઉપયોગ કેવી રીતે કરી શકાય તે દર્શાવ્યું છે. અહીં આપણે 1-D ઍરેના ઘટકોની સંખ્યા જાણવા માટે length પ્રૉપર્ટીનો ઉપયોગ કરેલો છે. 2-D ઍરે માટે તે તેની હારની સંખ્યા પરત કરે છે. 1-D ઍરે માટે તે આપેલી હારમાં સ્તંભની સંખ્યા પરત કરે છે. અહીં યાદ રાખો કે 2-D ઍરેમાં ઘટકોની સંખ્યા એ તેમાં રહેલી હારની સંખ્યા છે અને દરેક હાર એ 1-D ઍરે છે. ટૂંકમાં, length પ્રૉપર્ટી જ્યારે ફક્ત ઍરેના નામ સાથે વાપરવામાં આવે, ત્યારે તે તેના પ્રથમ પરિમાણનું કદ પરત કરે છે.

```
Array2Dchar.java - SciTE
File Edit Search View Tools Options Language Buffers Help
1 Array2Dchar.java

/* creating and intializing 2-D array */
class Array2Dchar
{
    public static void main (String [] s)
    {
        char names[] []= {
                          {'J','a','v','a'},
                          {'C'},
                          {'C', '+', '+'},
                          {'B', 'a', 's', 'i', 'c'},
                          {'P','a','s','c', 'a', 'l'}
                         };

        System.out.println("Number of elements in 2-D array: "
                          + names.length + "\n");
        System.out.print
                  ("Five names stored in 2-D Array of characters:\n");
        display(names,5);
    } // main
        static void display(char arr[][], int rows)
    {
        for (int i=0; i<rows; i++)
        {
            System.out.print ("Row " + i + " have " + arr[i].length +
                              " character elements: ");
            for (int j=0; j<arr[i].length; j++)
            {
                System.out.print ( arr[i] [j] );
            }
            System.out.println();
        }
    } // display
} // class
```

```
>javac Array2Dchar.java
>Exit code: 0
>java -cp . Array2Dchar
Number of elements in 2-D array: 5

Five names stored in 2-D Array of characters:
Row 0 have 4 character elements: Java
Row 1 have 1 character elements: C
Row 2 have 3 character elements: C++
Row 3 have 5 character elements: Basic
Row 4 have 6 character elements: Pascal
>Exit code: 0
```

આકૃતિ 9.13 : અલગ-અલગ સંખ્યામાં સ્તંભ ધરાવતા 2-D ઍરેનો પ્રોગ્રામમાં ઉપયોગ

હવે પછી આપણે જોઈશું કે અક્ષરોના 1-D ઍરેને જાવામાં ઉપલબ્ધ string ક્લાસના ઉપયોગથી વ્યાખ્યાયિત કરી શકાય છે. આકૃતિ 9.14માં આપણે નામનો સંગ્રહ કરવા માટે બાઇટનો 2-D ઍરે વાપર્યો છે. અહીં તે દર્શાવે છે કે char પ્રકારના ડેટાને બાઇટ પ્રકારમાં પરિવર્તિત કર્યા છે અને નામના અક્ષરોને તેની અનુરૂપ ASCII કિંમત આપવામાં આવી છે.

```
File Edit Search View Tools Options Language Buffers Help
1 Array2Dbyte.java

/* creating and intializing 2-D array */
class Array2Dbyte
{
    public static void main (String [] s)
    {
        byte names[] []= {
                          {'J','a','v','a'},
                          {67},
                          {'C', '+', '+'},
                          {'B', 'a', 's', 'i', 'c'},
                          {'P','a','s','c', 'a', 'l'}
                         };
        System.out.print ("Five names stored in 2-D Array of bytes:\n");
        display(names,5);
    } // main
        static void display( byte arr[][], int rows)
    {
        for (int i=0; i<rows; i++)
        {
            for (int j=0; j<arr[i].length; j++)
            {
                System.out.print ( arr[i] [j] + "\t" );
            }
            System.out.println();
        }
    } // display
} // class
```

```
>java -cp . Array2Dbyte
Five names stored in 2-D Array of bytes:
74    97    118   97
67
67    43    43
66    97    115   105   99
80    97    115   99    97    108
>Exit code: 0
```

આકૃતિ 9.14 : 2-D ઍરે : અક્ષરોને અનુરૂપ પૂર્ણાંક સંખ્યાનો ઉપયોગ કરી બાઇટમાં સંગ્રહ

યાદ રાખવાના મુદ્દાઓ :

- એરે એકસમાન પ્રકારના ડેટાનો સંગ્રહ છે.
- એરેના ઘટકોને તેના દરેક પરિમાણના ઇન્ડેક્સને કૌંસમાં [ ] આપીને વાપરી શકાય છે.
- ઇન્ડેક્સની કિંમત શૂન્યથી શરૂ થાય છે.
- બહુ-પરિમાણીય ઍરેમાં બીજા, ત્રીજા,...પરિમાણોનું કદ અલગ-અલગ હોઈ શકે છે.
- પરિમાણનું કદ જાણવા માટે 'length' ઍટ્રિબ્યૂટનો ઉપયોગ થાય છે.
- ઍરેને ઑબ્જેક્ટ તરીકે ગણવામાં આવે છે.
- ઍરેને તેની પ્રારંભિક કિંમતો આપ્યા વિના ઘોષિત કરવાથી ઍરે ઑબ્જેક્ટ બનતો નથી.

સ્ટ્રિંગ (Strings)

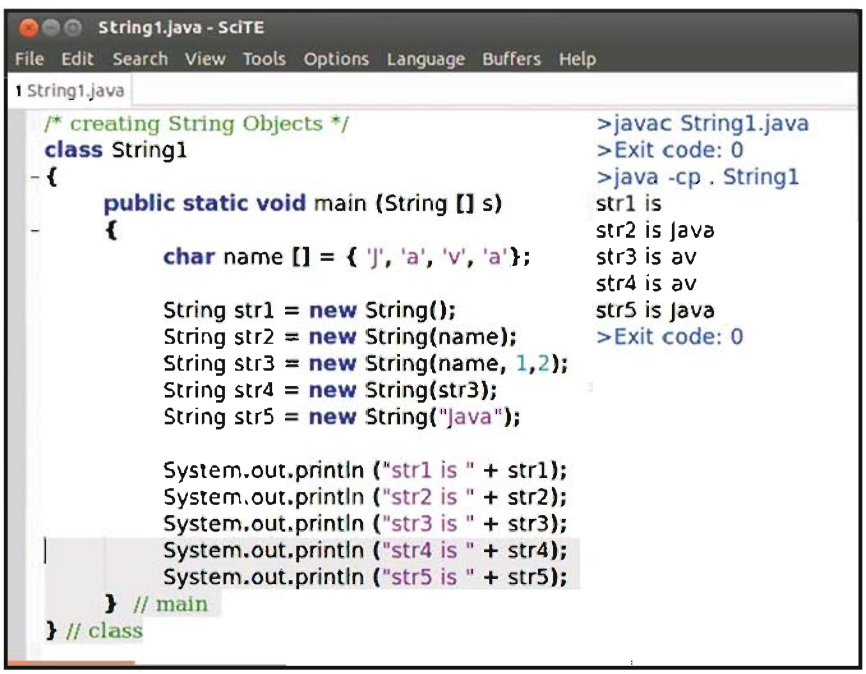
સ્ટ્રિંગ એ અક્ષરોની એક શ્રેણી સિવાય બીજું કંઈ જ નથી. આથી અક્ષરોના 1-D ઍરેને સ્ટ્રિંગ તરીકે ગણી શકાય. આપણે અગાઉ string લિટરલ વાપર્યા છે અને તેનો અભ્યાસ પણ કર્યો છે, કે જેમાં અક્ષરોની શ્રેણીને બેવડા અવતરણ-ચિહ્ન (double quotes) વચ્ચે લખવામાં આવે છે.

સ્ટ્રિંગનો સંગ્રહ કરી શકે તેવા ચલ વાપરવા માટે જાવામાં બે પ્રકારની સ્ટ્રિંગ આપેલી છે, જે 'String' અને 'StringBuffer' નામના બે ક્લાસ વડે નિયંત્રિત થાય છે. આ પ્રકરણમાં આપણે પ્રથમ પ્રકારની સ્ટ્રિંગનો અભ્યાસ કરીશું.

સ્ટ્રિંગ ઑબ્જેક્ટ બનાવવા માટે વપરાતાં કેટલાંક કન્સ્ટ્રક્ટર (constructor) નીચે મુજબ છે :

- String () – ચલ રહિત String () એક પણ અક્ષર વગરનો સ્ટ્રિંગ ઑબ્જેક્ટ બનાવે છે.
- String (char ary[]) – ary ચલનો પ્રારંભિક કિંમત તરીકે ઉપયોગ કરીને સ્ટ્રિંગ ઑબ્જેક્ટ બનાવે છે.
- String (char ary[], int start, int len) – આપેલા પ્રથમ ચલ aryના start સ્થાનથી len જેટલા ઘટકોનો સ્ટ્રિંગ ઑબ્જેક્ટ બનાવે છે.
- String (String strObj) – ચલ તરીકે આપેલા ઑબ્જેક્ટ જેવો જ સ્ટ્રિંગ ઑબ્જેક્ટ બનાવે છે.
- String (string literal) – ચલ તરીકે આપેલા string literalને નિર્દેશ કરતો સ્ટ્રિંગ ઑબ્જેક્ટ બનાવે છે.

આકૃતિ 9.15માં વિવિધ પ્રકારના String ક્લાસના કન્સ્ટ્રક્ટરનો ઉપયોગ દર્શાવેલ છે. ઑબ્જેક્ટ બનાવતા સમયે new પ્રક્રિયકનો ઉપયોગ ભૂલતા નહીં.




```
/* creating String Objects */
class String1
{
    public static void main (String [] s)
    {
        char name [] = { 'J', 'a', 'v', 'a'};

        String str1 = new String();
        String str2 = new String(name);
        String str3 = new String(name, 1,2);
        String str4 = new String(str3);
        String str5 = new String("Java");

        System.out.println ("str1 is " + str1);
        System.out.println ("str2 is " + str2);
        System.out.println ("str3 is " + str3);
        System.out.println ("str4 is " + str4);
        System.out.println ("str5 is " + str5);
    } // main
} // class
```

```
>javac String1.java
>Exit code: 0
>java -cp . String1
str1 is
str2 is Java
str3 is av
str4 is av
str5 is Java
>Exit code: 0
```

આકૃતિ 9.15 : વિવિધ કન્સ્ટ્રક્ટરના ઉપયોગ વડે string ક્લાસના ઑબ્જેક્ટ બનાવવા

જાવામાં અક્ષરનો સંગ્રહ કરવા માટે અક્ષર દીઠ બે બાઈટ વપરાય છે. જગ્યા બચાવવા માટે જો ASCII અક્ષરો હોય, તો char પ્રકારના ડેટા-ઍરે વાપરવાને બદલે bytes પ્રકારના ડેટા-ઍરેનો ઉપયોગ કરવો જોઈએ. આ માટે આપણે કન્સ્ટ્રક્ટરમાં ચલ તરીકે byteનો ઍરે વાપરી શકીએ. આકૃતિ 9.15માં આપેલા પ્રોગ્રામના ઍરે char name[ ]ને બદલી byte name[ ] કરો અને પ્રોગ્રામનો અમલ કરવા કોશિશ કરો.

નોંધ : લિટરલ (literals)નો મેમરીમાં સંગ્રહ કરવામાં આવે છે. જ્યારે બે સ્ટ્રિંગ-ઑબ્જેક્ટ એકસરખા સ્ટ્રિંગ લિટરલ વાપરીને બનાવ્યા હોય, ત્યારે બીજા ઑબ્જેક્ટ માટે મેમરીમાં જગ્યા ફાળવાતી નથી. બંને ઑબ્જેક્ટ એક જ મેમરી સ્થાનનો ઉલ્લેખ કરે છે.

જ્યારે સ્ટ્રિંગ ઑબ્જેક્ટ new પ્રક્રિયક વાપરીને બનાવવામાં આવે છે, ત્યારે બંને સ્ટ્રિંગ એકસમાન હોય, તોપણ મેમરીમાં અલગ જગ્યા ફાળવવામાં આવે છે. આકૃતિ 9.16માં આપેલું કોડલિસ્ટિંગ આ પ્રકારનું ઉદાહરણ છે.

String2.java - SciTE

File Edit Search View Tools Options Language Buffers Help

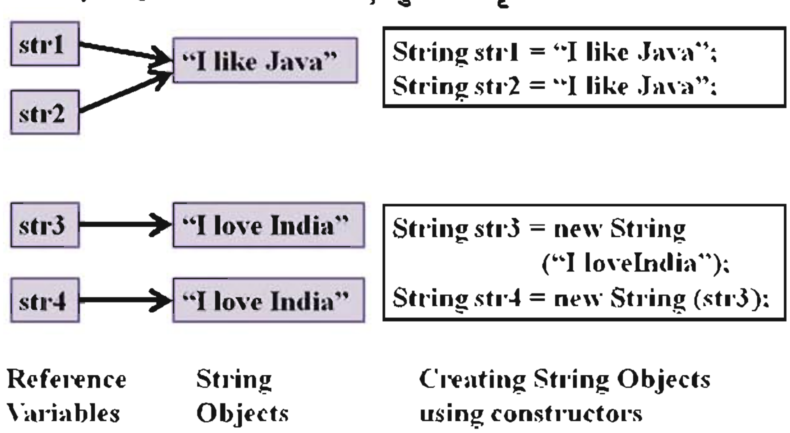
1 String2.java

```
/* String Objects */
class String2
{
    public static void main (String [] s)
    {
        String str1 = "I like Java";
        String str2 = "I like Java";
        String str3 = new String("I love India");
        String str4 = new String(str3);

        System.out.println ("str1==str2: "
                        + (str1==str2));
        System.out.println ("str3==str4: "
                        + (str3==str4));

        System.out.println ("str1: " + str1);
        System.out.println ("str2: " + str2);
        System.out.println ("str3: " + str3);
        System.out.println ("str4: " + str4);
    } // main
} // class
```

```
>javac String2.java
>Exit code: 0
>java -cp . String2
str1==str2: true
str3==str4: false
str1: I like Java
str2: I like Java
str3: I love India
str4: I love India
>Exit code: 0
```

આકૃતિ 9.16 : સ્ટ્રિંગ લિટરલ અને new પ્રક્રિયક વાપરીને એકસરખા સ્ટ્રિંગ ઑબ્જેક્ટ બનાવવા

આકૃતિ 9.16માં આપેલા કોડલિસ્ટિંગમાં "str1 == str2" એ str1 અને str2 એમ બે રેફરન્સ ચલમાં સંગ્રહ કરેલી વિગતની તુલના કરે છે અને નહિ કે str1 અને str2 વડે નિર્દેશ કરેલા ઑબ્જેક્ટની. આ બાબતની નોંધ કરો. અહીં str1 અને str2 ઑબ્જેક્ટ new પ્રક્રિયક વિના અને એકસરખા સ્ટ્રિંગ લિટરલ વાપરીને બનાવ્યા છે. આથી બંને ચલ str1 એ બનાવેલા ઇન્સ્ટન્સનો જ નિર્દેશ કરે છે, આકૃતિ 9.17માં દર્શાવ્યા પ્રમાણે str2 ઑબ્જેક્ટ માટે અલગ મેમરીની ફાળવણી કરવામાં આવી નથી. આ જ પ્રોગ્રામમાં str3 અને str4 સ્ટ્રિંગ ઑબ્જેક્ટ new પ્રક્રિયકનો ઉપયોગ કરીને બનાવ્યા છે. બંને ઑબ્જેક્ટમાંની માહિતી સરખી છે, પરંતુ બંને અલગ-અલગ મેમરીસ્થાનનો સંદર્ભ ધરાવે છે. આ બંને ઑબ્જેક્ટ માટે અલગ-અલગ મેમરીની ફાળવણી કરવામાં આવી છે, જુઓ આકૃતિ 9.17.



આકૃતિ 9.17 : એકસમાન સ્ટ્રિંગ

સ્ટ્રિંગ ક્લાસમેથડ (String Class Methods)

બે સ્ટ્રિંગની તુલના કરવી, સ્ટ્રિંગની લંબાઈ મેળવવી, બે સ્ટ્રિંગને જોડવી, સ્ટ્રિંગમાંથી આંશિક સ્ટ્રિંગ (સબસ્ટ્રિંગ) મેળવવી, સ્ટ્રિંગને પરાવર્તિત કરવી, સ્ટ્રિંગનું વિભાજન કરવું, અક્ષર અથવા અક્ષરસમૂહને સ્ટ્રિંગમાં શોધવા વગેરે કાર્યો માટે સ્ટ્રિંગ-ક્લાસ કેટલીક મેથડ પૂરી પાડે છે. આપણે પ્રોગ્રામ દ્વારા કેટલીક મેથડનો ઉપયોગ કરવાના ઉદાહરણ જોઈશું. આકૃતિ 9.18માં લખેલો પ્રોગ્રામ સ્ટ્રિંગ અથવા સબસ્ટ્રિંગની સરખામણી કરવા માટેની વિવિધ રીત દર્શાવે છે.

આકૃતિ 9.18 : સ્ટ્રિંગ ક્લાસની મેથડનો ઉપયોગ કરી સ્ટ્રિંગની તુલના

કોષ્ટક 9.1માં સ્ટ્રિંગની તુલના કરવા માટેની વિવિધ મેથડનું વર્ણન અને વાક્યરચના દર્શાવી છે.

મેથડ	વર્ણન
boolean equals (String str)	મેથડકૉલ કરતી સ્ટ્રિંગ અને પ્રાચલ str (ઑબ્જેક્ટ) સરખાં હોય, તો true પરત કરે છે.
boolean equalsIgnoreCase (String str)	મેથડકૉલ કરતી સ્ટ્રિંગ અને પ્રાચલ str ઑબ્જેક્ટના અક્ષરોના કેસ (case) (કૅપિટલ અને બીજી એબીસીડી)ની અવગણના કરીને તુલના કરે છે. જો બંને સરખાં હોય, તો true પરત કરે છે. (case insensitive)
int compareTo (String str)	જો મેથડકૉલ કરનાર સ્ટ્રિંગ ઑબ્જેક્ટ પ્રાચલ strની તુલનાએ સમાન હોય, મોટી હોય અથવા નાની હોય, તો અનુક્રમે 0, >0, <0 પૂર્ણાંક સંખ્યા પરત કરે છે.
int compareToIgnoreCase (String str)	CompareTo મેથડ પ્રમાણે જ પરંતુ case પ્રત્યે અસંવેદનશીલ

કોષ્ટક 9.1 : સ્ટ્રિંગની તુલના માટે સ્ટ્રિંગ ક્લાસની વિવિધ મેથડ

સ્ટ્રિંગ ક્લાસ નીચે જણાવેલાં અન્ય કાર્યો કરવા માટે અનેક મેથડ પૂરી પાડે છે. આમાંની કેટલીક કોષ્ટક 9.2માં આપેલી છે.

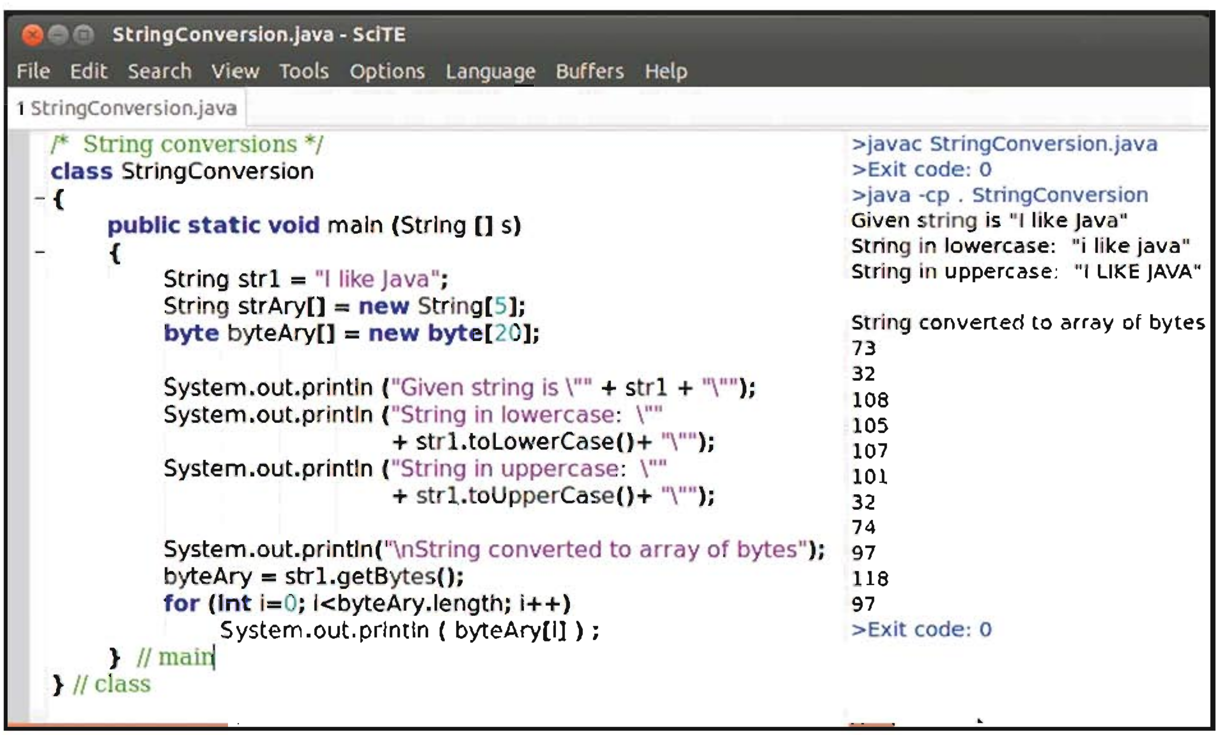
- સ્ટ્રિંગનો અમુક ભાગ અલગ કરવો.
- અક્ષરો કે શબ્દસમૂહ (સબસ્ટ્રિંગ) બદલવા.
- સ્ટ્રિંગનું સબસ્ટ્રિંગમાં વિભાજન કરવું.
- અક્ષરોની સંખ્યા મેળવવી.
- આપેલા ઈન્ડેક્સસ્થાન પરનો અક્ષર મેળવવો.

- સ્ટ્રિંગને bytes પ્રકારના એરેમાં પરિવર્તિત કરવી.
- સ્ટ્રિંગના અક્ષરોને સ્મૉલ અથવા કેપિટલ કરવા.
- સ્ટ્રિંગના અંતમાં બીજી સ્ટ્રિંગ ઉમેરવી.
- સ્ટ્રિંગ કે તેના ભાગની નકલ કરવી.

મેથડ	વર્ણન
int length()	મેથડકૉલ કરતી સ્ટ્રિંગ ઑબ્જેક્ટમાં રહેલા અક્ષરોની સંખ્યા પરત કરે છે.
char indexAt (int index)	મેથડકૉલ કરતી સ્ટ્રિંગ ઑબ્જેક્ટમાં પ્રાચલ ઇન્ડેક્સ સ્થાને આવેલા અક્ષરને પરત કરે છે. ઇન્ડેક્સ શૂન્યથી ગણાય છે.
byte [] getBytes()	મેથડકૉલ કરતી સ્ટ્રિંગના અક્ષરોને byte એરેમાં પરત કરે છે.
void getChars (int fromIndx, int toIndx, char target [], int targetIndx)	મેથડકૉલ કરતી સ્ટ્રિંગ ઑબ્જેક્ટના fromIndx સ્થાનથી toIndx-1 સ્થાન સુધીના અક્ષરોની લક્ષ્ય એરેમાં targetIndx સ્થાન પછી નકલ કરે છે.
String concat(String str)	મેથડકૉલ કરતી સ્ટ્રિંગના અંતમાં પ્રાચલ strના અક્ષરોને ઉમેરીને સ્ટ્રિંગ ઑબ્જેક્ટ પરત કરે છે.
String toLowerCase()	મેથડકૉલ કરતી સ્ટ્રિંગના બધા જ અક્ષરોને સ્મૉલ અક્ષરોમાં બનાવીને સ્ટ્રિંગ પરત કરે છે.
String toUpperCase()	મેથડકૉલ કરતી સ્ટ્રિંગના બધા જ અક્ષરોને કેપિટલ અક્ષરમાં પરિવર્તિત કરીને સ્ટ્રિંગ પરત કરે છે.

કોષ્ટક 9.2 : સ્ટ્રિંગક્લાસની અન્ય મેથડ

આકૃતિ 9.19માં આપેલાં કોડલિસ્ટિંગમાં આમાંની કેટલીક મેથડનો ઉપયોગ દર્શાવ્યો છે.




```
/* String conversions */
class StringConversion
{
    public static void main (String [] s)
    {
        String str1 = "I like Java";
        String strAry[] = new String[5];
        byte byteAry[] = new byte[20];

        System.out.println ("Given string is \"" + str1 + "\"");
        System.out.println ("String in lowercase:  \""
                             + str1.toLowerCase()+ "\"");
        System.out.println ("String in uppercase:  \""
                             + str1.toUpperCase()+ "\"");

        System.out.println("\nString converted to array of bytes");
        byteAry = str1.getBytes();
        for (int i=0; i<byteAry.length; i++)
            System.out.println ( byteAry[i] ) ;
    } // main
} // class
```

```
>javac StringConversion.java
>Exit code: 0
>java -cp . StringConversion
Given string is "I like Java"
String in lowercase:  "i like java"
String in uppercase:  "I LIKE JAVA"

String converted to array of bytes
73
32
108
105
107
101
32
74
97
118
97
>Exit code: 0
```

આકૃતિ 9.19 : સ્ટ્રિંગને પરિવર્તિત કરતી મેથડનો ઉપયોગ

નોંધ : એરે ચલ માટે length એ એટ્રિબ્યૂટ કે પ્રોપર્ટી છે, જ્યારે સ્ટ્રિંગ ઑબ્જેક્ટ માટે length એ મેથડ છે. આથી સ્ટ્રિંગ ઑબ્જેક્ટ સાથે length મેથડ વાપરતા સમયે () કૌંસનો ઉપયોગ કરવો જરૂરી છે.

જાવામાં સ્ટ્રિંગ ક્લાસમાં સ્ટ્રિંગને ઉલટાવવા માટે કોઈ મેથડ ઉપલબ્ધ નથી. આથી, આપણે સ્ટ્રિંગને ઉલટાવવા માટે પ્રોગ્રામ લખીએ. આકૃતિ 9.20માં આપેલા કોડલિસ્ટિંગ અને તેના આઉટપુટનો અભ્યાસ કરો.

StringReverse.java - SciTE

File Edit Search View Tools Options Language Buffers Help

1 StringReverse.java

```
/* Reverse String */
class StringReverse
{
    public static void main (String [] s)
    {
        String str = new String("I Like Java");
        System.out.println ("Given   String: " + str);
        System.out.println ("Reverse String: " + reverseStr(str));
    } // end main

    static String reverseStr (String str)  // reverse string
    {
        int len = str.length();
        byte byteAry[] = new byte[len];
        byteAry = str.getBytes();        // convert to byte array
        for (int i=0, j=len-1; i<len/2; i++, j--)  // exch half array
        {
            byte tmp = byteAry[i];
            byteAry[i] = byteAry[j] ;
            byteAry[j]=tmp;
        }
        return new String(byteAry);   // used constructor
    }
} // class
```

```
>javac StringReverse.java
>Exit code: 0
>java -cp . StringReverse
Given   String: I Like Java
Reverse String: avaJ ekiL I
>Exit code: 0
```

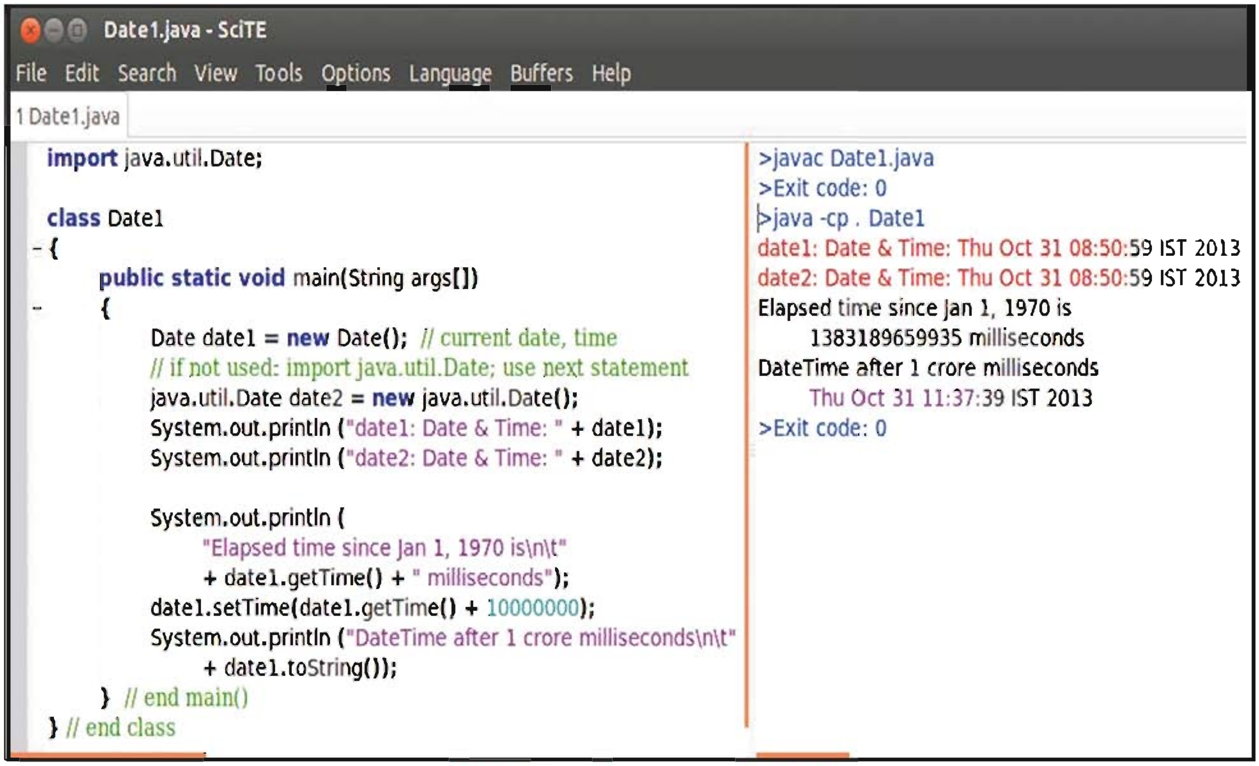
આકૃતિ 9.20 : સ્ટ્રિંગને ઉલટાવવી

અહીં, આપણે વપરાશકર્તા વ્યાખ્યાયિત reverseStr નામની મેથડ લખી છે, જે નીચે જણાવેલાં પગલાં પ્રમાણે સ્ટ્રિંગને ઉલટાવે છે :

1. સ્ટ્રિંગક્લાસની length() મેથડ વાપરીને સ્ટ્રિંગમાં રહેલા અક્ષરોની સંખ્યા (સ્ટ્રિંગની લંબાઈ) નક્કી કરો.
2. સ્ટ્રિંગક્લાસની getBytes() મેથડના ઉપયોગથી સ્ટ્રિંગને byte એરેમાં પરિવર્તિત કરો.
3. byte એરેમાં ડાબી બાજુના અડધા ઘટકોની (ડાબી બાજુથી જમણી બાજુ તરફ) જમણી બાજુના અડધા ઘટકો (છેલ્લે જમણી બાજુથી ડાબી બાજુ તરફ) સાથે અદલાબદલી કરો.
4. કન્સ્ટ્રક્ટર વાપરીને byte એરેમાંથી સ્ટ્રિંગ ઑબ્જેક્ટ બનાવો અને તેને પરત કરો.

Date ક્લાસ (Date Class)

આપણે જાવામાં ઉપલબ્ધ String ક્લાસ અને Arrays ક્લાસનો અભ્યાસ કર્યો. જાવાલાઇબ્રેરીમાં java.util નામના પૅકેજમાં Date ક્લાસ પૂરો પાડે છે. Date ક્લાસ તારીખ અને સમય બંનેનો સમાવેશ કરે છે અને મિલિસેકન્ડ સુધીની ચોકસાઈ સહિત કિંમત જણાવે છે. આકૃતિ 9.21માં કોડલિસ્ટિંગ અને તેના આઉટપુટ સાથે Date ક્લાસનો ઉપયોગ દર્શાવ્યો છે.



આકૃતિ 9.21 : Date ક્લાસનો ઉપયોગ

કોષ્ટક 9.3માં Date ક્લાસની કેટલીક મેથડની યાદી આપેલી છે.

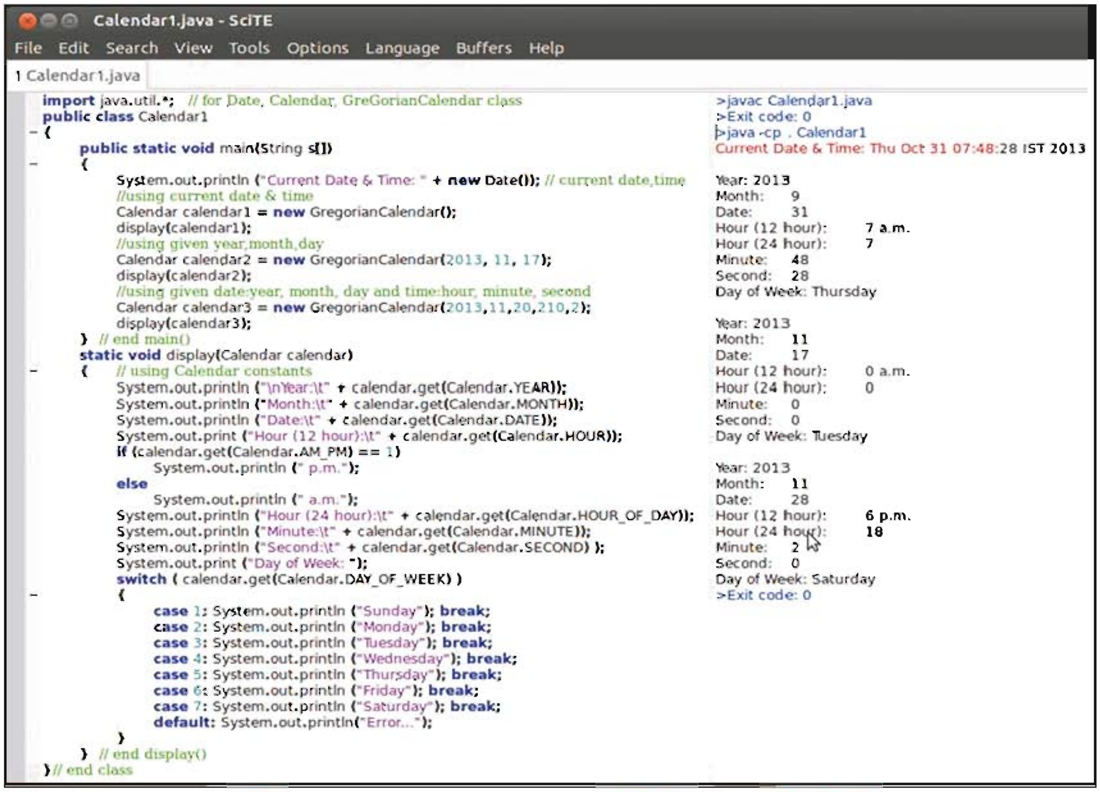
મેથડ	વર્ણન
Date()	સિસ્ટમના તત્કાલીન સમયનો ઉપયોગ કરીને Date ઑબ્જેક્ટ બનાવે છે.
Date (long elapsedTime)	જાન્યુઆરી, 1, 1970 GMTથી પ્રાચલમાં આપેલા સમય વચ્ચેનો સમયગાળો મિલિસેકન્ડમાં ગણીને Date ઑબ્જેક્ટ બનાવે છે.
String toString()	Date ઑબ્જેક્ટના તારીખ અને સમયને સ્ટ્રિંગમાં રજૂ કરી તે સ્ટ્રિંગ પરત કરે છે.
long getTime()	જાન્યુઆરી 1, 1970 GMTથી અત્યાર સુધીની મિલિસેકન્ડ પરત કરે છે.
void setTime (long elapsedTime)	પ્રાચલમાં આપેલા વિતેલા સમયનો ઉપયોગ કરી નવી તારીખ અને સમય સેટ કરે છે.

કોષ્ટક 9.3 : Date ક્લાસની વિવિધ મેથડ

Calendar ક્લાસ (Calendar Class)

Date ક્લાસની જેમ Calendar ક્લાસ પણ java.util પેકેજમાં આપવામાં આવ્યો છે. આ ક્લાસ વર્ષ, માસ, તારીખ, કલાક, મિનિટ અને સેકન્ડ જેવી કૅલેન્ડરની વિગતવાર માહિતી મેળવવા માટે વાપરી શકાય છે. અહીં, આપણે calendar ક્લાસના સબક્લાસ GregorianCalendarના ઉપયોગ બાબત જોઈશું.

આકૃતિ 9.22માં ઉદાહરણ રૂપે Calendar ક્લાસનો ઉપયોગ કરતો પ્રોગ્રામ અને તેનું આઉટપુટ દર્શાવ્યું છે. અહીં તારીખ અને સમયના વિવિધ ઘટક પ્રદર્શિત કરવા માટે વપરાશકર્તાએ વ્યાખ્યાયિત display() ક્લાસમેથડનો ઉપયોગ કર્યો છે, તેની નોંધ કરો.



આકૃતિ 9.22 : Calendar ક્લાસ અને તેના અચલનો ઉપયોગ

get મેથડની જેમ set મેથડ વાપરીને Calendar ક્લાસના ક્ષેત્ર અચલ (field constants)ની કિંમત આપણે સેટ કરી શકીએ છીએ. ઉદાહરણ તરીકે, જો Calendar ક્લાસનો calendar ઑબ્જેક્ટ હોય, તો 'calendar.set(Calendar.DATE, 20)' મેથડકૉલનો અમલ કરતાં મહિનાની તારીખ 20 સેટ થશે.

કોષ્ટક 9.4માં Calendar ક્લાસમાં વ્યાખ્યાયિત પૂર્ણાંક સંખ્યાવાળા અચલોની યાદી આપેલી છે. આ અચલોને અર્થપૂર્ણ અને સ્વયંસ્પષ્ટ નામો આપવામાં આવેલાં છે :

અચલ	વર્ણન
YEAR	કૅલેન્ડરનું વર્ષ
MONTH	કૅલેન્ડરનો મહિનો (જાન્યુઆરી માટે 0 અને ડિસેમ્બર માટે 11)
DATE	મહિનાનો દિવસ
DAY_OF_MONTH	મહિનાનો દિવસ (DATE સમાન)
HOUR	12 કલાકની સંકેતલિપિ પ્રમાણે કલાક
HOUR_OF_DAY	24 કલાકની સંકેતલિપિ પ્રમાણે કલાક
MINUTE	મિનિટ
SECOND	સૅકન્ડ
AM_PM	AM માટે 0 અને PM માટે 1
DAY_OF_WEEK	અઠવાડિયામાં દિવસનો ક્રમ (રવિવાર માટે 1 અને શનિવાર માટે 7)
WEEK_OF_MONTH	મહિનામાં અઠવાડિયાનો ક્રમ
WEEK_OF_YEAR	વર્ષમાં અઠવાડિયાનો ક્રમ
DAY_OF_YEAR	વર્ષમાં દિવસનો ક્રમ (પ્રથમ દિવસ માટે 1)

કોષ્ટક 9.4 : Calendar ક્લાસમાં વ્યાખ્યાયિત અચલ

સારાંશ

આ પ્રકરણ ઍરે, સ્ટ્રિંગ અને ડેઇટનો ઉપયોગ કેવી રીતે કરવો, તે સમજાવે છે. એકસમાન પ્રકારના ચલોના સમૂહ માટે ઍરેનો ઉપયોગ થાય છે. જાવા મૂળભૂત રીતે ફક્ત 1-D ઍરે પૂરા પાડે છે. 1-D ઍરેનો ઍરે વાપરીને આપણે વ્યાવહારિક રીતે બહુ-પરિમાણીય ઍરે મેળવી શકીએ છીએ. ઍરેને Arrays ક્લાસના ઑબ્જેક્ટ તરીકે ગણવામાં આવે છે, તેથી ઍરેનું નામ એ સંદર્ભચલ (reference variable) છે, જે તેના ઘટકો જે જગ્યાએ સંગૃહીત કરવામાં આવ્યા છે તે મેમરી સ્થાનાંકનો ઉલ્લેખ કરે છે. ઍરે-ઑબ્જેક્ટ સાથે Arrays ક્લાસની તમામ મેથડ વાપરી શકાય છે. આવી મેથડનાં ઉદાહરણો છે : sort, fill. જાવામાં આપેલા String ક્લાસ અક્ષરોની શ્રેણી સાથે કામ કરવાનું સામર્થ્ય પૂરું પાડે છે. સ્ટ્રિંગ ઉપર કરી શકાતાં કાર્યોનાં ઉદાહરણો : સ્ટ્રિંગની સરખામણી, સ્ટ્રિંગમાં રહેલા અક્ષરોની સંખ્યા શોધવી, સ્ટ્રિંગમાં રહેલા અક્ષરોના કેસને રૂપાંતરિત કરવા. તારીખ અને સમય સાથે કામ કરવા માટે Date અને Calendar ક્લાસ આપવામાં આવ્યા છે. Calendar ક્લાસની મેથડ વાપરીને આપણે વર્ષ, માસ, તારીખ, કલાક, મિનિટ અને સેકન્ડ જેવા ઘટકોની કિંમત મેળવી શકીએ છીએ, તેમજ સેટ કરી શકીએ છીએ.

સ્વાધ્યાય

1. ઍરે વિશે ઉદાહરણ આપી સમજાવો.

2. 1-D અને 2-D ઍરે વચ્ચેનો તફાવત જણાવો.

3. નીચે જણાવેલા ક્લાસનો ઉપયોગ સમજાવો :

   a. String b. Date c. Calendar

4. **નીચે આપેલા વિકલ્પમાંથી સાચો વિકલ્પ જણાવો :**

   (1) નીચેનામાંથી શું ઍરેની પ્રારંભિક ઇન્ડેક્સ કિંમતનો ઉલ્લેખ કરે છે ?

   (a) 0 (b) 1 (c) null (d) ઉપરના તમામ વિકલ્પ

   (2) sales[5][12] ઍરેમાં દ્વિતીય પરિમાણનું કદ કેટલું છે ?

   (a) 5 (b) 12 (c) 60 (d) 10

   (3) sales[5][12] ઍરે માટે sales.length પદાવલી શું પરત કરશે ?

   (a) 5 (b) 12 (c) 60 (d) 120

   (4) જો પ્રારંભિક કિંમતો આપ્યા વિના sales [5][12] ઍરે વ્યાખ્યાયિત કરવામાં આવે, તો sales[0][0]ની શરૂઆતની કિંમત શું હશે ?

   (a) 0 (b) પૂર્વનિર્ધારિત કિંમત (c) કમ્પાઇલેશન એરર (d) 60

   (5) String ક્લાસના ઑબ્જેક્ટમાં 'length' શું નિર્દેશ કરે છે ?

   (a) એટ્રિબ્યૂટ (b) મેથડ (c) ક્લાસ વેરિયેબલ (d) ક્લાસનું નામ

   (6) જો 'str' એ String ક્લાસનો ઑબ્જેક્ટ હોય અને તેમાં "Thank GOD" માહિતી હોય, તો str.length()ની કિંમત કેટલી હશે ?

   (a) 9 (b) 10 (c) 8 (d) 11

(7) જો આપણે Calendar ક્લાસની get મેથડમાં DAY_OF_WEEK પ્રાચલ તરીકે વાપરીએ, તો ક્યા પ્રકારની કિંમત પરત થશે ?

(a) int (b) char (c) String (d) boolean

પ્રાયોગિક સ્વાધ્યાય

1. એક પ્રોગ્રામ લખો, જે દિવસના 6 am થી 6 pm સુધીના દરેક કલાકના તાપમાનની 12 કિંમતોનો સંગ્રહ કરવા માટે ઍરેનો ઉપયોગ કરે. આ કિંમતો આપવા માટે પ્રારંભિક કિંમતો અથવા એસાઇન્મેન્ટ વિધાનોનો ઉપયોગ કરો. દિવસનું મહત્તમ અને ન્યૂનતમ તાપમાન છાપો.

2. એક પ્રોગ્રામ લખો, જે પાંચ વેચનાર વ્યક્તિઓના અઠવાડિક વેચાણની માહિતીનો સંગ્રહ કરે. વેચનાર વ્યક્તિને તેના અઠવાડિક વેચાણના સરેરાશના પ્રમાણે અઠવાડિક પ્રોત્સાહન રકમ આપવામાં આવે છે. જો સરેરાશ અઠવાડિક વેચાણ ₹ 10,000 સુધી હોય, તો 10 %, સરેરાશ અઠવાડિક વેચાણ ₹ 10000થી ₹ 30,000 સુધીમાં હોય, તો 15 % અને ₹ 30,000થી વધુ હોય, તો 20 % પ્રોત્સાહન રકમ છે. બધી વેચનાર વ્યક્તિઓનું દરરોજનું કુલ વેચાણ શોધો. આ ઉપરાંત વેચનાર વ્યક્તિની અઠવાડિક પ્રોત્સાહન રકમની ગણતરી કરો.

3. એક પ્રોગ્રામ લખો, જે આપેલી સ્ટ્રિંગ palindrome છે કે નહીં તે જણાવે. (સ્ટ્રિંગ palindrome કહેવાય, જો તેને ઉલટાવવાથી પણ સરખી રહે, જેમકે "1771" અને "madam".)

4. આજની તારીખને "DD-MMM-YYYY" સ્વરૂપમાં છાપવાનો પ્રોગ્રામ લખો. ઉદાહરણ તારીખ, 17th November 2013ને 17-Nov-2013 તરીકે છાપો. આ ઉપરાંત અઠવાડિયાનો દિવસ શબ્દમાં છાપો.

5. તમારી જન્મતારીખ અને સમય પ્રમાણે તારીખ અને સમય સેટ કરવાનો પ્રોગ્રામ લખો. તમારા જન્મદિવસે અઠવાડિયાનો ક્યો દિવસ હતો તે છાપો.

જાવામાં અપવાદરૂપ પરિસ્થિતિનું વ્યવસ્થાપન 10

સામાન્ય રીતે આપણે એવું માનીએ છીએ કે, કંપાઇલ કરેલો પ્રોગ્રામ એ ક્ષતિરહિત હોય છે અને તેથી સફળતાપૂર્વક તેનો અમલ થઈ શકે છે. પરંતુ એકાદ કિસ્સામાં એવું પણ બને કે આવો ક્ષતિરહિત પ્રોગ્રામ પણ અમલ દરમિયાન અધવચ્ચે અટકી પડે. ઉદાહરણ તરીકે, જો આપણે એવો પ્રોગ્રામ લખ્યો હોય કે જે કોઈ ચોક્કસ વેબસાઇટ સાથે જોડાણ કરીને વેબપેજને ડાઉનલોડ કરે, તો સામાન્ય સંજોગોમાં પ્રોગ્રામ અપેક્ષા મુજબ ચાલશે, પરંતુ ધારો કે, આ પ્રોગ્રામ એવા કમ્પ્યૂટર પર ચલાવવામાં આવે કે જેને ઇન્ટરનેટ સાથે જોડાણ ન હોય, તો પ્રોગ્રામ અનઅપેક્ષિત પરિણામો આપે તેવું બને. જ્યારે કોઈ પ્રોગ્રામમાં આપણે ફાઇલ સાથે કામ કરતાં હોઈએ, ત્યારે પણ આવું જ બનશે. ઉદાહરણ તરીકે, ધારો કે કોઈ પ્રોગ્રામ માત્ર વાંચી શકાય તેવી (Read Only) ફાઇલમાં સુધારા-વધારા કરવા પ્રયત્ન કરતો હોય અથવા એવી ફાઇલને ખોલવાનો પ્રયત્ન કરતો હોય કે જે કમ્પ્યૂટરમાં ઉપલબ્ધ જ ન હોય. આ પ્રકારના કિસ્સાઓને જાવામાં "અપવાદરૂપ પરિસ્થિતિ" (Exceptions) તરીકે ઓળખવામાં આવે છે. અપવાદરૂપ પરિસ્થિતિઓને પરિણામે પ્રોગ્રામ અપેક્ષિત ધોરણો અનુસાર ચાલતો નથી અને કદાચ પ્રોગ્રામ અધવચ્ચેથી અટકી પણ શકે છે.

અપવાદ એ પ્રોગ્રામના અમલ દરમિયાન ઉદ્‌ભવતી સમસ્યાનો સંકેત છે, જે મોટે ભાગે ભૂલસંદેશ દર્શાવે છે. જોકે, અપવાદ જવલ્લે જ ઉદ્‌ભવતા હોય છે, તેમ છતાં, પ્રોગ્રામ લખતી વખતે આવા અપવાદરૂપ પરિસ્થિતિઓનું નિયમન થઈ શકે તે માટે જરૂરી કાળજી લેવી જ જોઈએ. અપવાદનું નિયમન એ એવી આગોતરી વ્યવસ્થા છે કે જે, જાણે કે કોઈ સમસ્યા સર્જાઈ જ નથી તેમ માની પ્રોગ્રામનો અમલ ચાલુ જ રાખવા દે છે અથવા પ્રોગ્રામનો અનિયંત્રિત રીતે અણધાર્યો અંત લાવતા પહેલાં તે ઉપયોગકર્તાને તેની જાણ કરે છે.

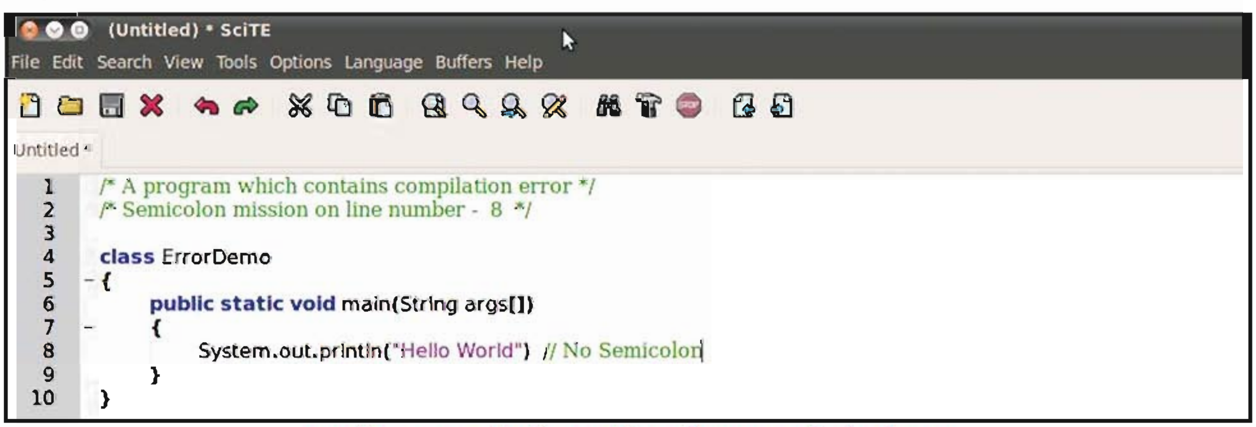
આ પ્રકરણમાં આપણે આપણા પ્રોગ્રામમાં ઉદ્‌ભવતા અપવાદરૂપ પરિસ્થિતિઓનું નિયમન કરવા માટેની યુક્તિઓ શીખીશું, તદ્‌ઉપરાંત જાવામાં ઉપલબ્ધ કેટલાક પ્રમાણભૂત અપવાદરૂપ પરિસ્થિતિઓ, તેમજ આપણા પ્રોગ્રામમાં અપવાદરૂપ પરિસ્થિતિ સર્જાય તેમ છતાં પ્રોગ્રામના ચોક્કસ ભાગનો હંમેશા અમલ થાય જ તે માટેની યુક્તિઓ પણ શીખીશું. અંતમાં, આપણે આપણા પોતાના અપવાદના પ્રકારોને વર્ણવીને તેનો ઉપયોગ કરવાની યુક્તિ વિશે જોઈશું.

અપવાદના પ્રકારો

જાવામાં, તમામ પ્રકારની ક્ષતિવાળી પરિસ્થિતિને અપવાદ તરીકે ઓળખવામાં આવે છે. પ્રોગ્રામમાં ઉદ્‌ભવતી ભૂલોને મુખ્યત્વે "કંપાઇલ કરતી વખતે ઉદ્‌ભવતી ભૂલો" (compile-time errors) અને "અમલ દરમિયાન ઉદ્‌ભવતી ભૂલો" (run-time errors) એમ બે પ્રકારોમાં વિભાજિત કરી શકાય.

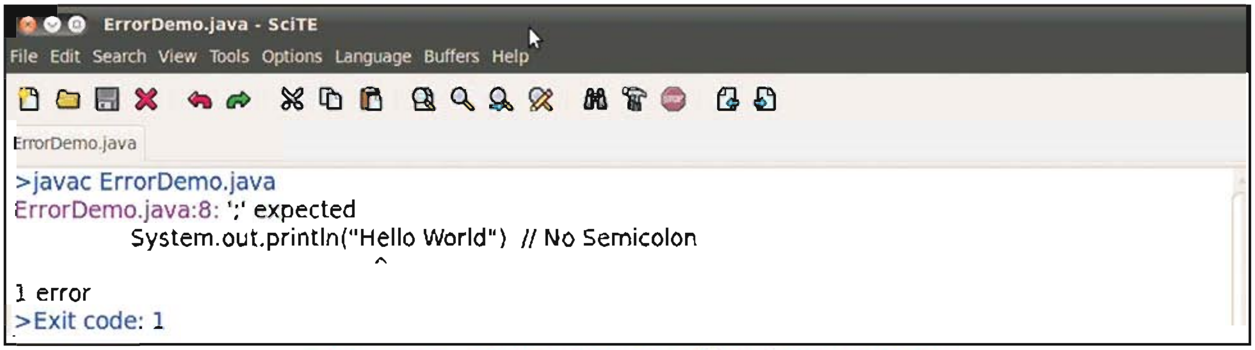
કંપાઇલ કરતી વખતે ઉદ્‌ભવતી ભૂલો

અગાઉના પ્રકરણમાં કોઈ પણ જાવા પ્રોગ્રામને કેવી રીતે કંપાઇલ કરવો તે આપણે શીખી ગયા છીએ. કોઈ પણ પ્રોગ્રામિંગ ભાષાની મૂળ સૂચનાઓ (sourcecode)ને કમ્પ્યૂટર દ્વારા અમલમાં મૂકી શકાય તેવી સૂચનાઓ (objectcode)માં રૂપાંતરિત કરવા માટે કંપાઇલરનો ઉપયોગ કરવામાં આવે છે. જો પ્રોગ્રામમાં કોઈ જોડણીની ભૂલ (syntax error) હશે, તો આપણને કંપાઇલેશન કરતી વખતે જ ભૂલ દર્શાવવામાં આવશે અને તેને કારણે આપણે ".class" ફાઇલ બનાવી શકીશું નહીં. કેટલીક સામાન્ય જોડણીની ભૂલોમાં અલ્પવિરામ " , " ચિહ્ન ન હોવું, અવ્યાખ્યાયિત ચલનો ઉપયોગ, કોઈ ચાવીરૂપ શબ્દની ખોટી જોડણી અને ઊઘડતા કૌંસની સંખ્યાના પ્રમાણમાં બંધ થતા કૌંસની સંખ્યાનો વધારો કે ઘટાડો વગેરેનો સમાવેશ થાય છે. આકૃતિ 10.1માં દર્શાવેલ જાવાપ્રોગ્રામમાં અર્ધવિરામ ચિહ્નની કમી છે.



આકૃતિ 10.1 : કંપાઇલ કરતી વખતે ભૂલ દર્શાવતો પ્રોગ્રામ

ઉપરનો પ્રોગ્રામ જ્યારે કંપાઇલ કરવામાં આવશે ત્યારે કંપાઇલ કરતી વખતે ભૂલ દર્શાવાશે. આવું એટલા માટે થશે, કારણ કે આપણે સૂચનાની લીટી નંબર-8ના અંતે અર્ધવિરામ (semicolon) ';' મૂકવાનું ભૂલી ગયા છીએ. આકૃતિ 10.2માં દર્શાવ્યા મુજબ જાવા કંપાઇલર જ્યારે કંપાઇલેશનનું પરિણામ દર્શાવે છે, ત્યારે તે જ્યાં ભૂલ જણાઈ હોય તે લીટીના ક્રમ (line number) સહિત ભૂલનો પ્રકાર પણ સૂચવે છે.



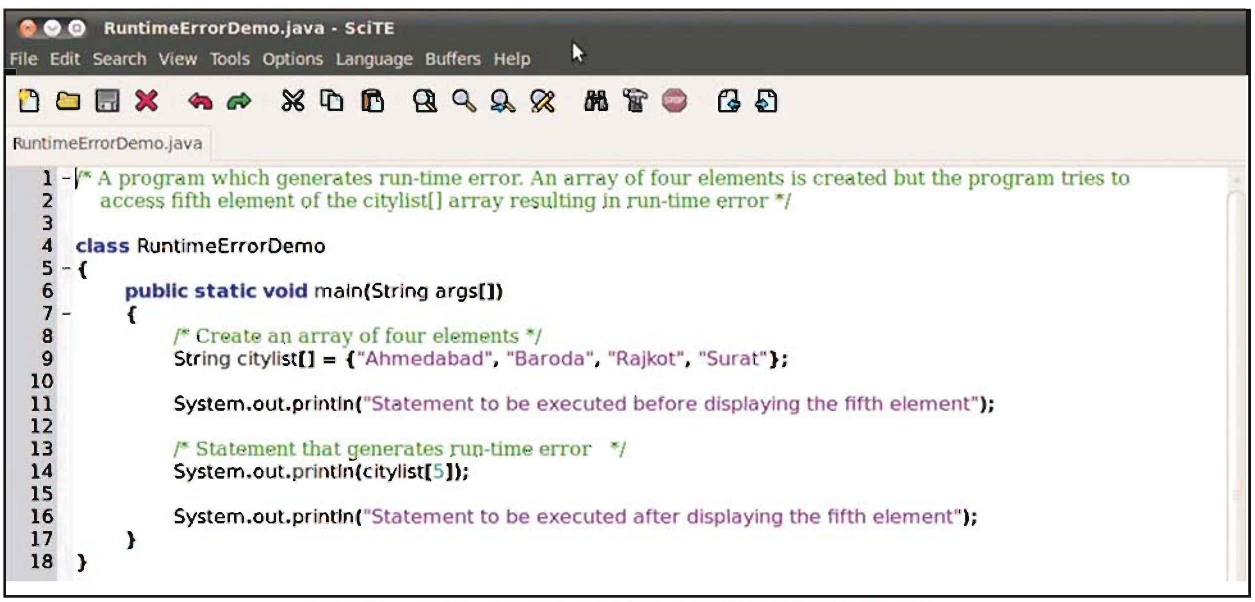
આકૃતિ 10.2 : આકૃતિ 10.1માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

પ્રોગ્રામને કંપાઇલ કરતી વખતે ઉદ્ભવતી ક્ષતિઓ મોટે ભાગે પ્રોગ્રામરની ભૂલો હોય છે અને તેને જ્યાં સુધી સુધારવામાં ન આવે ત્યાં સુધી તે પ્રોગ્રામને કંપાઇલ થવા દેતી નથી.

નોંધ : કમ્પ્યૂટરવિજ્ઞાનના ક્ષેત્રે, કોઈ પણ આદેશ કે પ્રોગ્રામનો સફળતાપૂર્વક અમલ થયો છે કે નહીં તેનો નિર્દેશ "Exit code" કે "Exit status" કરે છે. સંજ્ઞા 0 (શૂન્ય), આદેશ સફળતાપૂર્વક અમલમાં મુકાયાનો નિર્દેશ કરે છે, જ્યારે સંજ્ઞા 1 એવું દર્શાવે છે કે, કમાન્ડનો અમલ કરતી વખતે કોઈક સમસ્યા ઉદ્ભવી છે. આકૃતિ 10.2માં છેલ્લી લીટી Exit code : 1નો નિર્દેશ કરે છે. તેનો અર્થ એ થાય કે, ErrorDemo.java નામના કંપાઇલેશન પ્રોગ્રામનું કાર્ય સફળ રહ્યું નથી.

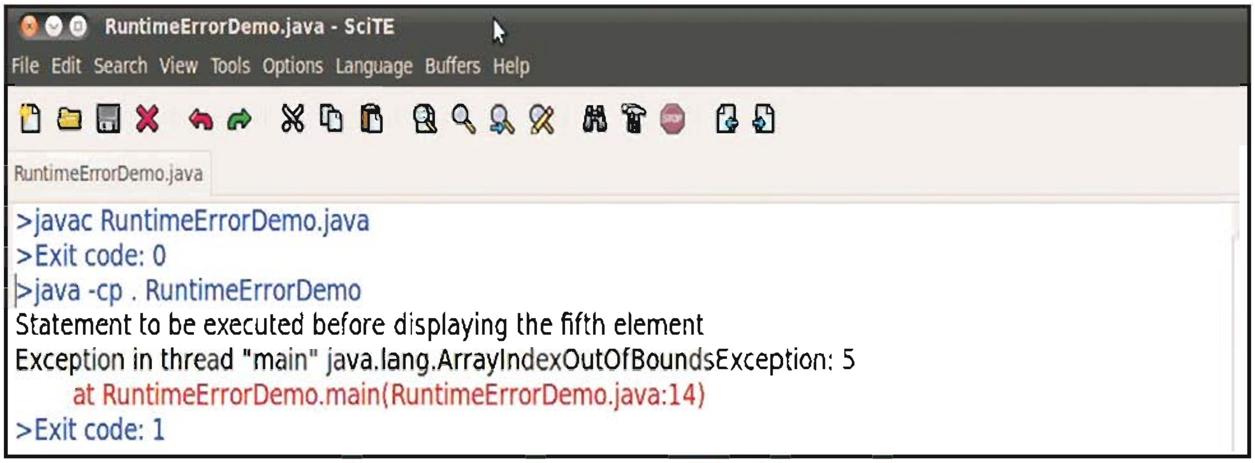
પ્રોગ્રામના અમલ દરમિયાન ઉદ્ભવતી ભૂલો

જો પ્રોગ્રામની સૂચનાઓમાં જોડણીની કોઈ ભૂલ નહીં હોય, તો પ્રોગ્રામ સફળતાપૂર્વક કંપાઇલ થશે અને આપણને ".class" ફાઇલ મળશે. જો કે, આમ છતાં પ્રોગ્રામ આપણી અપેક્ષા પ્રમાણે જ ચાલશે તેવી કોઈ ખાતરી મળતી નથી. તો ચાલો, આકૃતિ 10.3માં દર્શાવેલ અન્ય ઉદાહરણ જોઈએ, જે કંપાઇલ થઈ જશે, પણ અમલ કરતી વખતે અસામાન્ય સંજોગોમાં અટકી જશે.



આકૃતિ 10.3 : અમલ દરમિયાન ભૂલો દર્શાવતો પ્રોગ્રામ

આકૃતિ 10.3માં દર્શાવેલ પ્રોગ્રામમાં જોડણીની કોઈ ભૂલ ન હોવાથી તે કંપાઇલ થઈ જશે. પ્રોગ્રામમાં "citylist[]" નામનો ઍરે બનાવ્યો છે, જે ચાર જુદાં-જુદાં શહેરોનાં નામ સાચવશે. 14મી સૂચનામાં આપણે citylist નામના ઍરેના એવા ઘટકની વિગતો દર્શાવવાનો પ્રયત્ન કરીએ છીએ, જે હયાત જ નથી. અહીં આ કિસ્સો અપવાદરૂપ પરિસ્થિતિ છે. પ્રોગ્રામના અમલ દરમિયાન જ અપવાદ ઉદ્ભવે છે. પ્રોગ્રામના અમલનું પરિણામ આકૃતિ 10.4માં દર્શાવાયેલ છે.



આકૃતિ 10.4 : આકૃતિ 10.3માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

આકૃતિ 10.4માં દર્શાવેલ પરિણામમાંથી આપણે એ નોંધી શકીએ છીએ કે પ્રોગ્રામની સૂચના ક્રમ 14 પરથી પ્રોગ્રામનો અમલ અણધારી રીતે અંત પામ્યો. પરિણામમાં એક કારણદર્શક સંદેશ "ArrayIndexOutOfBoundsException" દર્શાવાયો છે, જે પ્રોગ્રામના અમલ દરમિયાન ઉદ્‌ભવેલ અપવાદનો નિર્દેશ કરે છે.

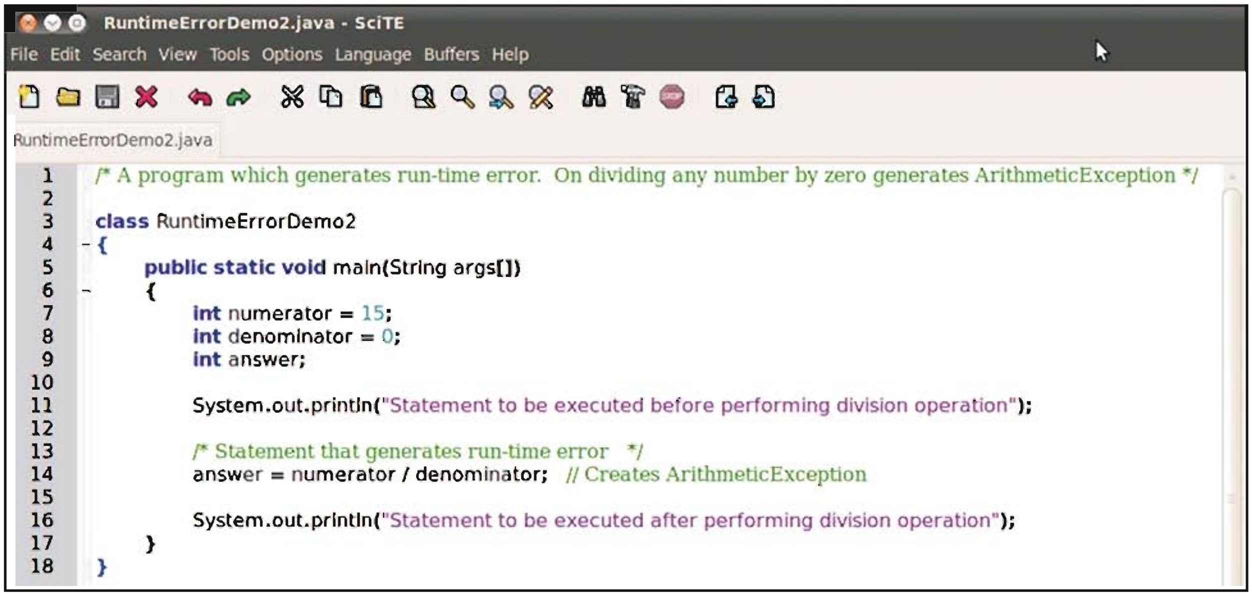
હવે આપણે આવા કેટલાક અન્ય કિસ્સા જોઈએ કે જે અપવાદ સર્જે છે. જાવામાં દરેક પ્રકારના અપવાદ માટે, સંબંધિત એક્સેપ્શન ક્લાસ (Exception class) ઉપલબ્ધ છે.

Java.lang અને java.io પૅકેજ વિવિધ અપવાદો સાથે કામ કરતા વર્ગોનો પદાનુક્રમ ધરાવે છે. બહોળા પ્રમાણમાં જોવા મળતા કેટલાક અપવાદોની યાદી કોષ્ટક 10.1માં આપવામાં આવેલ છે.

એક્સેપ્શન ક્લાસ (Exception Class)	અપવાદ માટેનું કારણ (Condition resulting in Exception)	ઉદાહરણ (Example)
ArrayIndexOutOfBoundsException	ઍરેના એલિમેન્ટ તરીકે ઍરેની મર્યાદાની બહારની હોય તેવી કિંમતનો ઉપયોગ.	int a[] = new int[4]; a[13] = 99;
ArithmeticException	કોઈ પણ સંખ્યાને શૂન્ય વડે ભાગવાનો પ્રયત્ન.	int a = 50 / 0;
FileNotFoundException	અસ્તિત્વમાં ન હોય તેવી ફાઇલનો ઉપયોગ કરવાનો પ્રયત્ન.	
NullPointerException	જ્યાં કોઈ ઑબ્જેક્ટ આવશ્યક હોય તેવા કિસ્સામાં ખાલી કિંમત (null)નો ઉપયોગ કરવાનો પ્રયત્ન.	String s = null; System.out.println(s.length());
NumberFormatException	કોઈ શબ્દમાળા (string)ને સાંખ્યિક રૂપમાં ફેરવવા માટેનો પ્રયત્ન.	String s = "xyz"; int i = Integer.parseInt(s);
PrinterIOException	મુદ્રણ સમયે થતી ઉદ્‌ભવેલ ઇનપુટ/આઉટપુટ ભૂલ (I/O error)	

કોષ્ટક 10.1 : બહોળા પ્રમાણમાં જોવા મળતા કેટલાક અપવાદોની યાદી

આકૃતિ 10.5 અપવાદ સર્જતો એક વધુ પ્રોગ્રામ દર્શાવે છે.



આકૃતિ 10.5 : ગાણિતિક અપવાદનું દ્રષ્ટાંત આપતો પ્રોગ્રામ

આકૃતિ 10.5માં દર્શાવેલ પ્રોગ્રામમાં આપણે એક પૂર્ણાંક સંખ્યાને શૂન્ય વડે ભાગવાનો પ્રયત્ન કરીએ છીએ. કોઈ પણ સંખ્યાને જ્યારે શૂન્ય વડે ભાગવામાં આવે છે, ત્યારે પરિણામ અનંતતામાં પરિણમે છે. અહીં પ્રોગ્રામ સફળતાપૂર્વક કંપાઇલ થશે, પરંતુ ગાણિતિક અપવાદને કારણે અનપેક્ષિત પરિણામ આપશે. પ્રોગ્રામના અમલનું પરિણામ આકૃતિ 10.6માં દર્શાવેલ છે.

આકૃતિ 10.6 : આકૃતિ 10.5માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

આપણે શૂન્ય વડે ભાગાકાર કરતા હોવાથી, જ્યારે ભાગાકારની ક્રિયા કરતું વિધાન અમલમાં આવશે, ત્યારે JVM તરત જ પ્રોગ્રામનો અંત લાવશે.

અપવાદરૂપ પરિસ્થિતિનું વ્યવસ્થાપન

અપવાદ એ ભૂલની સ્થિતિ છે. અપવાદરૂપ પરિસ્થિતિનું વ્યવસ્થાપન એ ક્ષતિઓનું વ્યવસ્થાપન કરવાની વસ્તુલક્ષી (object-oriented) યુક્તિ છે. અપવાદરૂપ પરિસ્થિતિઓનું વ્યવસ્થાપન કરતી વખતે એ બાબતની ખાતરી રાખવાનો પ્રયત્ન કરવામાં આવે છે કે, અણધારી રીતે પ્રોગ્રામનો અંત ન આવી જાય કે તે અનપેક્ષિત પરિણામ ન આપે. આ વિભાગમાં આપણે અપવાદરૂપ પરિસ્થિતિઓનું વ્યવસ્થાપન કેવી રીતે કરવું તે શીખીશું.

અપવાદરૂપ પરિસ્થિતિના વ્યવસ્થાપન માટેનો પ્રોગ્રામ (Exception handler) લખવા માટે જાવા try, catch અને finally જેવા કી વર્ડનો ઉપયોગ કરે છે. try, catch અને finally જેવા ચાવીરૂપ શબ્દો અપવાદ ઉપસ્થિત થાય ત્યારે ઉપયોગમાં લેવાય છે. આ કી વર્ડ વિધાનોના સમૂહને રજૂ કરે છે.

- tryના વિભાગ (Block)માં એવી સૂચનાઓનો સમાવેશ થાય છે કે જે, એક કે વધુ અપવાદોનો ઉદ્‌ભવ થવા દે છે.

- catch વિભાગમાં એવી સૂચનાઓ હોય છે કે જે, સંબંધિત try વિભાગમાં સર્જાય તેવા ચોક્કસ પ્રકારના અપવાદોની સંભાળ લેવા માટે લખવામાં આવી હોય છે.
- finally વિભાગનો અમલ હંમેશાં પ્રોગ્રામનો અંત આવે તે પહેલાં કરવામાં આવે છે. પછી ભલે, try વિભાગમાં કોઈ અપવાદો સર્જાયા હોય કે ન સર્જાયા હોય.

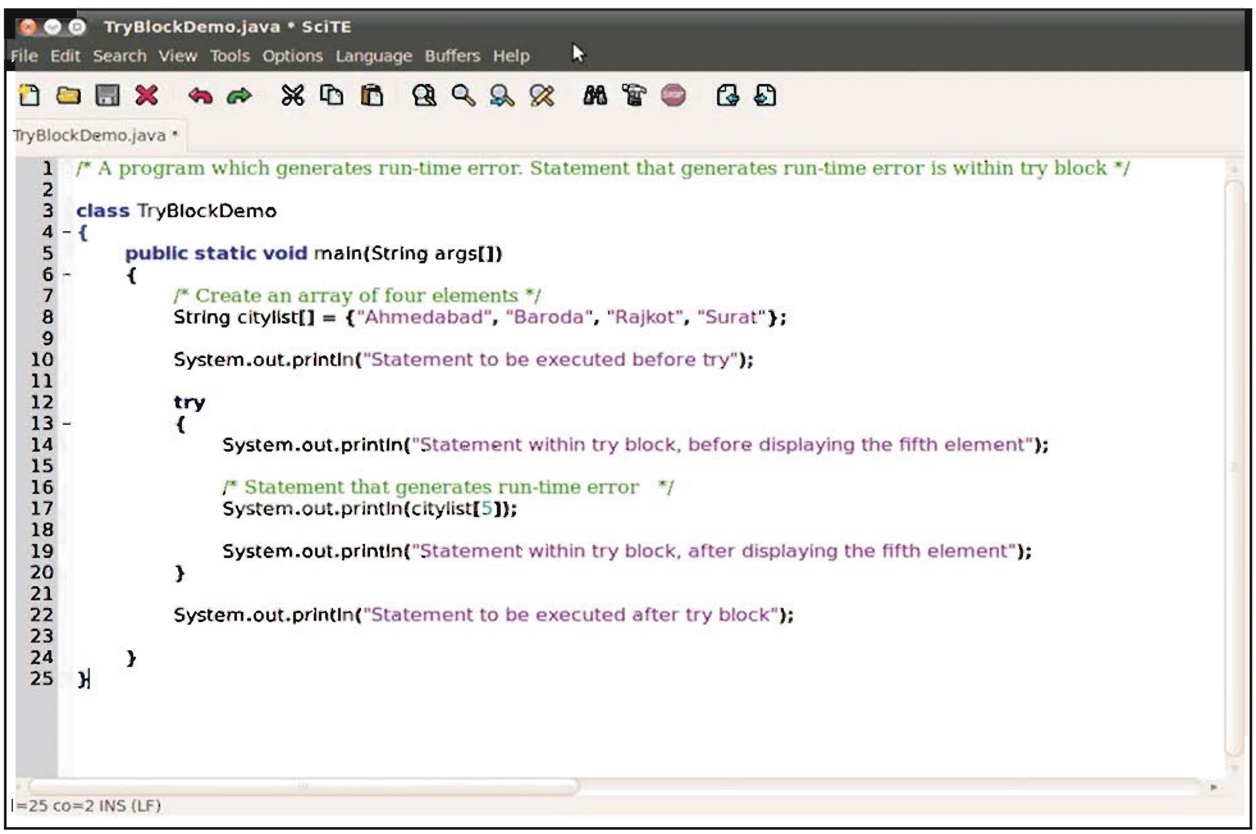
હવે આ ત્રણેય વિભાગોને એક પછી એક વિગતવાર સમજીએ.

Try વિભાગ

Try વિધાન કૌંસમાં સમાવિષ્ટ વિધાનોનો વિભાગ છે. આ વિધાનો આપણે જે અપવાદની દેખરેખ રાખવા ઇચ્છીએ છીએ, તે માટેની જરૂરી સૂચનાઓ છે. જો તેના અમલ દરમિયાન કોઈ સમસ્યા ઉદ્‌ભવે, તો એક અપવાદ ઊભો કરવામાં આવશે. જાવામાં દરેક પ્રકારનો અપવાદ એક ઑબ્જેક્ટને અનુરૂપ હોય છે. try વિભાગ એક કે તેથી વધુ અપવાદો ઊભા કરી શકે છે. try વિભાગની વાક્યરચના નીચે દર્શાવ્યા મુજબ છે :

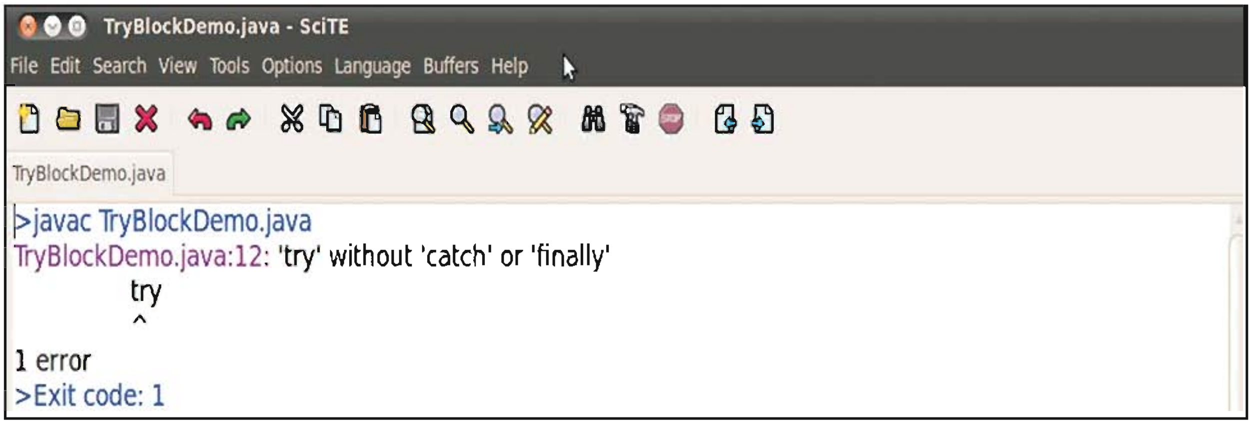
```
try
{
   // set of statements that may generate one or more exceptions.
}
```

હવે try વિભાગની અંદર લખવામાં આવતાં વિધાનો જોઈએ. તે એક અપવાદ સર્જે છે કે જે આપણે અગાઉ જોઈ ગયા છીએ. આકૃતિ 10.7માં દર્શાવેલ સૂચનાઓનો જ્યારે અમલ કરવામાં આવશે, ત્યારે તેમાં અપવાદના વ્યવસ્થાપન (Exception handler)ની વ્યવસ્થા ન હોવાને કારણે પ્રોગ્રામનો અંત લાવશે. પ્રોગ્રામના અમલ દરમિયાન ભૂલો સર્જાવાની શક્યતા ધરાવતી સૂચનાઓના ભાગને try વિભાગમાં જ લખવો જોઈએ.



આકૃતિ 10.7 : try વિભાગ સમજાવતો પ્રોગ્રામ

આકૃતિ 10.7માં દર્શાવેલ પ્રોગ્રામને કંપાઇલ કરવાનો પ્રયત્ન કરવામાં આવશે તો તે કંપાઇલેશન વખતે ભૂલ દર્શાવશે, કારણ કે, try બ્લૉક પછી catch વિભાગ અથવા finally વિભાગ હોવો જરૂરી છે. કંપાઇલેશન દરમિયાન સર્જાતી ભૂલ આકૃતિ 10.8માં દર્શાવેલ છે.



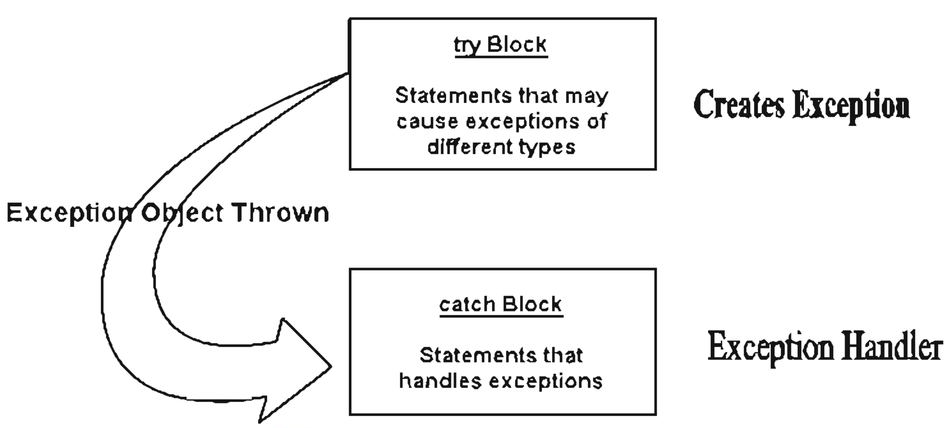
આકૃતિ 10.8 : આકૃતિ 10.7માં દર્શાવેલ પ્રોગ્રામની કંપાઈલેશન દરમિયાન સર્જાયેલ ભૂલ

Catch વિભાગ

Catch વિભાગ હંમેશા try વિભાગની તરત પાછળ હોવો જોઈએ. વ્યવસ્થાપન કરવા માટે જરૂરી સૂચનાઓ આ વિભાગમાં આવે છે. Catch વિભાગ અપવાદનું વ્યવસ્થાપન કરનારો વિભાગ હોવાથી જાવામાં તેને "એક્સેપ્શન હેન્ડલર" (Exception Handler) કહે છે. કોઈ એક try વિભાગ માટે એક કે તેથી વધુ catch વિભાગ હોઈ શકે છે. Catch વિભાગની વાક્યરચના નીચે દર્શાવ્યા મુજબ છે :

```
try
{
      // Set of statements that may generate one or more exceptions
 }
catch(Exception_Type Exception_object)
{
      // Code to handle the exception
 }
```

Catch વિભાગમાં કી વર્ડ તરીકે catch લખી તેની પાછળ એક પેરામીટર આપવામાં આવે છે. અપવાદના વ્યવસ્થાપન માટેની સૂચનાઓ કૌંસની વચ્ચે લખવાની હોય છે. આ વિભાગે કયા પ્રકારના અપવાદ સાથે કામ પાર પાડવાનું છે તે પેરામીટર દ્વારા દર્શાવવામાં આવે છે. જાવા વિવિધ પ્રકારના અપવાદો માટે સહાય આપે છે. આ મુદ્દાની ચર્ચાના પાછળના ભાગે આપણે અનેક catch વિભાગનો ઉપયોગ કરી કેવી રીતે વિવિધ પ્રકારના અપવાદોને પાર પાડી શકાય તે જોઈશું. આકૃતિ 10.9 try-catch વિભાગની રચના સમજાવે છે.



આકૃતિ 10.9 : અપવાદ-વ્યવસ્થાપન રચના

હવે યોગ્ય અપવાદ-વ્યવસ્થાપન ગોઠવતા પ્રોગ્રામનું ઉદાહરણ જોઈએ. આ સમજવા માટે હવે આપણે catch વિભાગમાં ઑબ્જેક્ટને ઝીલીને ArrayIndexOutOfBounds અપવાદનું વ્યવસ્થાપન કરીશું. પ્રકરણના પાછળના ભાગે આપણે પ્રોગ્રામનો અંત લાવ્યા વિના પ્રોગ્રામનો અમલ ચાલુ રાખવા માટેની સૂચનાઓને કેવી રીતે લખવી તે જોઈશું.

```java
1  /* A program which generates run-time error. Statement that generates run-time error is within
2     try block, there is a corresponding catch block immediately below the try block */
3
4  class TryCatchDemo
5  {
6          public static void main(String args[])
7          {
8                  /* Create an array of four elements */
9                  String citylist[] = {"Ahmedabad", "Baroda", "Rajkot", "Surat"};
10                 System.out.println("Statement to be executed before try");
11                 try
12                 {
13                         System.out.println("Statement within try block, before displaying the fifth element");
14
15                         /* Statement that generates run-time error   */
16                         System.out.println(citylist[5]);
17
18                         System.out.println("Statement within try block, after displaying the fifth element");
19                 }
20                 catch(ArrayIndexOutOfBoundsException eobj)
21                 {
22                         System.out.println("Within Catch Block");
23                         System.out.println("Caught Exception object of type : " + eobj);
24                 }
25                 System.out.println("Statement to be executed after try...catch");
26
27         }
28 }
```

આકૃતિ 10.10 : try...catch વિભાગનો ઉપયોગ સમજાવતો પ્રોગ્રામ

આકૃતિ 10.10માં દર્શાવેલ સૂચનાઓ સફળતાપૂર્વક કંપાઇલ થશે અને તેનો અમલ પણ થશે. આકૃતિ 10.10માં દર્શાવેલ પ્રોગ્રામમાં સૂચનાક્રમ 16 પરના વિધાનથી અપવાદ સર્જાશે. કારણકે, 16મી લીટી પર આપણે citylist[ ] ઍરેના પાંચમા ઘટકનો ઉપયોગ કરવાનો પ્રયત્ન કરીએ છીએ, જોકે ખરેખર ઍરેમાં તો માત્ર ચાર ઘટક જ છે. આપણો પ્રોગ્રામ એવી ઘટકકિંમત આપીને ઍરે ઘટકનો ઉપયોગ કરવા પ્રયત્ન કરે છે કે જે ઍરેની મર્યાદાની બહાર છે, જે સ્વાભાવિક રીતે જ પ્રોગ્રામને એક અપવાદ તરફ દોરી જાય છે. જ્યારે અપવાદ ઉદ્ભવે છે, ત્યારે ArrayIndexOutOfBoundsException પ્રકારનો ઑબ્જેક્ટ બનાવીને છોડવામાં આવશે. એ પછી તેને સંબંધિત catch વિભાગ આ અપવાદનું વ્યવસ્થાપન કરે છે અને અનપેક્ષિત રીતે પ્રોગ્રામનો અંત આવવા દેતો નથી. catch વિભાગ "eobj" ઑબ્જેક્ટનો નિર્દેશ ધરાવે છે, જેને try વિભાગ દ્વારા સર્જન કરીને મોકલવામાં આવેલ છે. આકૃતિ 10.11 પ્રોગ્રામનું પરિણામ દર્શાવે છે.

```
>javac TryCatchDemo.java
>Exit code: 0
>java -cp . TryCatchDemo
Statement to be executed before try
Statement within try block, before displaying the fifth element
Within Catch Block
Caught Exception object of type : java.lang.ArrayIndexOutOfBoundsException: 5
Statement to be executed after try...catch
>Exit code: 0
```

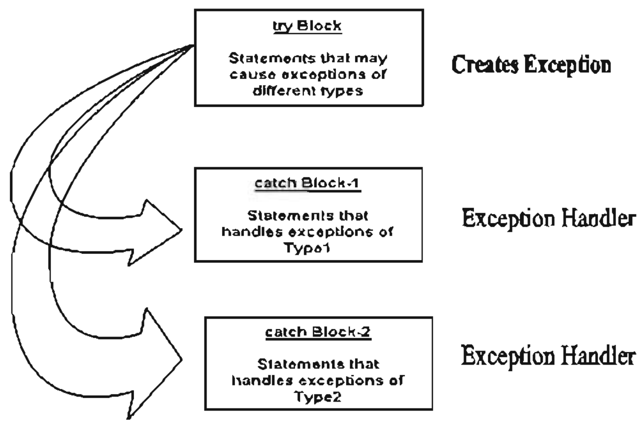
આકૃતિ 10.11 : આકૃતિ 10.10માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

આકૃતિ 10.11 પરથી આપણે અવલોકન કરી શકીએ છીએ કે અપવાદની ઉપસ્થિતિમાં પ્રોગ્રામનો અંત આવ્યો નહીં. "after try...catch" લખાણ ધરાવતા વિધાનનો અમલ કરી દર્શાવવામાં આવશે.

એક કરતાં વધુ catch વિભાગ

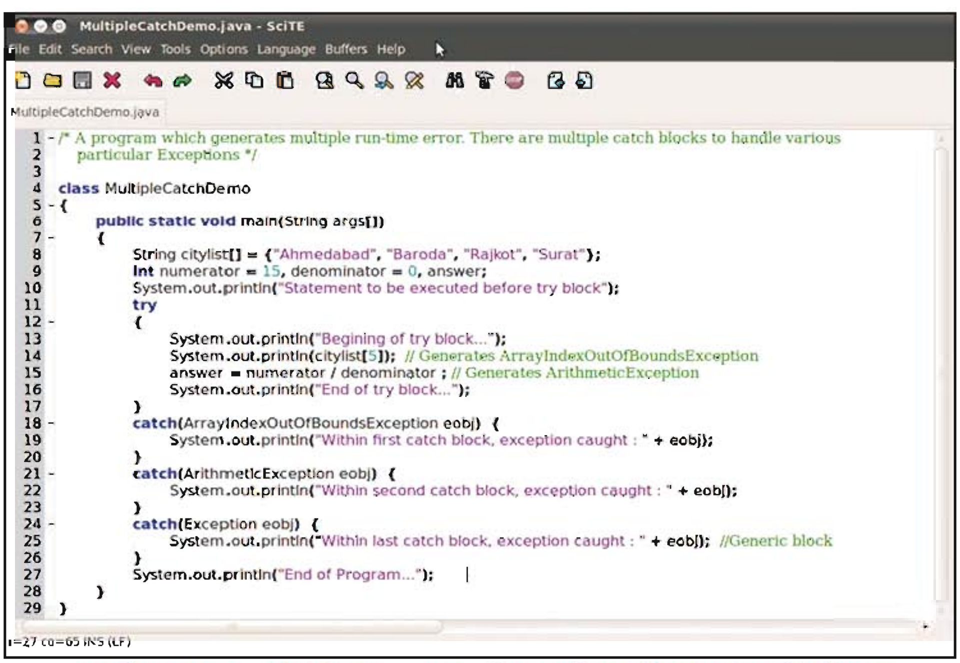
એક જ પ્રોગ્રામમાં અનેક અપવાદરૂપ પરિસ્થિતિ સર્જાઈ શકે છે. ઉદાહરણ તરીકે, ધારો કે આપણે એક ચોક્કસ ફાઈલને દૂરસ્થિત કમ્પ્યૂટર પર અપલોડ કરવી છે, તો તે બે સ્પષ્ટ અપવાદો તરફ દોરી જશે. એક, જો ફાઈલ આપણા કમ્પ્યૂટર પર ઉપલબ્ધ નહીં હોય, બીજો જો આપણું કમ્પ્યૂટર ઈન્ટરનેટ સાથે જોડાણ ધરાવતું ન હોય. એક કરતાં વધુ અપવાદોને સંભાળવા માટે જાવામાં જોગવાઈ છે. અગાઉ ચર્ચા કર્યા મુજબ, અપવાદ સર્જાવાની શક્યતા ધરાવતી સૂચનાઓ try વિભાગમાં જ લખાવી જોઈએ. એ સિવાય દરેક પ્રકારના અપવાદને અલગ રીતે સંભાળવા માટે એક કરતાં વધુ catch વિભાગ હોઈ શકે.

try વિભાગ જો વિવિધ પ્રકારના અનેક અપવાદો થ્રો (Throw) કરે, તો આપણે જુદા-જુદા અપવાદોનું વ્યવસ્થાપન કરી શકે તેવા અનેક catch વિભાગ લખી શકીએ. આ વ્યવસ્થા પ્રોગ્રામરને દરેક પ્રકારના અપવાદ માટે અલગ તર્ક (logic) લખવા મદદરૂપ થાય છે. આકૃતિ 10.12 અનેક catch વિભાગનો ઉપયોગ દર્શાવે છે.



10.12 : અનેક catch વિભાગ સાથે અપવાદ-વ્યવસ્થાપનની રચના

આકૃતિ 10.13 એવો પ્રોગ્રામ દર્શાવે છે, જેમાં એક કરતાં વધુ catch વિભાગનો ઉપયોગ કરવામાં આવેલ છે. અહીં try વિભાગની અંદર બે જુદા જ પ્રકારના અપવાદ ઊભા થયા. ArrayIndexOutOfBoundsException પ્રકારનો અપવાદ પ્રથમ catch વિભાગ દ્વારા પકડી લેવામાં આવશે અને ArithmaticExpression પ્રકારનો અપવાદ બીજા catch વિભાગ દ્વારા પકડી પાડવામાં આવશે.




```
/* A program which generates multiple run-time error. There are multiple catch blocks to handle various
   particular Exceptions */

class MultipleCatchDemo
{
	public static void main(String args[])
	{
		String citylist[] = {"Ahmedabad", "Baroda", "Rajkot", "Surat"};
		int numerator = 15, denominator = 0, answer;
		System.out.println("Statement to be executed before try block");
		try
		{
			System.out.println("Begining of try block...");
			System.out.println(citylist[5]);  // Generates ArrayIndexOutOfBoundsException
			answer = numerator / denominator ; // Generates ArithmeticException
			System.out.println("End of try block...");
		}
		catch(ArrayIndexOutOfBoundsException eobj)  {
			System.out.println("Within first catch block, exception caught : " + eobj);
		}
		catch(ArithmeticException eobj)  {
			System.out.println("Within second catch block, exception caught : " + eobj);
		}
		catch(Exception eobj)  {
			System.out.println("Within last catch block, exception caught : " + eobj);  //Generic block
		}
		System.out.println("End of Program...");
	}
}
```

l=27 co=65 INS (LF)

આકૃતિ 10.13 : એકથી વધુ catch વિભાગનો ઉપયોગ સમજાવતો પ્રોગ્રામ

છેલ્લો catch વિભાગ કોઈ પણ પ્રકારના અપવાદનું વ્યવસ્થાપન કરી શકે છે. તે એક ડીફોલ્ટ પ્રકારનો catch વિભાગ છે અને જ્યારે અનેક catch વિભાગ હોય, ત્યારે આ વિભાગને છેલ્લા વિભાગ તરીકે રાખવો પડે. પ્રોગ્રામ લખતી વખતે કોઈ ચોક્કસ catch બ્લૉકનો ક્રમ ક્યો રાખ્યો છે તે અસર કરતો નથી પરંતુ ડિફોલ્ટ વિભાગ તો બધા catch વિભાગ પછી છેલ્લે રાખવો પડે છે. ઉદાહરણ તરીકે, આકૃતિ 10.13માં દર્શાવેલ પ્રોગ્રામમાં આપણે 18મી લીટી પરથી શરૂ થતા catch વિભાગને 21મી લીટી પરથી શરૂ થતા catch વિભાગ વડે તેમના ક્રમમાં અદલબદલ કરી શકીએ. જોકે, 24મી લીટી પરથી શરૂ થતા catch વિભાગને છેલ્લે જ રાખવો પડે.

એક કરતાં વધુ try વિભાગને એકની અંદર બીજો, બીજાની અંદર ત્રીજો એમ નેસ્ટિંગ કરીને લખી શકાય, પરંતુ દરેક try વિભાગના સંબંધિત catch વિભાગ લખવાની કાળજી લેવી જરૂરી છે.

Finally વિભાગ

સામાન્ય રીતે, Finally વિભાગ try વિભાગનો અમલ કર્યા પછી અંતે clean up કરવા માટે વપરાય છે. સંબંધિત try વિભાગની અંદરથી ક્યા અપવાદો થ્રો કરવામાં (thrown) આવશે તેની પરવા કર્યા વગર જ્યારે આપણે એ ખાત્રી કરવા ઇચ્છીએ છીએ કે, કોઈ ચોક્કસ સૂચનાઓનો અમલ કરવાનો છે, ત્યારે આપણે finally વિભાગનો ઉપયોગ કરીએ છીએ. સંબંધિત try વિભાગ દ્વારા અપવાદ થ્રો કરવામાં આવ્યા છે કે નહીં તેની પરવા કર્યા વિના finally વિભાગનો અમલ ચોક્કસપણે કરવામાં આવે જ છે. પ્રોગ્રામની પૂર્ણતાના સમયે જો ફાઇલ બંધ કરવી જરૂરી હોય અથવા કોઈ અગત્યનું સંસાધન મુક્ત કરવાનું હોય ત્યારે finally વિભાગ બહોળા પ્રમાણમાં ઉપયોગમાં લેવાય છે. finally વિભાગની વાક્યરચના નીચે મુજબ છે :

```
finally
  {
     // clean-up code to be executed last
   // statements within this block always get executed even though if run-time errors
        terminate the program abruptly
  }
```

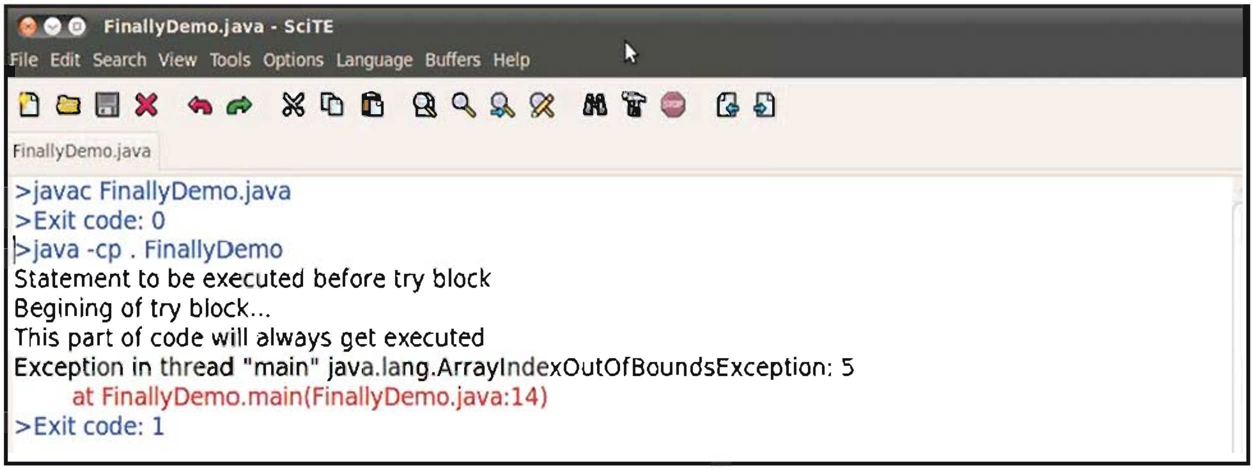
દરેક try વિભાગની પાછળ ઓછામાં ઓછો એક વિભાગ તો હોવો જ જોઈએ, જે catch વિભાગ હોય અથવા finally વિભાગ હોય. આકૃતિ 10.14 finally વિભાગનો ઉપયોગ કરવાનું ઉદાહરણ દર્શાવે છે.

FinallyDemo.java • SciTE

```java
/* A program which generates multiple run-time error. There is a finally block following try block.
   There are no catch blocks in this program */

class FinallyDemo
{
	public static void main(String args[])
	{
		String citylist[] = {"Ahmedabad", "Baroda", "Rajkot", "Surat"};
		int numerator = 15, denominator = 0, answer;
		System.out.println("Statement to be executed before try block");
		try
		{
			System.out.println("Begining of try block...");
			System.out.println(citylist[5]);  // Generates ArrayIndexOutOfBoundsException
			answer = numerator / denominator ; // Generates ArithmeticException
			System.out.println("End of try block...");
		}
		finally
		{
			System.out.println("This part of code will always get executed");
		}

		System.out.println("End of Program...");
	}
}
```

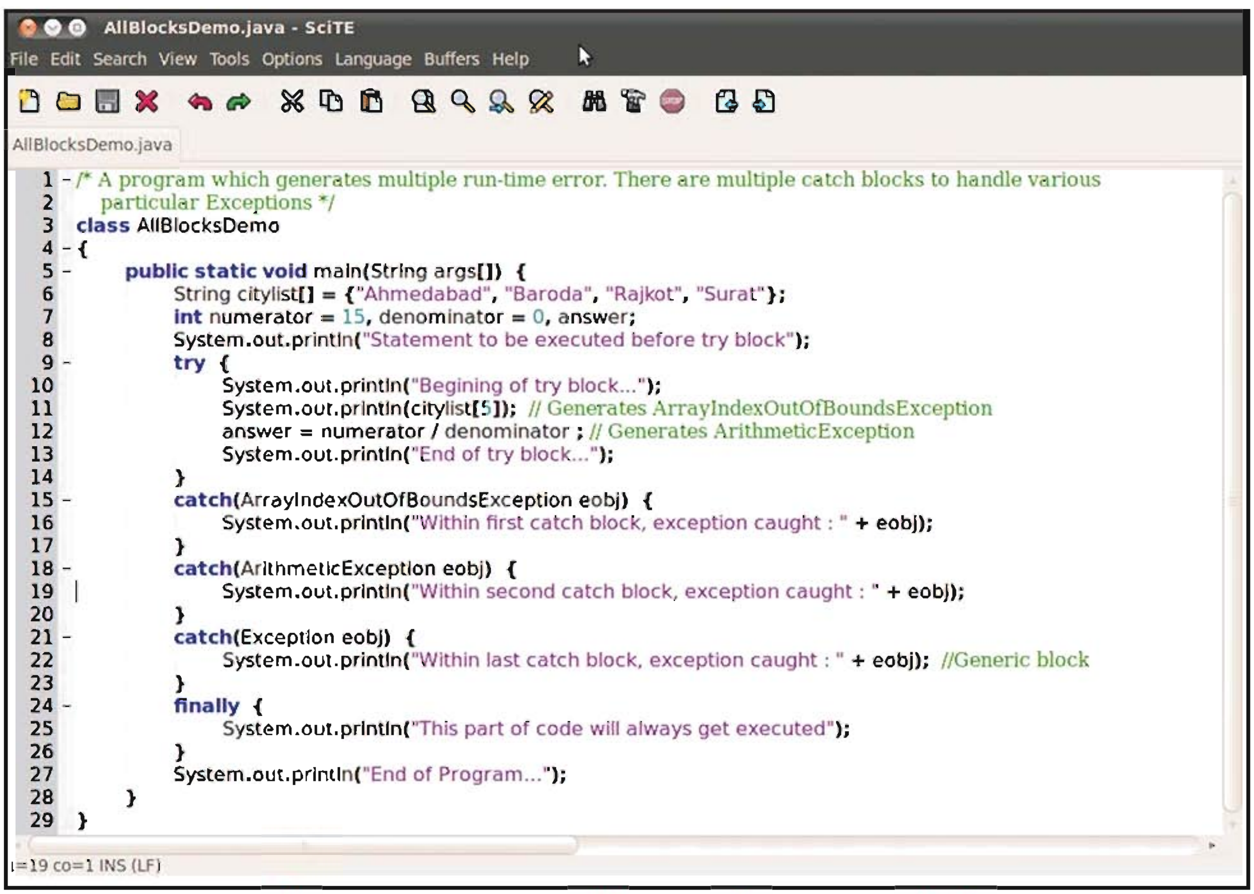
આકૃતિ 10.14 : catch વિભાગ સિવાય finally વિભાગ સમજાવતો પ્રોગ્રામ

આકૃતિ 10.14માં દર્શાવેલ પ્રોગ્રામમાં એક પણ catch વિભાગ ન હોવાને લીધે, 14મી લીટી પર ઉદ્‌ભવતા અપવાદને કારણે અધવચ્ચેથી પ્રોગ્રામનો અંત આવી જાય છે; 15 અને 16મી લીટી પર આવેલાં વિધાનોનો અમલ થશે નહીં. જોકે, finally વિભાગની ઉપસ્થિતિને લીધે, પ્રોગ્રામનો અંત આવે તે પહેલાં પ્રોગ્રામ finally વિભાગમાં પડેલાં વિધાનોનો અમલ કરે છે. પ્રોગ્રામનું પરિણામ આકૃતિ 10.15માં દર્શાવેલ છે.

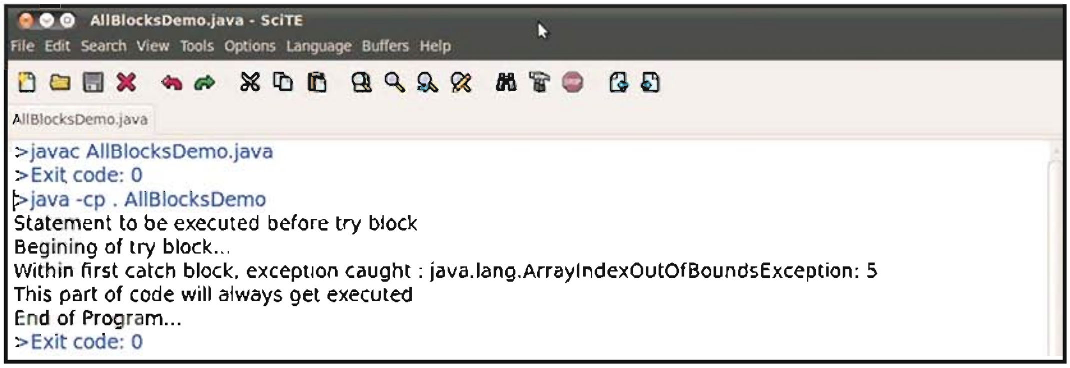

```
FinallyDemo.java - SciTE
File Edit Search View Tools Options Language Buffers Help
FinallyDemo.java
>javac FinallyDemo.java
>Exit code: 0
>java -cp . FinallyDemo
Statement to be executed before try block
Begining of try block...
This part of code will always get executed
Exception in thread "main" java.lang.ArrayIndexOutOfBoundsException: 5
	at FinallyDemo.main(FinallyDemo.java:14)
>Exit code: 1
```

આકૃતિ 10.15 : આકૃતિ 10.14માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

હવે ઘણા બધા catch વિભાગ અને એક finally વિભાગ એમ બધાનો એકસાથે ઉપયોગ કરાયો હોય, તેવું એક ઉદાહરણ જોઈએ. આકૃતિ 10.16માં દર્શાવેલ પ્રોગ્રામ આ બધા વિભાગ ધરાવે છે. એ રીતે, try વિભાગમાંથી ઉદ્‌ભવતા દરેક અપવાદ માટેના સંબંધિત અનેક catch વિભાગ સાથેનો આ એક સંપૂર્ણ પ્રોગ્રામ છે. પ્રોગ્રામનું પરિણામ આકૃતિ 10.17માં દર્શાવેલ છે.


```
AllBlocksDemo.java - SciTE
File Edit Search View Tools Options Language Buffers Help
AllBlocksDemo.java
/* A program which generates multiple run-time error. There are multiple catch blocks to handle various
   particular Exceptions */
class AllBlocksDemo
{
    public static void main(String args[]) {
        String citylist[] = {"Ahmedabad", "Baroda", "Rajkot", "Surat"};
        int numerator = 15, denominator = 0, answer;
        System.out.println("Statement to be executed before try block");
        try {
            System.out.println("Begining of try block...");
            System.out.println(citylist[5]);  // Generates ArrayIndexOutOfBoundsException
            answer = numerator / denominator ; // Generates ArithmeticException
            System.out.println("End of try block...");
        }
        catch(ArrayIndexOutOfBoundsException eobj) {
            System.out.println("Within first catch block, exception caught : " + eobj);
        }
        catch(ArithmeticException eobj) {
            System.out.println("Within second catch block, exception caught : " + eobj);
        }
        catch(Exception eobj) {
            System.out.println("Within last catch block, exception caught : " + eobj);  //Generic block
        }
        finally {
            System.out.println("This part of code will always get executed");
        }
        System.out.println("End of Program...");
    }
}
l=19 co=1 INS (LF)
```

આકૃતિ 10.16 : try, catch અને finally વિભાગ સમજાવતો પ્રોગ્રામ



```
>javac AllBlocksDemo.java
>Exit code: 0
>java -cp . AllBlocksDemo
Statement to be executed before try block
Begining of try block...
Within first catch block, exception caught : java.lang.ArrayIndexOutOfBoundsException: 5
This part of code will always get executed
End of Program...
>Exit code: 0
```

આકૃતિ 10.17 : આકૃતિ 10.16માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

પરિણામ પરથી એ સ્પષ્ટ થાય છે કે, 11મી લીટીથી પ્રોગ્રામના અમલનો અંકુશ પ્રથમ catch વિભાગ પર બદલાય છે, પછીથી તે finally વિભાગ પર બદલાય છે. છેલ્લા બે catch વિભાગનો અમલ થયો નહીં. આખો પ્રોગ્રામ ચાલ્યો નથી તેમ છતાં તેનો યોગ્ય રીતે અંત આવ્યો છે.

Finally વિભાગ કોઈ એક ચોક્કસ try વિભાગ સાથે જોડાયેલ હોય છે અને તે સંબંધિત try વિભાગ માટેના કોઈ પણ catch વિભાગ પછી તરત જ આવેલો હોવો જોઈએ. જો પ્રોગ્રામમાં કદાચ catch વિભાગ ન હોય તો finally વિભાગ try વિભાગની તરત પછી મૂકી શકાય. જો finally અથવા catch વિભાગ સચોટ જગ્યાએ મૂકવામાં નહીં આવ્યા હોય, તો પ્રોગ્રામ કંપાઇલ થઈ શકશે નહીં.

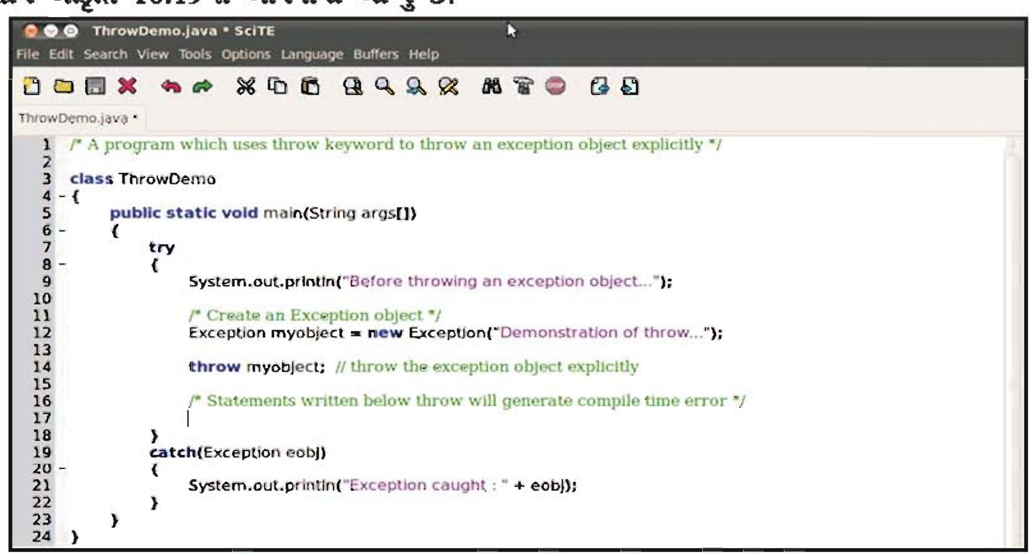
Throw વિધાન

અપવાદના ઑબ્જેક્ટને નિશ્ચિત રૂપે થ્રો કરવા (throw) કી વર્ડનો ઉપયોગ કરવામાં આવે છે. અત્યાર સુધી આપણે જોયેલા ઉદાહરણરૂપ પ્રોગ્રામોમાં JVM અપવાદરૂપ ઑબ્જેક્ટ તૈયાર કરીને આપોઆપ થ્રો કરતું હતું. ઉદાહરણ તરીકે, જ્યારે આપણે શૂન્ય વડે ભાગાકાર કરવાની ક્રિયા કરવાનો પ્રયત્ન કર્યો, ત્યારે ArithmeticExceptionનો ઑબ્જેક્ટ બનાવવામાં આવ્યો અને JVM દ્વારા આપોઆપ તેને થ્રો કરવામાં આવ્યો.

અપવાદનો ઑબ્જેક્ટ બનાવીને નિશ્ચિત રૂપે થ્રો કરવા જાવા વ્યવસ્થા પૂરી પાડે છે. આપણે જે ઑબ્જેક્ટને થ્રો કરીએ, તે java.lang.Throwable (Throwable ક્લાસ અથવા તેના કોઈ પણ સબ-ક્લાસનો ઑબ્જેક્ટ) પ્રકારનો હોવો જ જોઈએ, અન્યથા કંપાઇલ કરતી વખતે ભૂલ ઉદ્‌ભવશે. અપવાદ ઑબ્જેક્ટને થ્રો કરવા માટેની વાક્યરચના નીચે મુજબ છે :

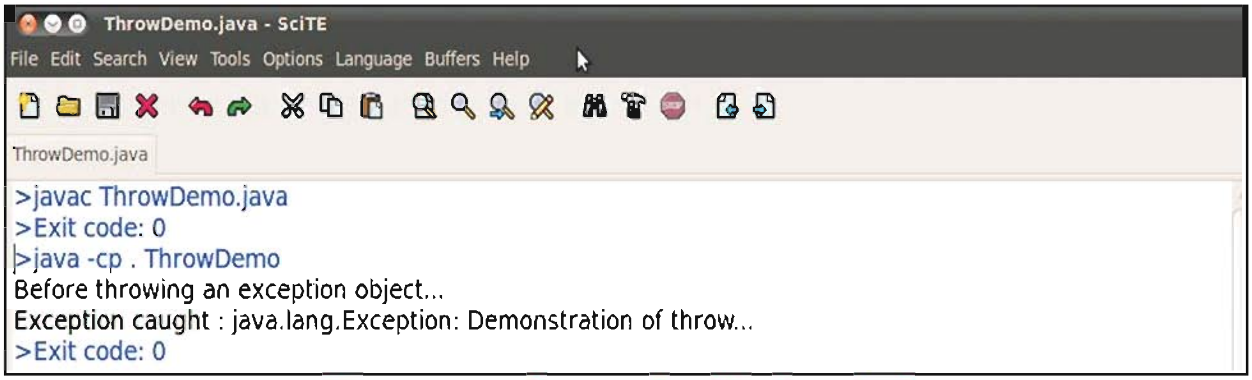
throw exception_object;

જ્યારે throw વિધાન મળશે ત્યારે, સંબંધિત catch વિભાગની શોધ શરૂ થઈ જશે. try કે catch વિભાગ પછીના કોઈ પણ વિધાનનો અમલ કરવામાં આવશે નહીં. આકૃતિ 10.18માં લખેલ સૂચનાઓ throw વિધાનનો ઉપયોગ દર્શાવે છે અને એનું પરિણામ આકૃતિ 10.19માં આપવામાં આવ્યું છે.



```
/* A program which uses throw keyword to throw an exception object explicitly */

class ThrowDemo
{
	public static void main(String args[])
	{
		try
		{
			System.out.println("Before throwing an exception object...");

			/* Create an Exception object */
			Exception myobject = new Exception("Demonstration of throw...");

			throw myobject;  // throw the exception object explicitly

			/* Statements written below throw will generate compile time error */

		}
		catch(Exception eobj)
		{
			System.out.println("Exception caught : " + eobj);
		}
	}
}
```

આકૃતિ 10.18 : throw વિધાનનો ઉપયોગ દર્શાવતો પ્રોગ્રામ



આકૃતિ 10.19 : આકૃતિ 10.18માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

આકૃતિ 10.18માં દર્શાવેલ પ્રોગ્રામમાં આપણે Exception ક્લાસનો eobj નામનો એક ઑબ્જેક્ટ બનાવ્યો. એ જ ઑબ્જેક્ટને throw વિધાનનો ઉપયોગ કરીને આપણે થ્રો કર્યો. નિશ્ચિતપણે થ્રો કરવામાં આવતા અપવાદ-ઑબ્જેક્ટને સંભાળવા માટે catch વિભાગ હોવો જ જોઈએ.

throws ઉપવાક્ય

આપણે try-catch-finally વિભાગને જોયા. અત્યાર સુધી આપણે ચર્ચેલા પ્રોગ્રામ એ સાદા પ્રોગ્રામ હતા, જેમાં પદ્ધતિ (method)ના ઉપયોગનો સમાવેશ કરવામાં આવ્યો ન હતો. અહીં એવો વિચાર આવે કે, જો method કે constructor દ્વારા અપવાદ ઉદ્ભવે તો શું થાય ? એવા સંજોગોમાં try-catch વિભાગને આપણે ક્યાં મૂકીશું ? method દ્વારા ઉદ્ભવતા અપવાદનું વ્યવસ્થાપન કરવા માટે બે વૈકલ્પિક રસ્તા છે :

- method કે constructorમાં જ try-catch વિભાગ લખો કે જે કદાચ અપવાદનું ઉદ્ભવકેન્દ્ર છે.
- try વિભાગની અંદર જ method કે constructorને ઇન્વોક કરીને (કે જે કદાચ અપવાદ સર્જી શકે.)

constructor કે methodની અંદરની સૂચના અપવાદ થ્રો કરી શકે તેમ છે, તે જણાવવા method કે constructorને વ્યાખ્યાયિત કરતી વખતે throws ઉપવાક્યનો ઉપયોગ કરી શકાય. તે એવો પણ સંકેત કરે છે કે, methodમાં અપવાદનું વ્યવસ્થાપન કરી શકે તેવો catch વિભાગ નથી. જ્યારે આપણે constructor કે method લખીએ છીએ કે જે તેને બોલાવનાર તરફ અપવાદ મોકલી શકે, તો તે હકીકતનું દસ્તાવેજીકરણ કરવું ઉપયોગી છે. throws ચાવીરૂપ શબ્દ methodની ઘોષણા કરતી વખતે જ ઉપયોગ કરાય છે.

methodની ઘોષણા કરતી વખતે throws ઉપવાક્યનો નીચે મુજબ ઉપયોગ કરી શકાય છે :

method_Modifiers return_type method_Name(parameters) throws Exception list... {

.....

// body of the method

.....

}

એક method અનેક અપવાદો થ્રો કરી શકે છે. method દ્વારા થ્રો કરી શકાતા દરેક પ્રકારના અપવાદને methodના headerમાં દર્શાવવો જરૂરી છે. ઉદાહરણ તરીકે methodનું header નીચે મુજબ હોઈ શકે.

performDivision() throws ArithmeticException, ArrayIndexOutOfBoundsException

{

.....

// body of the method

.....

}

આકૃતિ 10.20માં દર્શાવેલ પ્રોગ્રામ ઉપયોગકર્તા-નિર્મિત methodની ઉપસ્થિતિમાં throws ઉપવાક્યના ઉપયોગનું નિદર્શન કરે છે. એ પછીની સૂચનાઓમાં એ નોંધવું ખાસ જરૂરી છે કે, જો method અપવાદ ઑબ્જેક્ટને બહાર પાડે, તો તેને અનુરૂપ catchhandler હોવું જોઈએ. જોકે, તેમ છતાં, જો આપણે methodની અંદર જ અપવાદનો પ્રકાર ઝીલતા હોઈએ, તો તેને થ્રો કરવાની કોઈ જરૂર રહેતી નથી.

```
/* A program which uses throws keyword to throw an exception from any method */

class ThrowsDemo
{
	public static void main(String args[])
	{
		try
		{
			performDivision();	// This method throws exception
		}
		catch(ArithmeticException eobj)
		{
			System.out.println("Exception caught : " + eobj);
		}
	}

	/* Method that throws ArithmeticException object */

	public static void performDivision() throws ArithmeticException
	{
		int ans;
		ans = 15 / 0;
	}
}
```

આકૃતિ 10.20 ચાવીરૂપ શબ્દ throwsનો ઉપયોગ સમજાવતો પ્રોગ્રામ

આકૃતિ 10.20માં દર્શાવેલ પ્રોગ્રામમાં આપણે PerformDivision() નામની એક મેથડ (method) લખી છે, આ મેથડમાં ArithmeticExceptionનો ઑબ્જેક્ટ ઉદ્ભવશે. આપણા પ્રોગ્રામની main() મેથડમાં performDivision() નામની મેથડને કોલ કરવામાં આવે છે. અત્રે એ તદ્દન સ્વાભાવિક છે કે, PerformDivision() મેથડમાં અપવાદ-વ્યવસ્થાપનતંત્ર ન હોવાને લીધે બોલાવનાર મેથડ દ્વારા જ અપવાદ ઝીલવામાં આવશે અને તેથી બોલાવનાર મેથડમાં અપવાદ-વ્યવસ્થાપન માટે Exception handler હોવું જોઈએ. એ જ રીતે, જાવાક્લાસના Constructorsમાં પણ અપવાદો હોઈ શકે છે.

જરૂરિયાત મુજબના અપવાદોનું સર્જન

જાવા જે-તે વિનિયોગની ખાસ સમસ્યાઓ અનુસાર આપણા પોતાના અપવાદોનું સર્જન કરવા દે છે. ઉદાહરણ તરીકે, ધારોકે, આપણે ગુણપત્રક તૈયાર કરવાનો પ્રોગ્રામ લખતા હોઈએ, જેમાં ઉપયોગકર્તાને વિવિધ વિષયના ગુણ દાખલ કરવા માટે પૂછવામાં આવતું હોય. દરેક વિષયના ગુણ 0થી 100ની વચ્ચે જ હોવા જોઈએ, ધારોકે, ઉપયોગકર્તા કોઈ વિષયના ગુણ તરીકે ઋણ સંખ્યા અથવા 100થી ઉપરની સંખ્યા દાખલ કરે, તો પ્રોગ્રામે તરત જ અપવાદ ઊભો કરવો જોઈએ. આવા પ્રકારના અપવાદ જે-તે વિનિયોગ પૂરતા ખાસ હોય છે. જાવા આવા વિનિયોગ માટે ખાસ એવા અપવાદો માટે આંતરપ્રસ્થાપિત અપવાદ-વર્ગ (Built-in Exception Classes) આપતા નથી.

એક્સેપ્શન ક્લાસનો સબક્લાસ બનાવીને આપણે ઉપયોગકર્તા-નિર્મિત અપવાદો તૈયાર કરી શકીએ. આ અપવાદોને throws વિધાનની મદદથી બાહ્ય રીતે થ્રો કરી શકાય. જોકે, આ અપવાદ ઝીલીને તેને યોગ્ય રીતે પાર પાડવા જરૂરી છે. એક પ્રોગ્રામ જોઈએ, જે ગુણની યથાર્થતા ચકાસવા જરૂરિયાત મુજબના અપવાદનું સર્જન કરતો હોય.

આકૃતિ 10.21માં દર્શાવેલ પ્રોગ્રામ ઉપયોગકર્તા તરફથી ઇનપુટ સ્વીકારે છે. 21મી લીટી પર, કી-બોર્ડથી ઇનપુટ સ્વીકારવા આપણે java.util.Scanner ક્લાસનો ઉપયોગ કર્યો છે. scanner ક્લાસની nextInt() પદ્ધતિ કૉન્સોલ (console) પરથી પૂર્ણાંક સંખ્યા વાંચવામાં મદદરૂપ બને છે. scanner ક્લાસની કાર્યપદ્ધતિ વિશેની ચર્ચા આપણે હવે પછીના પ્રકરણમાં કરીશું, જેમાં ફાઇલ અને I/Oની ચર્ચા કરવામાં આવેલ છે.

અહીં આપણે બે વર્ગ પ્રસ્થાપિત કર્યા. જરૂરિયાત અનુસારના અપવાદ (Custom Exception) તૈયાર કરવા એક વધારાના ક્લાસની જરૂરિયાત છે. "InvalidMarksException" વર્ગ java.lang પૅકેજના અપવાદવર્ગ (Exception Class)ને વિસ્તારે છે, તે સિંગલ પેરામીટર કન્સ્ટ્રક્ટર (Single Parameter Constructor) ધરાવે છે, જે ભૂલના પ્રકારને વર્ણવવા માટે શબ્દમાળાને (String) સ્વીકારે છે.

```
CustomExceptionDemo2.java - SciTE
File Edit Search View Tools Options Language Buffers Help
CustomExceptionDemo2.java

/* A program that creates Custom Exceptions */
/* Class InvalidMarksException is a user defined class which is inherited from java.lang.Exception class
   This program wont proceed untill valid marks are entered*/

import java.util.Scanner;

class InvalidMarksException extends java.lang.Exception
{
        public InvalidMarksException(String message)
        {
                super(message);
        }
}


class CustomExceptionDemo2
{
        public static void main(String args[])
        {
                Scanner kbinput = new Scanner(System.in);
                int marks;
                boolean continueLoop = true;
                do {
                        System.out.println("Enter the marks : ");
                        marks = kbinput.nextInt();
                        System.out.println("You entered " + marks);
                        try
                        {
                                if(marks < 0 || marks > 100)  {
                                        throw new InvalidMarksException("Wrong marks..."); // Throw Customized Exception ob
                                }
                                else {
                                        System.out.println("Marks are Valid");
                                }
                                continueLoop = false;
                        }
                        catch(InvalidMarksException eobj)
                        {
                                System.out.println("Exception caught : " + eobj);
                        }
                }
                while(continueLoop);
        }
}
```

આકૃતિ 10.21 : ઉપયોગકર્તા નિર્મિત અપવાદવર્ગ

મુખ્ય પદ્ધતિમાં, (વિનિયોગ માટે ખાસ) વ્યાવસાયિક તર્ક મુજબ સૂચના લખવામાં આવી છે, જે ગુણ નિર્ધારિત મર્યાદાની અંદર જ છે કે કેમ તેની ખાત્રી કરે છે. જો દાખલ કરેલ ગુણ નિર્ધારિત મર્યાદાની અંદર નહીં આવતા હોય તો આપણે "InvalidMarksException" પ્રકારનો ઑબ્જેક્ટ તૈયાર કરીશું અને તેને થ્રો કરીશું. આ અપવાદને પાર પાડવા માટે એક catch વિભાગ હોવો જરૂરી છે. જ્યાં સુધી ગુણની યોગ્ય સંખ્યા દાખલ નહીં કરાય, ત્યાં સુધી આ પ્રોગ્રામ આગળ વધશે નહીં. જો ઉપયોગકર્તા ગુણની અયોગ્ય સંખ્યા દાખલ કરે તેવા કિસ્સામાં જ્યાં સુધી યોગ્ય સંખ્યા દાખલ નહીં થાય, ત્યાં સુધી ઉપયોગકર્તાને યોગ્ય સંખ્યા દાખલ કરવા કહ્યા કરવામાં આવશે. અત્રે એ નોંધવું જોઈએ કે, try-catch વિભાગને do-while લૂપની અંદર મૂકવામાં આવેલ છે. પ્રોગ્રામનું પરિણામ આકૃતિ 10.22માં દર્શાવવામાં આવેલ છે.

```
Terminal
File Edit View Terminal Help

$ javac CustomExceptionDemo2.java
$ java CustomExceptionDemo2
Enter the marks :
145
You entered 145
Exception caught : InvalidMarksException: Wrong marks...
Enter the marks :
-47
You entered -47
Exception caught : InvalidMarksException: Wrong marks...
Enter the marks :
78
You entered 78
Marks are Valid
$
```

આકૃતિ 10.22 : આકૃતિ 10.21માં દર્શાવેલ પ્રોગ્રામનું પરિણામ

નોંધ : SciTE એ પ્રોગ્રામ ટાઈપ કરવા માટેનું એડિટર છે. જો પ્રોગ્રામ કી-બોર્ડ પાસેથી ડેટા સ્વીકારતો હોય તો પ્રોગ્રામને કમાન્ડપ્રોમ્પ્ટ પર અમલમાં મૂકવો સલાહભર્યું છે. જોકે, એવો સાદો પ્રોગ્રામ કે જેને ઉપયોગકર્તા સાથે સંવાદની બહુ જરૂર ન હોય તેવા પ્રોગ્રામને SciTE એડિટરની અંદર જ અમલમાં મૂકી શકાય છે.

અપવાદરૂપ પરિસ્થિતિના વ્યવસ્થાપનના ફાયદા

આખા પ્રકરણમાં, આપણે જાવાપ્રોગ્રામમાં અપવાદરૂપ પરિસ્થિતિના વ્યવસ્થાપનના ઉપયોગનું સમર્થન કર્યું. હવે એ સ્પષ્ટ થવું જોઈએ કે, એક સારા પ્રોગ્રામનો અધવચ્ચે અચાનક અંત આવી જાય તેના કરતાં હંમેશાં અપેક્ષિત અપવાદોનું વ્યવસ્થાપન થાય તેમ કરવું જોઈએ. આપણા જાવાપ્રોગ્રામમાં અપવાદ-વ્યવસ્થાપનનો ઉપયોગ કરવાના ફાયદા ટૂંકમાં જોઈએ. જાવાપ્રોગ્રામમાં અપવાદ-વ્યવસ્થાપનનો ઉપયોગ કરવાના કેટલાક ફાયદા નીચે મુજબ છે :

- તે આપણને પ્રોગ્રામના સામાન્ય પ્રવાહને જાળવવામાં મદદરૂપ નીવડે છે. અપવાદ-વ્યવસ્થાપનની ગેરહાજરીમાં પ્રોગ્રામનો પ્રવાહ ક્યારેક અવરોધાય છે.
- તે સામાન્ય સૂચનાઓને બદલે અપવાદ-વ્યવસ્થાપન માટેની અલગ સૂચના લખવાની છૂટ આપે છે.
- ભૂલના પ્રકારોને એકજૂથમાં લાવી શકાય છે અને પ્રોગ્રામમાં અલગ તારવી શકાય છે.
- ક્લાયન્ટને પ્રોગ્રામ પૂરા પાડતાં પહેલાં, પ્રોગ્રામ ડીબગ થયેલ છે તેની ખાતરી આપી શકાય છે.
- પ્રોગ્રામનો અમલ કરતી વખતે અમલ દરમિયાન ઉદ્ભવતી વિવિધ ભૂલોને પકડી પાડવા સરળ તંત્ર-વ્યવસ્થા પૂરી પાડે છે.

સારાંશ

આ પ્રકરણમાં આપણે એ શીખ્યા કે, ઉદ્ભવેલી ભૂલને દર્શાવવા પ્રોગ્રામ અપવાદોનો ઉપયોગ કરી શકે છે. પ્રોગ્રામ try, catch અને finally વિભાગનાં સંયોજનનો ઉપયોગ કરીને અપવાદને શોધી શકે છે.

અપવાદને બહાર પાડવા માટે આપણે throw વિધાનનો ઉપયોગ કરીએ છીએ અને તેને અપવાદના ઑબ્જેક્ટ (java.lang.throwable નો સબક્લાસ) પણ પ્રદાન કરીએ છીએ. એ પદ્ધતિ કે જે ન શોધી શકાયેલ અપવાદ મોકલે છે, તેની ઘોષણામાં throws ઉપવાક્યનો સમાવેશ હોવો જ જોઈએ. try વિધાનમાં ઓછામાં ઓછો એક catch વિભાગ અથવા finally વિભાગ અને એક કરતાં વધુ catch વિભાગ હોઈ શકે. અમુક ઘટના ક્યારેય ન બનવી જોઈએ, તે ચકાસવા નિશ્ચિતપણે ઉપયોગ કરાય છે. તે પ્રોગ્રામર દ્વારા સુધારા-વધારા કરવા માટે ઉપયોગમાં લેવાય છે.

સ્વાધ્યાય

1. કંપાઈલેશન વખતની ભૂલ અને અમલ દરમિયાનની ભૂલ વચ્ચે શું તફાવત છે ?
2. અપવાદ એટલે શું ? જાવામાં મળતા કેટલાક અપવાદનાં ઉદાહરણ આપો.
3. જાવામાં અપવાદોનું વ્યવસ્થાપન કેવી રીતે કરવામાં આવે છે ?
4. try, catch અને finally વિભાગની ઉપયોગિતા શું છે ?
5. throw અને throws કી વર્ડની ઉપયોગિતા શું છે ?
6. જરૂરિયાત અનુસારનો અપવાદ તમે કેવી રીતે બનાવશો ?
7. નીચે આપેલા પ્રશ્નો માટે દરેકની સાથે આપેલા ચાર વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરી જવાબ આપો :

   (1) ઑબ્જેક્ટ આધારિત પ્રોગ્રામિંગ પરિભાષામાં નીચેના પૈકી કઈ ભૂલની સ્થિતિ (error condition) ગણાય છે ?

   (a) વિસંગતતા (anomaly) (b) ટૂંકાણ (abbreviation)

   (c) અપવાદ (exception) (d) ચલન (deviation)

(2) તમામ જાવા-અપવાદો માટે નીચેનામાંથી કયો શબ્દ સાચો છે ?

(a) ભૂલ (Errors) (b) અમલ દરમિયાનના અપવાદ (Run-time exceptions)

(c) બહાર પાડી શકાય તેવા (Throwables) (d) છોડી દઈ શકાય તેવા (Omissions)

(3) નીચેનામાંથી કયું વિધાન સાચું છે ?

(a) અપવાદો એ ભૂલ કરતાં વધુ ગંભીર છે.

(b) ભૂલ એ અપવાદો કરતાં વધુ ગંભીર છે.

(c) અપવાદો અને ભૂલો બન્ને એકસરખા ગંભીર છે.

(d) અપવાદો અને ભૂલ એક જ બાબત છે.

(4) નીચેનામાંથી કયું તત્ત્વ try વિભાગમાં સમાવવામાં આવતું નથી ?

(a) કી વર્ડ try (b) કી વર્ડ catch

(c) છગડિયો કૌંસ (d) ક્યારેક અપવાદ સર્જતાં વિધાનો

(5) અપવાદ સર્જાય ત્યારે નીચેનામાંથી કયો વિભાગ વ્યવસ્થાપન કરે છે કે યોગ્ય ક્રિયા હાથ ધરે છે ?

(a) try (b) catch

(c) throws (d) handles

(6) નીચેનામાંથી કયું catch વિભાગની અંદર હોવું જોઈએ ?

(a) finally વિભાગ (b) અપવાદનું વ્યવસ્થાપન કરતું એક વિધાન

(c) અપવાદનું વ્યવસ્થાપન કરવા જરૂરી બધાં વિધાન

(d) throws કી વર્ડ

(7) જ્યારે try વિભાગ કોઈ અપવાદ નહીં સર્જે અને તમે અનેક catch વિભાગ બનાવ્યા હશે, ત્યારે શું થશે ?

(a) તમામનો અમલ થશે.

(b) માત્ર બંધબેસતું પ્રથમ અમલમાં આવશે.

(c) એક પણ catch વિભાગ અમલમાં આવશે નહીં.

(d) માત્ર પ્રથમ catch વિભાગ અમલમાં આવશે.

(8) નીચેનામાંથી try...catch વિભાગનો ઉપયોગ કરવાનો ફાયદો કયો છે ?

(a) અપવાદરૂપ ઘટના દૂર કરવામાં આવે છે.

(b) અપવાદરૂપ ઘટના ઓછી કરવામાં આવે છે.

(c) અપવાદરૂપ ઘટનાઓને નિયમિત ઘટનાઓ સાથે સાંકળવામાં આવે છે.

(d) અપવાદરૂપ ઘટનાઓને, નિયમિત ઘટનાઓથી અલિપ્ત કરવામાં આવે છે.

(9) નીચેના પૈકી કઈ પદ્ધતિ (method) અપવાદને થ્રો કરી શકે ?

(a) throws ઉપવિધાન સાથેની પદ્ધતિ

(b) catch વિભાગ સાથેની પદ્ધતિ

(c) try વિભાગ સાથેની પદ્ધતિ

(d) finally વિભાગ સાથેની પદ્ધતિ

(10) કોઈ મેથડનો પૂર્ણ રીતે ઉપયોગ શક્ય છે કે કેમ તે જાણવા માટે નીચેના પૈકી ક્યું ઓછું અગત્યનું છે ?

(a) મેથડના પરિણામનો પ્રકાર

(b) મેથડને જરૂરી આર્ગ્યુમેન્ટનો પ્રકાર

(c) મેથડમાં રહેલ વિધાનોની સંખ્યા

(d) મેથડ દ્વારા થ્રો (throw) કરવામાં આવતા અપવાદોનો પ્રકાર

પ્રાયોગિક સ્વાધ્યાય

1. એક એવો જાવાપ્રોગ્રામ લખો કે જે ઍરેમાં એલિમેન્ટ ઉમેરવા "add()", નામની પદ્ધતિ (method)નો ઉપયોગ કરતો હોય, પદ્ધતિમાં એક try વિભાગ પણ ઉમેરો, જેથી કરીને પદ્ધતિ ArrayIndexOutOfBoundsExceptionને સંભાળી શકે.

2. ઉપર્યુક્ત ઉદાહરણમાં, try...catch વિભાગને પદ્ધતિમાંથી કાઢી નાંખો. "add()" પદ્ધતિને શરૂ કરનારી પદ્ધતિ throws ચાવીરૂપ શબ્દનો ઉપયોગ કરીને અપવાદનું વ્યવસ્થાપન કરી શકવી જોઈએ.

3. જ્યારે તમે કોઈ ઋણ સંખ્યાનું વર્ગમૂળ (square root) મેળવવા પ્રયત્ન કરતા હો, ત્યારે એક ગાણિતિક અપવાદ (ArithmeticException) થ્રો કરે અને ઝીલે (catch કરે) એવો જાવાપ્રોગ્રામ લખો. ઉપયોગકર્તાને કોઈ સંખ્યા દાખલ કરવા જણાવો અને એના ઉપર Math.sqrt() પદ્ધતિ (method)નો ઉપયોગ કરવાનો પ્રયત્ન કરો. આ વિનિયોગ વર્ગમૂળ દર્શાવશે અથવા અપવાદને ઝીલીને યોગ્ય સંદેશ દર્શાવશે. આ પ્રોગ્રામને SqrtException.java નામ આપી સાચવી દો.

4. એક એવો જાવાપ્રોગ્રામ લખો કે જે, જન્મતારીખની યથાર્થતા ચકાસે. આ માટે જરૂરિયાત પ્રમાણેનો અપવાદ "InvalidBirthDateException" બનાવો. ઉપયોગકર્તાને કોઈ પણ તારીખ દાખલ કરવા જણાવો. જો જન્મતારીખ હાલની તારીખ પછીની કોઈ તારીખ હોય તો "InvalidBirthDateException" નામનો અપવાદ બહાર પાડો (throw કરો). આ અપવાદને પાર પાડવા માટે યોગ્ય સૂચનાઓ (code) લખો.

5. એક એવો જાવાપ્રોગ્રામ લખો કે જે, બૅંકવ્યવહારોને ગોઠવે. balanceAmount અને withdrawAmount નામના બે ચલ લો. જો withdrawAmount એ balanceAmount કરતાં ઓછી હોય, તો પ્રોગ્રામ તરત જ ધ્યાન દોરે તેવું થવું જોઈએ.

6. એક એવો જાવાપ્રોગ્રામ લખો કે જે, બૅંકવ્યવહારોને ગોઠવે. balanceAmount અને withdrawAmount નામના બે ચલ લો. આ માટે ખાસ જરૂરિયાત માટેનો "InvalidTransaction" નામનો અપવાદ બનાવો. જો withdrawAmount એ balanceAmount કરતાં ઓછી હોય, તો તમારો પ્રોગ્રામ તરત જ "InvalidTransaction" નામનો અપવાદ થ્રો કરવો જોઈએ. આ અપવાદને પાર પાડવા માટે યોગ્ય સૂચનાઓ (Exception handlers) લખો.

7. એક એવો જાવાપ્રોગ્રામ લખો કે જે, જન્મતારીખની યથાર્થતા ચકાસે. આ માટે ખાસ જરૂરિયાત અનુસારના "InvalidDateException", "InvalidMonthException" અને "InvalidYearException" નામના ત્રણ અપવાદ બનાવો. જો તારીખ ઋણ હોય અથવા 30 કરતાં વધુ હોય, તો "InvalidDateException" અપવાદ થ્રો કરો. જો મહિનો ઋણ હોય અથવા 12 કરતાં વધુ હોય, તો "InvalidMonthException" અપવાદ થ્રો કરો. જો વર્ષ 1950 અથવા હાલના વર્ષ પછીનું વર્ષ હોય, તો "InvalidYearException" અપવાદ થ્રો કરો.

ફાઇલ-વ્યવસ્થાપન 11

આ પ્રકરણમાં આપણે જાવાપ્રોગ્રામ દ્વારા હાર્ડડિસ્ક પરની જુદી-જુદી ફાઇલ અને ડિરેક્ટરીને કેવી રીતે ઉપયોગમાં લેવી, ઓળખવી અને તેનું સંવર્ધન કરવું તે જોઈશું. સમગ્ર પ્રકરણ દરમિયાન આપણે ઇનપુટ/આઉટપુટ પ્રક્રિયા અને ફાઇલ-વ્યવસ્થાપન પર ધ્યાન કેન્દ્રિત કરીશું. આપણે java.io પૅકેજમાં ઉપલબ્ધ સૌથી સામાન્ય ક્લાસ (classes) અને java.util પૅકેજના થોડા ક્લાસનો ઉપયોગ કરીશું. તો હવે, ફાઇલ અને ડિરેક્ટરીનાં વિહંગાવલોકન (overview) સાથે તેની ચર્ચા શરૂ કરીએ.

કમ્પ્યૂટર ફાઇલને સમજીએ :

કમ્પ્યૂટર સિસ્ટમનાં સંગ્રહ-સાધનોને નાશવંત સંગ્રહ (Volatile Storage) અને અવિનાશી સંગ્રહ (Non-volatile Storage) એમ બે વિભાગમાં વહેંચવામાં આવે છે.

નાશવંત સંગ્રહ એ કામચલાઉ છે. જ્યારે કમ્પ્યૂટર બંધ થાય છે, ત્યારે ચલમાં સાચવેલી માહિતી નાશ પામે છે. જાવા-પ્રોગ્રામ કે જે, ચલમાં કિંમતો સાચવે છે, તે રેન્ડમ ઍક્સેસ મેમરી (RAM)નો ઉપયોગ કરે છે. સામાન્ય રીતે, ચલ, ઑબ્જેક્ટ અને તેના અનુસંધાન RAM માં સાચવવામાં આવે છે. એક વાર પ્રોગ્રામનો અંત આવે અથવા કમ્પ્યૂટર બંધ થાય ત્યારે ડેટા નાશ પામે છે.

અવિનાશી સંગ્રહ એ કાયમી સંગ્રહ છે; જ્યારે કમ્પ્યૂટરને મળતો વીજપ્રવાહ બંધ થાય, ત્યારે પણ ડેટા નાશ પામતો નથી. જાવાપ્રોગ્રામ જ્યારે ડિસ્ક ઉપર સાચવવામાં આવે છે, ત્યારે આપણે કાયમી સંગ્રહનો ઉપયોગ કરીએ છીએ. કમ્પ્યૂટર ફાઇલ એ અવિનાશી સાધન પર સંગ્રહવામાં આવતો ડેટાસમૂહ છે. ફાઇલો હાર્ડડિસ્ક, USB ડ્રાઇવ, ઑપ્ટિકલ ડિસ્ક અને કૉમ્પેક્ટ ડિસ્ક જેવાં કાયમી સંગ્રહસાધનો પર અસ્તિત્વ ધરાવે છે. ફાઇલમાં સંગ્રહવામાં આવતો ડેટા ઘણી વાર સતત ડેટા (persistent data) તરીકે પણ ઓળખાય છે.

ફાઇલોને વળી પાછી ટેક્સ્ટફાઇલ અને દ્વિઅંકી (Binary) ફાઇલ એમ મુખ્યત્વે બે ભાગમાં વહેંચી શકાય.

ટેક્સ્ટફાઇલ એવો ડેટા ધરાવે છે કે જે કોઈ ટેક્સ્ટ એડિટર દ્વારા વાંચી શકાય છે, કારણકે, ડેટા ASCII અથવા Unicode જેવી પદ્ધતિના ઉપયોગ દ્વારા સંજ્ઞાંકિત કરાયેલ હોય છે. ટેક્સ્ટફાઇલ ડેટાફાઇલ હોઈ શકે છે, જે હકીકતો ધરાવતી હોય, જેવી કે, પે-રોલ ફાઇલ જે કર્મચારી ક્રમ, નામ અને પગાર સમાવે છે; અથવા કેટલીક ટેક્સ્ટફાઇલ એ પ્રોગ્રામફાઇલ અથવા વિનિયોગ-ફાઇલ પણ હોઈ શકે, જે સૉફ્ટવેર માટેની સૂચનાઓ સાચવી શકે છે. gedit, vi, pico જેવા એડિટર દ્વારા તૈયાર કરાયેલી ફાઇલો એ ટેક્સ્ટફાઇલનાં ઉદાહરણ છે. તેમનાં અનુલંબન તરીકે txt, java અથવા c હોઈ શકે છે.

દ્વિઅંકી ફાઇલ એવો ડેટા ધરાવે છે, જે લખાણ તરીકે સંજ્ઞાંકિત (Encoded) કરાયો નથી. તેનું લખાણ દ્વિઅંકી સ્વરૂપે હોય છે, જેનો અર્થ એ છે કે, ડેટાને બાઇટ સ્વરૂપે ઉપયોગમાં લેવાય છે. દ્વિઅંકી ફાઇલનાં કેટલાંક ઉદાહરણરૂપ અનુલંબનો .jpeg, .mp3 અને .class છે.

જાવા ભાષા વિવિધ ક્રિયાઓને સમર્થન આપે છે, જે ફાઇલ અથવા ડિરેક્ટરી પર કરી શકાય છે. જાવાપ્રોગ્રામના ઉપયોગ દ્વારા ફાઇલ પર કરી શકાતી કેટલીક ક્રિયાઓની યાદી નીચે મુજબ છે :

- ફાઇલ કે ડિરેક્ટરીનો પથ નક્કી કરવો.
- ફાઇલ ખોલવી.
- ફાઇલમાં લખવું.
- ફાઇલમાંથી વાંચવું.

- ફાઈલને બંધ કરવી.
- ફાઈલને કાઢી નાંખવી.
- ફાઈલના ગુણધર્મો (attributes)ની પૃચ્છા કરવી.

જાવા એવા તૈયાર આંતરપ્રસ્થાપિત ક્લાસ (inbuilt classes) આપે છે, જેમાં આવાં બધાં કામો પાર પાડવા આપણને મદદરૂપ થવા માટેની પદ્ધતિ (method) આપેલી જ હોય છે. આ ક્લાસ java.io પૅકેજમાં ઉપલબ્ધ હોય છે. જાવાસ્ટ્રીમ (streams)ના ખ્યાલનો ઉપયોગ કરે છે અને તે બાઇટ્સ અને અક્ષરો પર I/O ક્રિયાઓને પાર પાડવા બે જુદી-જુદી કક્ષાના જાવા- ક્લાસ પૂરા પાડે છે. ફાઈલમાંથી માહિતી વાંચવી અને ફાઈલમાં માહિતી લખવાની વિસ્તૃત ચર્ચા આગળના વિભાગમાં કરવામાં આવશે.

જાવામાં ફાઈલ-ક્લાસ (File Class in Java)

java.io.File ક્લાસમાં ફાઈલ કે ડિરેક્ટરીની લાક્ષણિકતા વિશેની માહિતીનો સમાવેશ કરવામાં આવે છે. ફાઈલ કે ડિરેક્ટરીઓના ગુણધર્મો (ઍટ્રિબ્યૂટ્સ) મેળવવા ફાઈલ-ક્લાસનો ઉપયોગ કરી શકાય. આપણે ફાઈલ કે ડિરેક્ટરીને create/ rename/delete કરી શકીએ છીએ. આપણે ફાઈલના વિવિધ ગુણધર્મો પણ જાણી શકીએ. જેવા કે, ફાઈલના ઉપયોગની પરવાનગી, ફાઈલનું કદ અથવા ફાઈલમાં છેલ્લે ક્યારે સુધારા-વધારા કરવામાં આવ્યા હતા તે સમય વગેરે. ફાઈલ-ક્લાસ સાથે સંબંધ ધરાવતો હોય તેવા ફાઈલ-ઑબ્જેક્ટનું સર્જન કરવાનો ગર્ભિત અર્થ એ થતો નથી કે, તે ફાઈલ કે ડિરેક્ટરી અસ્તિત્વ ધરાવે છે. એક ફાઈલ-ઑબ્જેક્ટમાં હાર્ડડિસ્ક પરની ફાઈલ કે ડિરેક્ટરીનું પથનામ અથવા સરનામું મૂકવામાં આવે છે.

ફાઈલ અથવા ડિરેક્ટરી પર વિવિધ પ્રક્રિયાઓ કરવા માટે ઉપયોગમાં લઈ શકાય, તેવી ફાઈલ-ક્લાસની આશરે 30 પદ્ધતિઓ છે. જો કે તેમ છતાં, ફાઈલમાંથી વાંચવા માટે કે ફાઈલમાં લખવા માટે ફાઈલ-ક્લાસ કોઈ પદ્ધતિ પૂરી પાડતા નથી. આવી ક્રિયાઓ પાર પાડવા માટે ઘણાં બધા 'સ્ટ્રીમ-ક્લાસ' (પ્રવાહક્લાસ) ઉપલબ્ધ છે. તો ચાલો, આપણે ફાઈલ-ક્લાસના વધુ ઉપયોગમાં લેવાતા સર્જકો (constructors) અને પદ્ધતિઓ (methods) વિશે શીખવાની શરૂઆત કરીએ.

ફાઈલ-ક્લાસના સર્જકો (Constructors of File Class)

ફાઈલ-ક્લાસના ઉપયોગ દ્વારા આપણે, કોઈ પણ ફાઈલને તેનો સંબંધિત પથ આપીને અથવા શબ્દમાળા સ્વરૂપે તેનો સંપૂર્ણ પથ આપીને અનુસંધાન આપી શકીએ. કોઈ ફાઈલ અથવા ડિરેક્ટરીનો ઉલ્લેખ કરવા ફાઈલ-ક્લાસ નીચે મુજબના સર્જકો પૂરા પાડે છે :

```
File(String path)
File(String directory_path, String file_name)
File(File directory, String file_name)
```

હવે, ઉપર્યુક્ત સર્જકો માટે એક ઉદાહરણ લઈએ. લિનક્સમાં "/etc" ડિરેક્ટરીમાં ઉપલબ્ધ "passwd" ફાઈલ, સિસ્ટમમાં હયાત ઉપયોગકર્તાની માહિતી સાચવે છે. ધારો કે, આપણે તેના ગુણધર્મો (attributes) દર્શાવવા હોય, તો નીચે દર્શાવેલ ત્રણ રીતના ઉપયોગ દ્વારા તેના જાવાફાઈલ ઑબ્જેક્ટ બનાવી શકાય :

- File fileobj = new File("/etc/passwd"); એવો પથ દર્શાવીને.
- ડિરેક્ટરી અને ફાઈલનામને બે જુદા-જુદા આર્ગ્યુમૅન્ટ દર્શાવીને

  ```
  File fileobj = new File("/etc", "passwd");
  ```
- dirobj ઑબ્જેક્ટમાં મૂકેલી ડિરેક્ટરીના અનુસંધાન વાપરીને

  ```
  File dirobj = new File("/etc");
  File fileobj = new File(dirobj, "passwd");
  ```

ફાઈલ-ક્લાસની પદ્ધતિઓ

કોષ્ટક 11.1 ફાઇલ-ક્લાસની વધુ ઉપયોગમાં લેવાતી પદ્ધતિઓનો સારાંશ રજૂ કરે છે.

પદ્ધતિ (Method)	વર્ણન (Description)
boolean exists()	જો ફાઇલ કે ડિરેક્ટરી હયાત હશે, તો true કિંમત પરત કરશે અન્યથા false કિંમત પરત કરે છે.
boolean isFile()	જો ફાઇલ હયાત હશે, તો true કિંમત પરત કરશે અન્યથા false કિંમત પરત કરે છે.
boolean isDirectory()	જો ડિરેક્ટરી હયાત હશે, તો true કિંમત પરત કરશે અન્યથા false કિંમત પરત કરે છે.
boolean isHidden()	જો ફાઇલ કે ડિરેક્ટરી છુપાયેલી હશે, તો true કિંમત પરત કરે છે.
String getAbsolutePath()	ફાઇલ કે ડિરેક્ટરીનો સંપૂર્ણ પથ (absolute path) પરત કરે છે.
String getName()	ઑબ્જેક્ટ દ્વારા સંબોધાયેલ ફાઇલ કે ડિરેક્ટરીનું નામ પરત કરે છે.
String getPath()	ફાઇલ કે ડિરેક્ટરીનો પથ (path) પરત કરે છે.
long length()	તે ફાઇલમાં રહેલા બાઇટ્સની સંખ્યા પરત કરે છે.
String[] list()	ડિરેક્ટરીમાં ઉપલબ્ધ ફાઇલો અને ડિરેક્ટરીઓનાં નામ પરત કરે છે.
File[] listFiles()	ડિરેક્ટરીમાં ફાઇલ સૂચવતા પથનામના સારાંશ (abstract pathnames)નો ઍરે પરત કરે છે.

કોષ્ટક 11.1 : ફાઈલ-ક્લાસની વધુ ઉપયોગમાં આવતી પદ્ધતિઓ

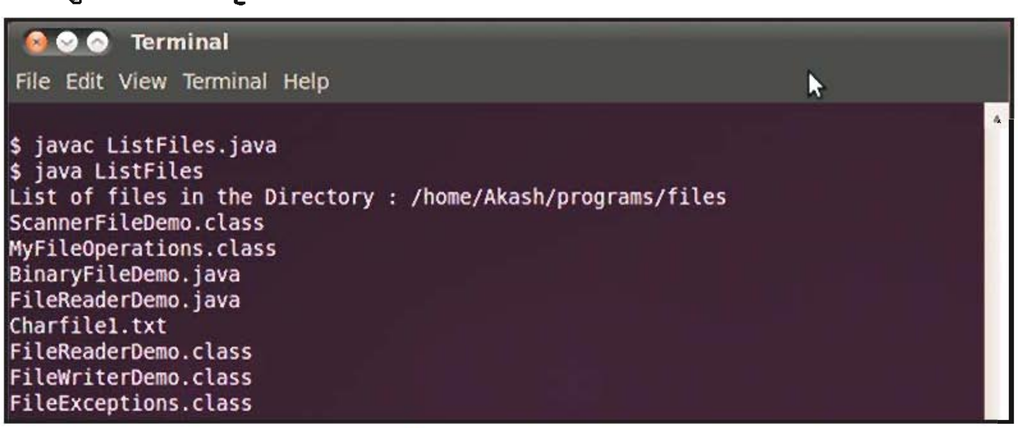
હવે કોઈ આપેલી ડિરેક્ટરીમાં ઉપલબ્ધ ફાઇલોનાં નામની યાદી આપતા પ્રોગ્રામને સમજવાનો પ્રયત્ન કરીએ. કોઈ ચોક્કસ ડિરેક્ટરીનો ઉલ્લેખ કરતા ફાઇલ-ક્લાસનો ઑબ્જેક્ટ બનાવ્યા પછી, આપણે તે ડિરેક્ટરીમાં ઉપલબ્ધ તમામ ફાઇલોની યાદી મેળવવા list() પદ્ધતિનો ઉપયોગ કરી શકીએ. કોડલિસ્ટિંગ 11.1માં આપેલ પ્રોગ્રામ "/home/Akash/programs/files" ડિરેક્ટરીમાં ઉપલબ્ધ ફાઇલોની યાદી મેળવવાનું નિદર્શન કરે છે. કોઈ ચોક્કસ ડિરેક્ટરી માટે ફાઇલો અને ડિરેક્ટરીઓની સંખ્યા ગણવા માટે પ્રોગ્રામમાં આપણે એક ચલનો ઉપયોગ કરેલ છે તેમજ ડિરેક્ટરીમાં ઉપલબ્ધ ફાઇલ-ઑબ્જેક્ટને સાચવવા ફાઇલ-ક્લાસનો listOfFiles નામનો ઍરેનો ઉપયોગ કરાયો છે.

```
//List the contents of a Directory
import java.io.File;
public class ListFiles
{
        public static void main(String args[])
        {
                //Provide a Directory Path
                String path = "/home/Akash/programs/files";
                String files;
                int countOfFiles;
                try
                {
```

```
                File folder = new File(path);
                //Store the list of files in an array of File[] objects
                File[] listOfFiles = folder.listFiles();
                //count the number of files in the folder
                countOfFiles = listOfFiles.length;

                System.out.print("List of files in the Directory : ");
                System.out.println(folder.getAbsolutePath());

                //Iterate to display the name of each file
                for(int i=0; i<countOfFiles; i++)
                {
                    if(listOfFiles[i].isFile())
                    {
                        files = listOfFiles[i].getName();
                        System.out.println(files);
                    }
                }
            }
            catch(Exception eobj)
            {
                System.out.println(eobj);
            }
        }
    }
```

કોડલિસ્ટિંગ 11.1 : આપેલ ડિરેક્ટરીમાં ઉપલબ્ધ ફાઈલોની યાદી મેળવવાનો પ્રોગ્રામ

કોડલિસ્ટિંગ 11.1નું પરિણામ આકૃતિ 11.1માં દર્શાવેલ છે.



11.1 : કોડલિસ્ટિંગ 11.1નું પરિણામ

નોંધ :

ઉપર્યુક્ત પ્રોગ્રામ "/home/Akash/programs/files" ડિરેક્ટરીમાં હયાત ફાઈલ અને ડિરેક્ટરીઓની યાદી રજૂ કરે છે. પરિણામ નિહાળવા માટે આપણે યોગ્ય પથ અને ફાઈલના નામનો ઉપયોગ કરી શકીએ.

સ્ટ્રીમ (Stream)નો પરિચય

ફાઈલને કેવી રીતે સુધારવી કે ફાઈલના લખાણને કેવી રીતે દર્શાવવું તે અત્યાર સુધી આપણે શીખ્યા નથી. આવી ક્રિયાઓ કરવા માટે આપણે સ્ટ્રીમ (stream)ના ખ્યાલને સમજવો પડશે. ફાઈલમાં લખવા કે ફાઈલમાંથી વાંચવાની ક્રિયા પાર પાડવા માટે જાવા સ્ટ્રીમ-ક્લાસનો ઉપયોગ કરે છે.

ઈનપુટ અને આઉટપુટ માટે વપરાતાં સાધનોના પ્રકાર વિશે આપણે અગાઉ શીખી જ ગયા છીએ. ઉદાહરણ તરીકે કી-બોર્ડ એ એક ઈનપુટ માટેનું સાધન છે, જ્યારે મૉનિટર એ આઉટપુટ માટેનું સાધન છે. હાર્ડડિસ્કમાં સાચવેલ ફાઈલમાંથી આપણે ડેટા વાંચી પણ શકીએ છીએ અને તેમાં લખી પણ શકીએ છીએ, તે કારણથી હાર્ડડિસ્કને ઈનપુટ તેમજ આઉટપુટ એમ બંને માટેનાં સાધન તરીકે વર્ગીકૃત કરી શકાય. વિવિધ ક્ષમતા અને વિવિધ ઉત્પાદકો તરફથી બજારમાં અનેક સાધનો ઉપલબ્ધ છે. ઉદાહરણ તરીકે, હાર્ડડિસ્ક અનેક ઉત્પાદકો દ્વારા બને છે અને 500GB અથવા 1 TB એમ જુદી-જુદી સંગ્રહક્ષમતાવાળી હોય છે. આ ઉપરાંત હાર્ડડિસ્કને USB અથવા SATA જેવા જુદાં-જુદાં કેબલ વડે જોડી શકાય છે. તેમ છતાં, ફાઈલોમાં ડેટા લખવા કે વાંચવાની ક્રિયા કરવાનો પ્રોગ્રામ લખતી વખતે જાવાના પ્રોગ્રામરે હાર્ડડિસ્કનો પ્રકાર કે તેની સંગ્રહક્ષમતા જેવી બાબતો વિશે ચિંતા કરવાની જરૂર નથી. આવું એટલા માટે શક્ય બને છે, કારણકે જાવા સ્ટ્રીમની કાર્યપદ્ધતિ પૂરી પાડે છે.

સ્ટ્રીમ એ ડેટા માટે મૂળ કે ગંતવ્ય તરીકે ઉપયોગમાં લેવાતા ઈનપુટ કે આઉટપુટ સાધનની ટૂંકી રજૂઆત છે. સ્ટ્રીમને આપણે પ્રોગ્રામમાં આવતા કે પ્રોગ્રામમાંથી જતા બાઈટની હારમાળા તરીકે કલ્પી શકીએ. આપણે સ્ટ્રીમનો ઉપયોગ કરીને ડેટાને લખી શકીએ કે વાંચી શકીએ.

જ્યારે આપણે સ્ટ્રીમમાં ડેટા લખતા હોઈએ, ત્યારે તે સ્ટ્રીમને આઉટપુટ-સ્ટ્રીમ (Output stream) કહે છે. આઉટપુટ-સ્ટ્રીમ પ્રોગ્રામમાંથી હાર્ડડિસ્ક પર ફાઈલમાં અથવા મૉનિટર પર અથવા નેટવર્કના કોઈ એક કમ્પ્યૂટર પર ડેટાને તબદીલ કરી શકે છે. પ્રોગ્રામ માટે બાહ્ય સાધન પરથી ડેટા વાંચવા માટે ઈનપુટ-સ્ટ્રીમ (Input stream) ઉપયોગમાં લેવાય છે. તે કી-બોર્ડ પરથી અથવા હાર્ડડિસ્ક પરની ફાઈલમાંથી ડેટાને પ્રોગ્રામમાં તબદીલ કરી શકે છે.

ઈનપુટ કે આઉટપુટ ક્રિયાઓ માટે સ્ટ્રીમનો ઉપયોગ કરવા પાછળનું મુખ્ય કારણ, આપણા પ્રોગ્રામને ઉપયોગમાં લેવાનારાં સાધનોથી સ્વતંત્ર બનાવવાનો છે. સ્ટ્રીમના બે મુખ્ય લાભ નીચે મુજબ છે :

- સાધનોની ટેક્નિકલ વિગતો વિશે જાણવાની પ્રોગ્રામરને ચિંતા કરવાની જરૂર રહેતી નથી.
- પ્રોગ્રામની મૂળ સૂચનાઓ (સૉર્સકોર્ડ)માં કોઈ પણ સુધારા કર્યા વિના પ્રોગ્રામ વિવિધ પ્રકારનાં ઈનપુટ/આઉટપુટ સાધનો માટે કામ કરી શકે છે.

જાવામાં સ્ટ્રીમ-ક્લાસ

બાઈટ-સ્ટ્રીમ (દ્વિઅંકી પ્રવાહ) / અને કૅરેક્ટર-સ્ટ્રીમ (શબ્દપ્રવાહ)ને સમજવા માટે ચાલો સૌપ્રથમ આપણે બાઈટ અને અક્ષરની રજૂઆત વચ્ચેનો ભેદ સમજવાનો પ્રયત્ન કરીએ. ચાલો, આપણે સંખ્યા "5"નું ઉદાહરણ લઈએ; કોષ્ટક 11.2માં દર્શાવ્યા મુજબ આપણે આ સંખ્યાને બે જુદી-જુદી રીતે રજૂ કરી શકીએ :

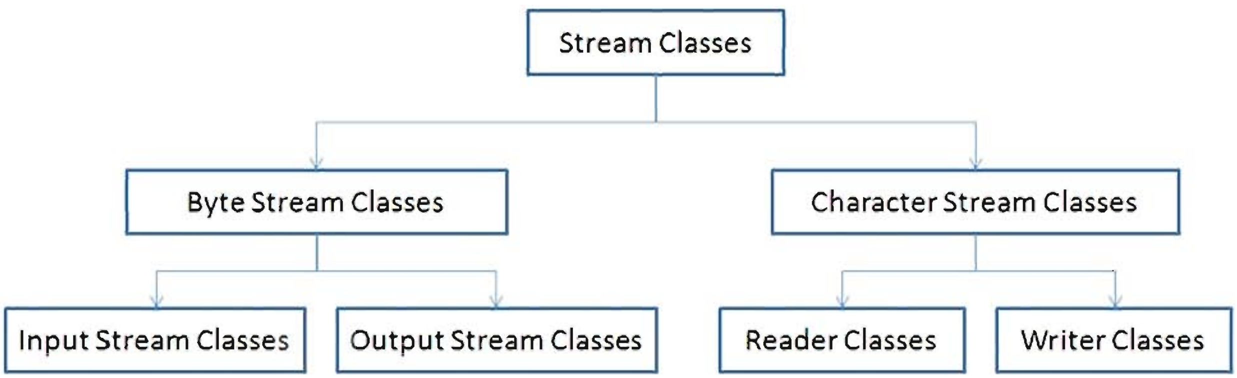
રજૂઆત	વિગત	બાઈનરી રજૂઆત
અક્ષર 5	ASCII કિંમત : 53	00110101
બાઈનરી અંક 5	બાઈનરી કિંમત : 5	00000101

કોષ્ટક 11.2 : અક્ષર અને બાઈનરી રજૂઆત

કોઈ પણ અક્ષરને સામાન્ય રીતે ASCII અથવા યુનિકોડ (Unicode) સ્વરૂપે સાચવવામાં આવે છે. પરંતુ, જ્યારે તેને ગણતરીના ઉદ્દેશ્યથી વાપરવામાં આવે છે, ત્યારે તેની બાઇનરી કિંમત અર્થપૂર્ણ બને છે. એક વધુ ઉદાહરણ લઈએ, int i = 32 વિધાન 'i' નામના ચલને પૂર્ણાંક પ્રકારના ચલ તરીકે ઘોષિત કરે છે, જેમાં 32 કિંમત સચવાય છે. જોકે 32ને '3' અને '2' એમ બે જુદા-જુદા અક્ષરો તરીકે રજૂ કરી શકાય. જ્યારે આપણે "Human beings have 32 teeth" એવું વાક્ય લખતા હોઈએ, ત્યારે શાબ્દિક રજૂઆત સલાહભરી છે, જ્યાં આપણે સંખ્યા ઉપર કોઈ પણ પ્રકારની ક્રિયાઓ કરતાં નથી. પરંતુ, જ્યારે આપણે ગાણિતિક ક્રિયાઓ કરવી હોય, ત્યારે આપણે int, float અથવા double જેવા ડેટાપ્રકાર (datatype)નો ઉપયોગ કરીએ છીએ, જે આપણને સંખ્યા દ્વિઅંકી સ્વરૂપે સાચવવા દે છે.

જાવા બે પ્રકારની સ્ટ્રીમને માન્યતા આપે છે, બાઇટ-સ્ટ્રીમ (byte stream) અને કેરેક્ટર-સ્ટ્રીમ (character stream). એવી સ્ટ્રીમ કે જે, ડેટાને ફાઈલમાં કે કોઈ સાધન તરફ બાઇટ સ્વરૂપે તબદીલ કરે તેને બાઇટ-સ્ટ્રીમ અથવા બાઇનરી સ્ટ્રીમ કહે છે. બાઇટ-સ્ટ્રીમના ઉપયોગથી તૈયાર થતી ફાઈલોને બાઇનરી ફાઈલ તરીકે ઓળખવામાં આવે છે. ધારો કે, જો આપણે ફાઈલમાં પૂર્ણાંક, ડબલ (double) અથવા બુલિયન (boolean) જેવા ચલ સાચવવા ઇચ્છતા હોઈએ, તો આપણે બાઇનરી ફાઈલ જ ઉપયોગમાં લેવી પડે. એરે કે ઑબ્જેક્ટ સાચવવા માટે પણ બાઇનરી ફાઈલનો ઉપયોગ કરી શકાય છે. એ જ રીતે, ટેક્સ્ટફાઈલો અને પ્રોગ્રામકોડ, બનાવવા કેરેક્ટર-સ્ટ્રીમનો ઉપયોગ કરાય છે. vi અથવા SciTE જેવા ટેક્સ્ટ એડિટરમાં તેને ખોલી શકાય છે.

જાવાક્લાસના બે ગણ પૂરા પાડે છે, કેરેક્ટર-સ્ટ્રીમ-ક્લાસ અને બાઇનરી સ્ટ્રીમ-ક્લાસ. java.io પૅકેજમાં ઉપલબ્ધ કેરેક્ટર-સ્ટ્રીમ-ક્લાસ કેરેક્ટર/ડેટા સાથે કામ કરે છે, જ્યારે java.io પૅકેજમાં રહેલ બાઇટ-સ્ટ્રીમ-ક્લાસ બાઇનરી ડેટા સાથે કામ કરે છે. આકૃતિ 11.2 સ્ટ્રીમ-ક્લાસનું વર્ગીકરણ દર્શાવે છે.



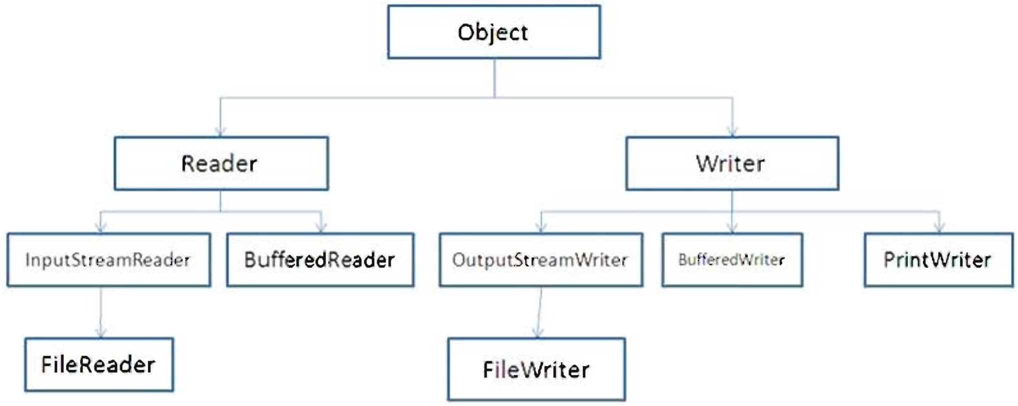
11.2 : સ્ટ્રીમ-ક્લાસનું વર્ગીકરણ

જાવા-સ્ટ્રીમને ઇનપુટ-સ્ટ્રીમ અને આઉટપુટ-સ્ટ્રીમ નામના બે મૂળ પ્રકારમાં વર્ગીકૃત કરી શકાય. ઇનપુટ-સ્ટ્રીમ ઉદ્‌ગમ (ફાઈલ, કી-બોર્ડ)માંથી ડેટા વાંચે છે, જ્યારે આઉટપુટ-સ્ટ્રીમ ગંતવ્યસ્થાન (ફાઈલ, આઉટપુટ સાધનો) પર ડેટા લખે છે.

java.io પૅકેજ સ્ટ્રીમ-ક્લાસનો એવો સમૂહ ધરાવે છે, જે ફાઈલમાં લખવાની અને વાંચવાની ક્રિયાને શક્ય બનાવે છે. આ ક્લાસનો ઉપયોગ કરવા માટે પ્રોગ્રામ દ્વારા java.io પૅકેજને આયાત (import) કરવું જરૂરી છે. આમ તો, કેરેક્ટર અને બાઇટ પર ક્રિયા કરવા માટે ઘણા ક્લાસ છે. જોકે આપણે અહીં સૌથી વધુ ઉપયોગમાં આવતા ક્લાસની જ ચર્ચા કરીશું. કારણ કે java.io પૅકેજમાં આવતા દરેક ક્લાસની ચર્ચા કરવી એ આ પાઠ્યપુસ્તકની મર્યાદાબહાર છે.

કેરેક્ટર-સ્ટ્રીમ-ક્લાસ

કેરેક્ટર-સ્ટ્રીમ-ક્લાસ એ java.io પૅકેજમાં ઉપલબ્ધ ક્લાસનો સમૂહ છે. તેને 16-bit યુનિકોડ અક્ષરો વાંચવા અને લખવા માટે ઉપયોગમાં લઈ શકાય છે. કેરેક્ટર-સ્ટ્રીમ-ક્લાસને વળી પાછા રીડર (Reader) અને રાઇટર (Writer) ક્લાસ એમ બે ભાગમાં વર્ગીકૃત કરી શકાય છે. રીડર-ક્લાસ (Reader class) એ ફાઈલમાંથી અક્ષરો વાંચવા માટે તૈયાર કરેલા ક્લાસનો સમૂહ છે. જ્યારે, રાઇટર-ક્લાસ (Writer class) એ ફાઈલમાં અક્ષરો લખવા માટે તૈયાર કરેલા ક્લાસનો સમૂહ છે. કેરેક્ટર-સ્ટ્રીમ-ક્લાસનો પદાનુક્રમ આકૃતિ 11.3માં દર્શાવેલ છે.



11.3 : કૅરેક્ટર-સ્ટ્રીમ-ક્લાસનો પદાનુક્રમ

આકૃતિ 11.3માં દર્શાવ્યા મુજબ, java.io.Reader ક્લાસ અને java.io.Writer ક્લાસ, ઓબ્જેક્ટ ક્લાસમાંથી ઉદ્ભવ્યા છે. તે એબ્સ્ટ્રૅક્ટ ક્લાસ છે (એવા ક્લાસ કે જે કોઈ ઓબ્જેક્ટ બનાવવા ઉપયોગમાં લઈ શકાતા નથી) અને તેના સબ-ક્લાસ દ્વારા અમલમાં મૂકવાની પદ્ધતિઓના ગણ સાથે મળે છે. InputStreamReader અને BufferedReader એ Reader Classના સબ-ક્લાસ (પેટા ક્લાસ) છે. FileReader ક્લાસ InputStreamReader ક્લાસનો સબ-ક્લાસ છે. એ જ રીતે, OutputStreamWriter, BufferedWriter અને PrintWriter એ Writer ક્લાસના સબ-ક્લાસ છે. FileWriter ક્લાસ એ OutputStreamWriter ક્લાસનો સબ-ક્લાસ છે.

હવે, આપણે ઉપર દર્શાવેલ કેટલાક ક્લાસના કન્સ્ટ્રક્ટર અને પદ્ધતિઓ સમજીએ. પદ્ધતિઓ અને કન્સ્ટ્રક્ટરનું વિગતવાર વર્ણન ઓનલાઇન http://docs.oracle.com/javase/6/docs/api/ પરથી મેળવી શકાય છે.

Writer ક્લાસ

કૅરેક્ટર-સ્ટ્રીમ લખવા માટે રાઇટર ક્લાસ એ મૂળ આધારરૂપ ક્લાસ છે. એબ્સ્ટ્રૅક્ટ રાઇટર-ક્લાસ ચોક્કસ કાર્યપ્રણાલી ઘોષિત કરે છે, જે તમામ કૅરેક્ટર આઉટપુટ-સ્ટ્રીમ માટે ઉપલબ્ધ બને છે. આપણે ફાઇલમાં લખવા માટેની ક્રિયા કરવા આપણા પ્રોગ્રામમાં FileWriter ક્લાસનો ઉપયોગ કરીશું.

રાઇટર-ક્લાસની પદ્ધતિ IOException થ્રો કરે. જ્યારે કોઈ I/O ક્રિયા નિષ્ફળ જાય, ત્યારે IOException બને છે. IOException એ ચકાસેલ અપવાદ છે, માટે આપણે તેની કાળજી લેવી જ પડે, અન્યથા કંપાઇલિંગ ભૂલ દર્શાવશે. કોષ્ટક 11.3 રાઇટર-ક્લાસની કેટલીક પદ્ધતિઓની યાદી રજૂ કરે છે. આ પદ્ધતિઓ તેના સબ-ક્લાસ દ્વારા ઉપયોગમાં લેવાય છે.

પદ્ધતિ (Method)	વર્ણન (Desctiption)
void close()	સ્ટ્રીમ (stream)ને બંધ કરે છે.
void write(int c)	સ્ટ્રીમમાં 'c'ના પાછળના 16 બીટ લખે છે.
void write(String s)	સ્ટ્રીમમાં શબ્દમાળા તરીકે 's' લખે છે.

કોષ્ટક 11.3 : FileWriter ક્લાસની કેટલીક પદ્ધતિઓ

OutputStreamWriter ક્લાસ રાઇટર ક્લાસને વિસ્તારે છે. તે અક્ષરોની સ્ટ્રીમને બાઇટની સ્ટ્રીમમાં પરિવર્તિત કરે છે. FileWriter ક્લાસ, OutputStreamWriterને વિસ્તારે છે અને ફાઇલમાં અક્ષરો પરિણામ રૂપે આપે છે. તેના કન્સ્ટ્રક્ટરોમાંના કેટલાક નીચે મુજબ છે :

FileWriter(String filepath) throws IOException

FileWriter(File fileobj) throws IOException

FileWriter(String filepath, boolean append) throws IOException

ઉપર્યુક્ત કન્સ્ટ્રક્ટરમાં filepath પેરામીટર એ ફાઇલનું પૂરું પથનામ છે અને fileobj એ ફાઇલક્લાસ ઑબ્જેક્ટ છે, જે ફાઇલને વર્ણવે છે. છેલ્લા કન્સ્ટ્રક્ટરમાં જો appendની કિંમત true મળશે, તો ફાઇલના છેડે અક્ષરો જોડવામાં આવશે, અન્યથા લખાણ ફાઇલના હાલના લખાણની ઉપર લખાઈ જશે. ઉદાહરણ તરીકે, FileWriterનો ઑબ્જેક્ટ આપણે નીચે મુજબ બનાવી શકીએ :

FileWriter fwobject = new FileWriter("/java/files/Charfile1.txt");

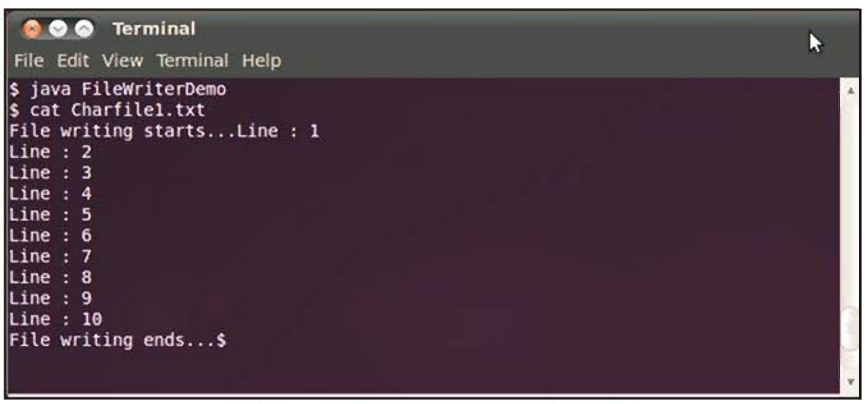
એક એવું ઉદાહરણ લઈએ જે ફાઇલમાં કેવી રીતે લખવું તે વર્ણવે. આપણે એવું અનુમાન કરીશું કે, ફાઇલો હાલની ડિરેક્ટરીમાં સચવાય છે. કોડલિસ્ટિંગ 11.2 એ દર્શાવે છે કે, "Charfile1.txt" ફાઇલ કેવી રીતે બનાવવી જે હયાત જ નથી. તદ્ઉપરાંત, આપણે write() પદ્ધતિનો ઉપયોગ કરીને તે ફાઇલમાં કેટલીક માહિતી લખીશું. આ પ્રોગ્રામમાં ઉપયોગમાં લેવાતી write() અને close() જેવી પદ્ધતિ રાઇટર ક્લાસમાંથી વારસામાં મેળવાયેલ છે. આ કોડલિસ્ટિંગનું પરિણામ આકૃતિ 11.4માં દર્શાવેલ છે. પ્રોગ્રામનો અમલ કર્યા બાદ, આપણે નવી બનાવેલી "Charfile1.txt" ફાઇલની અંદર પડેલ લખાણ દર્શાવવા cat કમાન્ડનો ઉપયોગ કરીશું.

```
// Write to a file using character stream
import java.io.*;
class FileWriterDemo
{
    public static void main(String args[])
    {
        FileWriter fwobject = null;
        try {

            // Create an object of FileWriter
            fwobject = new FileWriter("Charfile1.txt");

            //Write strings to the file
            fwobject.write("File writing starts...");

            for(int i = 1; i < 11 ; i++)
                fwobject.write("Line : " + i + "\n");

            fwobject.write("File writing ends...");
        }
        catch(Exception eobj)
        {
            System.out.println(eobj);
```

```
                }
                finally
                {
                        try {
                        // Close the filewriter
                        fwobject.close();
                        }
                        catch(Exception eobj)
                        {
                                System.out.println(eobj);
                        }
                }
        }
}
```

કોડલિસ્ટિંગ 11.2 : ફાઈલ લખવાની ક્રિયા વર્ણવતો પ્રોગ્રામ

ફાઈલમાં લખવાની પ્રક્રિયા પૂર્ણ થયા પછી સ્ટ્રીમ ઑબ્જેક્ટ બંધ કરાય તે અગત્યનું છે. ખુલ્લી રહેલી ફાઈલ કમ્પ્યૂટર સિસ્ટમમાં સંસાધનો વાપરે છે અને ફાઈલની રીત મુજબ અન્ય પ્રોગ્રામ તેનો ઉપયોગ પણ ન કરી શકે તેવું બને. જેવી ક્રિયા પૂરી થાય કે તરત જ ફાઈલ બંધ કરવી જરૂરી છે.



11.4 : કોડલિસ્ટિંગ 11.2નું પરિણામ

રીડર ક્લાસ (Reader Classes)

કૅરેક્ટર-સ્ટ્રીમ વાંચવા માટે રીડર ક્લાસ એ આધારરૂપ ક્લાસ છે. ઍબસ્ટ્રૅક્ટ રીડર ક્લાસ ચોક્કસ કાર્યપ્રણાલી ઘોષિત કરે છે, જે તમામ કૅરેક્ટર ઇનપુટ-સ્ટ્રીમ માટે ઉપલબ્ધ બને છે. આપણે ફાઈલમાંથી વાંચવાની ક્રિયા કરવા માટે આપણા પ્રોગ્રામમાં FileReader ક્લાસનો ઉપયોગ કરીશું. કોષ્ટક 11.4 રીડર-ક્લાસની કેટલીક પદ્ધતિઓની યાદી રજૂ કરે છે, જે પદ્ધતિઓ તેના સબ-ક્લાસ દ્વારા ઉપયોગમાં લેવાય છે.

પદ્ધતિ (Method)	વર્ણન (Description)
void close()	streamને બંધ કરે છે.
int read()	સ્ટ્રીમમાંથી પછીનો ઉપલબ્ધ અક્ષર વાંચે છે. સ્ટ્રીમનો અંત દર્શાવવા તે "-1" પરત કરે છે.

કોષ્ટક 11.4 : FileReader ક્લાસની થોડીક પદ્ધતિઓ

InputStreamReader ક્લાસ રીડર ક્લાસને વિસ્તારે છે. તે બાઇટની સ્ટ્રીમને અક્ષરોની સ્ટ્રીમમાં પરિવર્તિત કરે છે. FileReaderClass, InputStreamReaderને વિસ્તારે છે અને ફાઇલમાંથી અક્ષરો વાંચે છે. તેના કન્સ્ટ્રક્ટરોમાંનાં કેટલાંક નીચે મુજબ છે :

FileReader(String filepath) throws FileNotFoundException

FileReader(File fileobj) throws FileNotFoundException

FileReaderનો એક ઑબ્જેક્ટ આપણે નીચે મુજબ બનાવી શકીએ :

FileReader frobject = new FileReader("/java/files/Charfile1.txt");

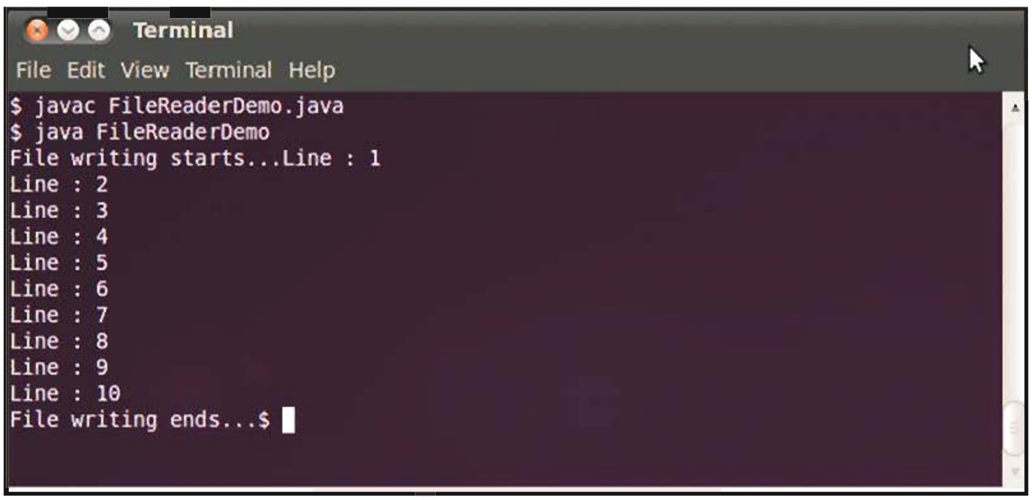
કોડલિસ્ટિંગ 11.2માં દર્શાવેલ પ્રોગ્રામમાં આપણે "Charfile1.txt" ફાઇલમાં લખવા પ્રયત્ન કર્યો. હવે ચાલો આપણે એક એવો પ્રોગ્રામ લખીએ, જે આપણે બનાવેલી ફાઇલમાંથી માહિતી વાંચતો હોય. કોડલિસ્ટિંગ 11.3માં દર્શાવેલ પ્રોગ્રામ FileReader ક્લાસની read() પદ્ધતિનો ઉપયોગ કરી ફાઇલમાંથી ડેટા વાંચે છે. આ read પદ્ધતિ એ Reader ક્લાસમાંથી વારસામાં મેળવાયેલ છે.

```
// Reading from a file using character stream
import java.io.*;
class FileReaderDemo
{
        public static void main(String args[])
        {
                FileReader frobject = null;
                try  {

                        // Create an object of FileReader
                        frobject = new FileReader("Charfile1.txt");
                        int i;
                        char ch;
                        while ( ( i = frobject.read()) != -1)
                        {
                                ch = (char) i ;
                                System.out.print(ch);
                        }
                }
                catch(Exception eobj)
                {
                        System.out.println(eobj);
                }
                finally
                {
```

```
                    try {
                    // Close the filewriter
                    frobject.close();
                }
                catch(Exception eobj)
                {
                    System.out.println(eobj);
                }
            }
        }
    }
```

કોડલિસ્ટિંગ 11.3 : ફાઇલમાંથી વાંચવાની ક્રિયા વર્ણવતો પ્રોગ્રામ

કોડલિસ્ટિંગ 11.3નું પરિણામ આકૃતિ 11.5માં દર્શાવેલ છે. અહીં એ જણાય છે કે, પ્રોગ્રામનો અમલ સ્ક્રીન ઉપર પરિણામ દર્શાવે છે. તે "Charfile1.txt" ફાઇલના તમામ લખાણને દર્શાવે છે.



11.5 : કોડલિસ્ટિંગ 11.3નું પરિણામ

ફાઇલનો અંત (End of File - EOF) દર્શાવવા એક ખાસ ચિહ્ન છે. ફાઇલમાંથી વાંચતી વખતે, ફાઇલનો અંત આવી ગયો છે, એવી પ્રોગ્રામને ખબર પડવી જ જોઈએ. જોકે, પ્રોગ્રામ ઇનપુટ-સ્ટ્રીમમાંથી વાંચે છે, માટે java read() પદ્ધતિ સ્ટ્રીમમાં ડેટાનો અંત દર્શાવવા "-1" પરત કરે છે.

બાઇટ-સ્ટ્રીમ-ક્લાસ (Byte Stream Classes)

અત્યાર સુધી આપણે એ જોયું કે, java.io પૅકેજનો ઉપયોગ કરીને અક્ષરો પર કેવી રીતે પ્રક્રિયા કરી શકાય. મોટા ભાગના જીવંત ઉપયોગોમાં આંકડાકીય ગણતરીઓની જરૂર પડે છે. જેમકે, ફાઇલમાં માલસામાન (ઇન્વેન્ટરી)ની વિગતો સાચવવી અને પડતરકિંમતની ગણતરી કરવી અથવા ફાઇલમાં કર્મચારીઓની માહિતી સાચવવી અને તેમના પગારની ગણતરી કરવી. આવા પ્રકારની ક્રિયાઓ માટે જાવા બાઇનરી સ્ટ્રીમ (binary stream) અને તેમના સહયોગી ક્લાસ આપે છે.

java.io પૅકેજમાં FileInputStream અને FileOutputStream ક્લાસ, આપણને ડિસ્કમાં કોઈ પણ ફાઇલમાં બાઇટ લખવા માટે કે ફાઇલમાંથી બાઇટ વાંચવા માટે સુવિધા આપે છે. આ ક્લાસ InputStream અને OutputSream ક્લાસના સબ-ક્લાસ છે.

FileOutputStream

FileOutputStream એ OutputStreamનો સબ-ક્લાસ અને કોઈ આઉટપુટ-સ્ટ્રીમ અથવા ફાઈલમાં બાઈટ લખવા માટે ઉપયોગમાં લેવાય છે. આ ક્લાસનો ઉપયોગ કરવા માટે સૌપ્રથમ આપણે ફાઈલ ઑબ્જેક્ટ બનાવવો પડે એ પછી, ફાઈલમાં બાઈટ લખવા OutputStream ઍબ્સ્ટ્રેક્ટ ક્લાસમાંથી લાવેલ write પદ્ધતિનો ઉપયોગ કરી શકીએ. FileOutputStream ક્લાસની વધુ ઉપયોગમાં લેવાતી કેટલીક પદ્ધતિઓ કોષ્ટક 11.5માં દર્શાવેલ છે.

પદ્ધતિ (Method)	વર્ણન (Description)
void close()	ફાઈલ આઉટપુટ-સ્ટ્રીમને બંધ કરે છે અને સ્ટ્રીમ સાથે સંકળાયેલ કોઈ પણ સિસ્ટમ સંસાધનોને મુક્ત કરે છે.
void write(int b)	આ આઉટપુટ-સ્ટ્રીમમાં નિર્દિષ્ટ બાઈટ લખે છે.
void write(byte[] b)	આ ફાઈલ આઉટપુટ-સ્ટ્રીમમાં, નિર્દિષ્ટ બાઈટ ઍરેમાંથી b.length બાઈટ લખે છે.

કોષ્ટક 11.5 : FileOutputStream ક્લાસની થોડીક પદ્ધતિઓ

FileOutputStreamનું કન્સ્ટ્રક્ટર ફાઈલના સ્થાનનો પથ દર્શાવતી શબ્દમાળા (string) અથવા ફાઈલ ક્લાસનો ઑબ્જેક્ટ જ સ્વીકારી શકે છે. ચાલો, સામાન્ય રીતે ઉપયોગમાં લેવાતાં આવા કેટલાક કન્સ્ટ્રક્ટર જોઈએ :

FileOutputStream(String name) throws FileNotFoundException

અથવા

FileOutputStream(File file) throws FileNotFoundException

FileOutputStream નાં દૃષ્ટાંત બનાવવાનાં ઉદાહરણ નીચે દર્શાવ્યા મુજબ છે :

FileOutputStream fosobject = new FileOutputStream("/home/Akash/myfile.txt");

અન્ય રીતે પણ આપણે ઉપયોગ કરી શકીએ :

File fobj = new File("/home/Akash/myfile.txt");

FileOutputStream fosobject = new FileOutputStream(fobj);

જો ફાઈલનું નામ એ કોઈ ફાઈલને બદલે કોઈ ડિરેક્ટરીનો નિર્દેશ કરતું હોય કે ફાઈલ હયાત જ ન હોય, કે કોઈ કારણસર ફાઈલ ખોલી શકાય તેમ ન હોય, તો ઉપર દર્શાવેલ કન્સ્ટ્રક્ટર FileNotFoundException થ્રો કરશે.

FileInputStream

FileInputStream એ InputStreamનો સબ-ક્લાસ છે અને ફાઈલમાંથી બાઈટ વાંચવા માટે ઉપયોગમાં લેવાય છે. કોષ્ટક 11.6માં દર્શાવ્યા પ્રમાણે તે ફાઈલોમાંથી વાંચવાની ક્રિયા કરવા માટે કેટલીક પદ્ધતિના ગણ પૂરા પાડે છે.

પદ્ધતિ (Method)	વર્ણન (Description)
void close()	ફાઈલ ઇનપુટ-સ્ટ્રીમને બંધ કરે છે અને સ્ટ્રીમ સાથે સંકળાયેલ કોઈ પણ સિસ્ટમ સંસાધનોને મુક્ત કરે છે.
int read()	આ ઇનપુટ-સ્ટ્રીમમાંથી ડેટાબાઈટ વાંચે છે.
int read(byte[] b)	આ ફાઈલ ઇનપુટ-સ્ટ્રીમમાં, નિર્દિષ્ટ બાઈટ ઍરેમાંથી b.length વાંચે છે.

કોષ્ટક 11.6 : FileInputStream ક્લાસની થોડીક પદ્ધતિઓ

આપણે એક એવો પ્રોગ્રામ જોઈએ કે જે કોડલિસ્ટિંગ 11.4માં દર્શાવ્યા પ્રમાણે ફાઈલમાં લખવા અને વાંચવાની ક્રિયા કરવા માટે ઉપર દર્શાવેલ ક્લાસનો ઉપયોગ કરતો હોય.

```
//Program to read and write bytes to binary file
import java.io.*;
class BinaryFileDemo {
        public static void main(String args[])
        {
                FileOutputStream outobject = null;
                FileInputStream inobject = null;
                String cities = " Rajkot \n Ahmedabad \n Vadodara \n Vapi \n";

                //Convert cities into byte array
                byte citiesarray[] = cities.getBytes();

                try {
                        // Create object of Binary output stream
                        outobject = new FileOutputStream("Binaryfile.dat");
                        //Write the array of bytes into file
                        outobject.write(citiesarray);
                        outobject.close();

                        //Create object of Binary input stream
                        inobject = new FileInputStream("Binaryfile.dat");
                        //Variable to read each byte
                        int i;
                        //Read each byte from the file and display
                        while((i = inobject.read())!=-1)
                        {
                                System.out.print((char)i);
                        }
                        inobject.close();

                }
                catch(Exception eobj)
                {
                        System.out.println(eobj);
                }
        }
}
```

કોડલિસ્ટિંગ 11.4 : બાઈનરી ફાઈલમાં બાઈટ લખવા અને વાંચવા માટેનો પ્રોગ્રામ

કોડલિસ્ટિંગ 11.4માં દર્શાવેલ પ્રોગ્રામ "Binaryfile.dat" ફાઇલમાં એક શબ્દમાળા (સ્ટ્રીંગ) લખે છે, પછીથી તે એ જ ફાઇલમાંથી ડેટા વાંચવા અને સ્ક્રીન પર દર્શાવવા FileInputStreamનો એક ઑબ્જેક્ટ બનાવે છે. આપણે એ નોંધવું જ જોઈએ કે, વાંચવાની ક્રિયા પૂર્ણ થયા પછી પૂર્ણાંક સંખ્યાને અક્ષરમાં પરિવર્તિત કરવા typecastingનો અમલ કરવામાં આવ્યો છે.

નોંધ : ટેક્સ્ટફાઇલ અને બાઇનરી ફાઇલ વચ્ચેનો તફાવત નિહાળવા માટે, કોડલિસ્ટિંગ 11.2 અને 11.4માં આપેલ પ્રોગ્રામના ઉપયોગ દ્વારા તૈયાર કરેલ ફાઇલો ખોલો.

કી-બોર્ડ પરથી મળતા ઈનપુટની પ્રક્રિયા

આ વિભાગમાં, આપણે જાવાપ્રોગ્રામને કી-બોર્ડ દ્વારા ડેટા ઈનપુટ કરવા માટેના વિવિધ રસ્તા જાણીશું. આપણે જાણીએ છીએ તે પ્રમાણે, કોઈ પણ પ્રોગ્રામ, કી-બોર્ડ/GUI મારફત જીવંત સંપર્ક દ્વારા અથવા 'કમાન્ડલાઇન આર્ગ્યુમેન્ટ' (કમ્પ્યૂટરને અપાતા આદેશની સાથે જ ટાઇપ કરાતો ડેટા) અથવા ફાઇલમાંથી જરૂરી ડેટા ઈનપુટ મેળવી શકે. જોકે, એવા ઘણા ક્લાસ છે, જે કી-બોર્ડ પરથી ઈનપુટ મેળવવાની સવલત કરી આપી શકે. અહીં આપણે સૌથી વધુ ઉપયોગમાં લેવાતી ટેક્નિક પૈકીની બે ટેક્નિકની ચર્ચા કરીશું, જેમાંની એક છે, java.util પૅકેજનો સ્કૅનર ક્લાસ (scanner class) અને બીજો છે java.io પૅકેજનો કૉન્સોલ ક્લાસ (console class).

સ્કૅનરક્લાસ

સ્કૅનરક્લાસ, java.util પૅકેજનો એક ભાગ છે; તે કી-બોર્ડ કે ફાઇલ પરથી ઈનપુટ મેળવવા માટે વિવિધ પદ્ધતિઓ પ્રદાન કરે છે. આ ક્લાસની ખાસ વિશેષતા એ છે કે, તે એક ચિહ્ન (delimiter)નો ઉપયોગ કરીને તે ઈનપુટ કરાતી શબ્દમાળાને ટોકન (શબ્દો)માં વિભાજિત કરે છે. (ખાલી જગ્યા (white space) એ સામાન્ય રીતે ઉપયોગમાં લેવાતું ચિહ્ન છે.) દરેક ટોકન જુદા-જુદા પ્રકારનાં હોઈ શકે છે. ઉદાહરણ તરીકે, "India-1947" જેવી શબ્દમાળા "String-int" કિંમત તરીકે વાંચી શકાય. હવે આપણે કન્સ્ટ્રક્ટર અને સ્કૅનરક્લાસની કેટલીક પદ્ધતિઓ વિશે જાણીએ :

Scanner(String str)

Scanner(InputStream isobject)

Scanner(File fobject) throws FileNotFoundException

સ્કૅનર-ઑબ્જેક્ટ એક શબ્દમાળા, ફાઇલ-ઑબ્જેક્ટ અથવા InputStream ઑબ્જેક્ટમાંથી બનાવી શકાય છે, ઉદાહરણ તરીકે ફાઇલ અને કી-બોર્ડ પરથી વાંચવા માટે આપણે નીચેની રીતે કન્સ્ટ્રક્ટરનો ઉપયોગ કરી શકીએ.

Scanner fileinput = new Scanner(new File("Students.dat"));

Scanner kbinput = new Scanner(System.in);

સ્કૅનરક્લાસ સાથે ઉપલબ્ધ કેટલીક અગત્યની પદ્ધતિઓની યાદી કોષ્ટક 11.7માં દર્શાવેલ છે.

પદ્ધતિ (Method)	વર્ણન (Descrption)
void close()	સ્કૅનરને બંધ કરે છે.
String next()	પછીનો ટોકન પરત કરે છે.
boolean hasNext()	જો ઈનપુટમાં ટોકન હશે, તો true કિંમત પરત કરે છે.
int nextInt()	ઈનપુટની પછીના ટોકનને Int તરીકે સ્કૅન કરે છે.
float nextFloat()	ઈનપુટની પછીના ટોકનને Float તરીકે સ્કૅન કરે છે.
String nextLine()	ઈનપુટની પછીના ટોકનને Line તરીકે સ્કૅન કરે છે.

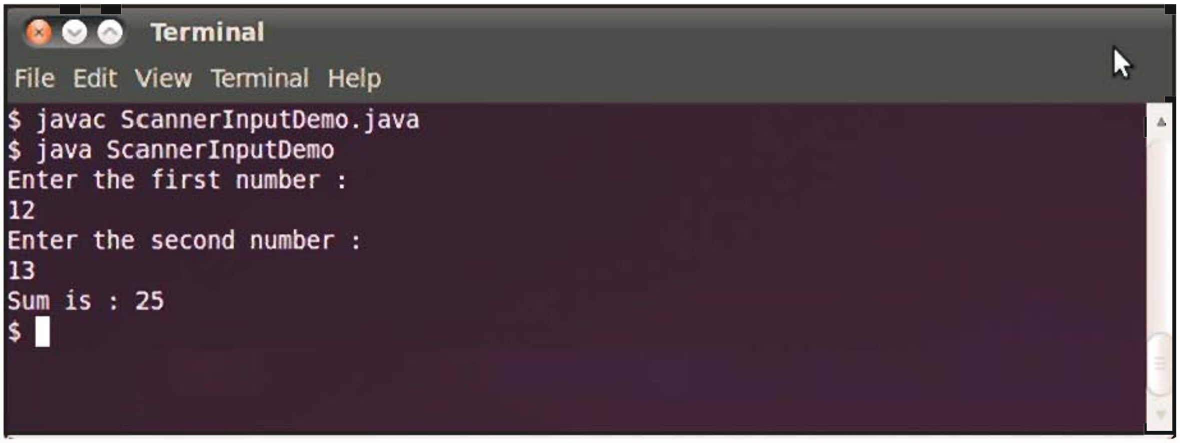
કોષ્ટક 11.7 : Scanner ક્લાસની પદ્ધતિઓ

આપણે હવે એક એવો પ્રોગ્રામ જોઈએ કે જે ઉપયોગકર્તા પાસેથી પરસ્પર સંવાદ દ્વારા બે સંખ્યા વાંચે અને તે બે સંખ્યાનો સરવાળો દર્શાવે. કોડલિસ્ટિંગ 11.5 આ પ્રોગ્રામ દર્શાવે છે.

```
//Accepts input at command prompt
import java.io.*;
import java.util.*;
class ScannerInputDemo
{
        public static void main(String args[])
        {
                Scanner kbinput = null;
                int number1;
                int number2;
                int sum=0;
                try
                {
                        // Create an object of Scanner class
                        // that reads from Standard Input
                        kbinput = new Scanner(System.in);
                        System.out.println("Enter the first number : ");
                        //Read the integer number from console
                        number1 = kbinput.nextInt();
                        System.out.println("Enter the second number : ");
                        //Read the integer number from console
                        number2 = kbinput.nextInt();
                        sum = number1 + number2;
                        System.out.println("Sum is : " + sum);
                }
                catch(Exception eobj)
                {
                        System.out.println(eobj);
                }
                finally {
                        try {
                                kbinput.close();
                        }
                        catch(Exception eobj)
                        {
                                System.out.println(eobj);
                        }
                }

        }
}
```

કોડલિસ્ટિંગ 11.5 : બે સંખ્યા ઉમેરવાનો પ્રોગ્રામ

કોડલિસ્ટિંગ 11.5માં આપણે સ્કૅનરક્લાસનો એક ઑબ્જેક્ટ બનાવ્યો. સ્કૅનર ક્લાસનો કન્સ્ટ્રક્ટર આર્ગ્યુમેન્ટ તરીકે "System.in" સ્વીકારે છે, જેથી કરીને તે સ્ટાન્ડર્ડ ઇનપુટ (કી-બોર્ડ) પરથી વાંચી શકે. આ બંને સંખ્યાઓ, સ્કૅનરક્લાસની nextInt() પદ્ધતિનો ઉપયોગ કરીને પૂર્ણાંક સંખ્યા તરીકે લેવાશે. આકૃતિ 11.6 પ્રોગ્રામનું પરિણામ દર્શાવે છે.



11.6 : કોડલિસ્ટિંગ 11.5નું પરિણામ

ફાઇલમાંથી વાંચવા માટે પણ સ્કૅનરક્લાસનો ઉપયોગ થઈ શકે છે. આપણે એક એવું ઉદાહરણ જોઈએ, જેમાં થોડા વિદ્યાર્થીઓ વિશે માહિતી ધરાવતી ફાઇલમાંથી ડેટા વાંચવા માટે આપણે સ્કૅનરક્લાસનો ઉપયોગ કરતા હોઈએ. એવું ધારી લઈએ છીએ કે "Students.dat" ફાઇલ હયાત છે જ. તેમાં 5 ફિલ્ડ છે : student_no, student_name, ત્રણ વિષયના ગુણની માહિતી ધરાવે છે. Students.dat ફાઇલનું માળખું અને ડેટા નીચે દર્શાવ્યા મુજબ છે :

1 Akash 45 65 55

2 Badal 10 20 30

3 Zakir 45 40 60

4 David 65 50 75

દરેક વિદ્યાર્થીનો ડેટા વાંચવા અને કુલ ગુણની ગણતરી કરીને તેને પરિણામ રૂપે રજૂ કરવા કોડલિસ્ટિંગ 11.6માં દર્શાવ્યા મુજબ આપણે એક પ્રોગ્રામ લખીશું.

```
//Accepts input from a file "Students.dat" and calculate the total marks of each student
import java.io.*;
import java.util.*;
class ScannerFileDemo
{
        public static void main(String args[])
        {
                Scanner fileinput = null;
                int rollno, mark1, mark2, mark3, totalmarks;
                String name = null;
                File fobject;
        try
        {
```

```
                // Specify the file from where data is to be read
                fobject = new File("Students.dat");
                // Create an object of Scanner class that reads from File
                fileinput = new Scanner(fobject);
                //Display the default Delimiter to separate fields within a file
                System.out.println("Default delimeter is : " + fileinput.delimiter() + "\n");
                // Iterate to read the values of each record
                while(fileinput.hasNext()) {
                rollno = fileinput.nextInt();
                name = fileinput.next();
                mark1=fileinput.nextInt();
                mark2=fileinput.nextInt();
                mark3=fileinput.nextInt();
                totalmarks = mark1 + mark2 + mark3;
                System.out.println("Total marks of  Rollno "  + rollno + " ," +name+ " are
                                                                     : " + totalmarks);
                }
                fileinput.close();
            }
            catch(Exception eobj)
            {
                System.out.println(eobj);
            }
    }
    }
```

કોડલિસ્ટિંગ 11.6 : પ્રત્યેક વિદ્યાર્થીના કુલ ગુણની ગણતરી કરવાનો પ્રોગ્રામ

પ્રોગ્રામનું પરિણામ આકૃતિ 11.7માં દર્શાવ્યા મુજબ મોનિટર પર દર્શાવાશે.

11.7 : કોડલિસ્ટિંગ 11.6નું પરિણામ

કૉન્સોલ ક્લાસ

અગાઉના ઉદાહરણમાં આપણે ઉપયોગકર્તાના ઇનપુટને વાંચવા સ્કૅનર ક્લાસનો ઉપયોગ કર્યો હતો. સ્કૅનર ક્લાસ ઉપરાંત java.io.Console નામનો એક અન્ય ક્લાસ પણ છે, જે કી-બોર્ડ પરથી ઇનપુટ મેળવવા ઉપયોગમાં લેવાય છે. આ ક્લાસનો ઉપયોગ ખાસ કરીને જ્યારે ઇનપુટને છૂપા સ્વરૂપે (ટાઇપ કરાતા અક્ષરો સ્ક્રીન પર ન દર્શાવાય તે રીતે) મેળવવાનું હોય, ત્યારે કરીએ છીએ.

કૉન્સોલ ક્લાસ ગુપ્ત સંકેત (પાસવર્ડ) વાંચવા માટેની એક પદ્ધતિ પૂરી પાડે છે. જ્યારે પાસવર્ડ વાંચવામાં આવે છે, ત્યારે ઉપયોગકર્તા દ્વારા મૉનિટર (સ્ક્રીન) પર ટાઇપ કરવામાં આવતા ઇનપુટને ગુપ્ત રખાય છે અથવા કૉન્સોલના સ્ક્રીન પર દર્શાવાતા નથી અને તે પરત મૂલ્ય તરીકે અક્ષરોનો ઍરે પરત કરશે. કૉન્સોલ ક્લાસ java.io પૅકેજનો એક ભાગ છે. બહોળા પ્રમાણમાં ઉપયોગમાં લેવાતી કૉન્સોલ ક્લાસની કેટલીક પદ્ધતિ કોષ્ટક 11.8માં દર્શાવેલ છે :

પદ્ધતિ (Methods)	વર્ણન (Description)
String readLine()	આ પદ્ધતિ કૉન્સોલ પરથી લખાણની એક જ લાઇન વાંચે છે.
char[] readPassword()	આ પદ્ધતિ કૉન્સોલ પરથી ઇકો (echoing)ની અસરમુક્ત અવસ્થામાં પાસવર્ડ અથવા પાસફ્રેઝ (passphrase) વાંચે છે.
Console printf(String format, Object args)	આ પદ્ધતિ કૉન્સોલના પરિણામની સ્ટ્રીમમાં ચોક્કસ સ્વરૂપ સાથેની શબ્દમાળા લખવા માટે ઉપયોગમાં લેવાય છે.

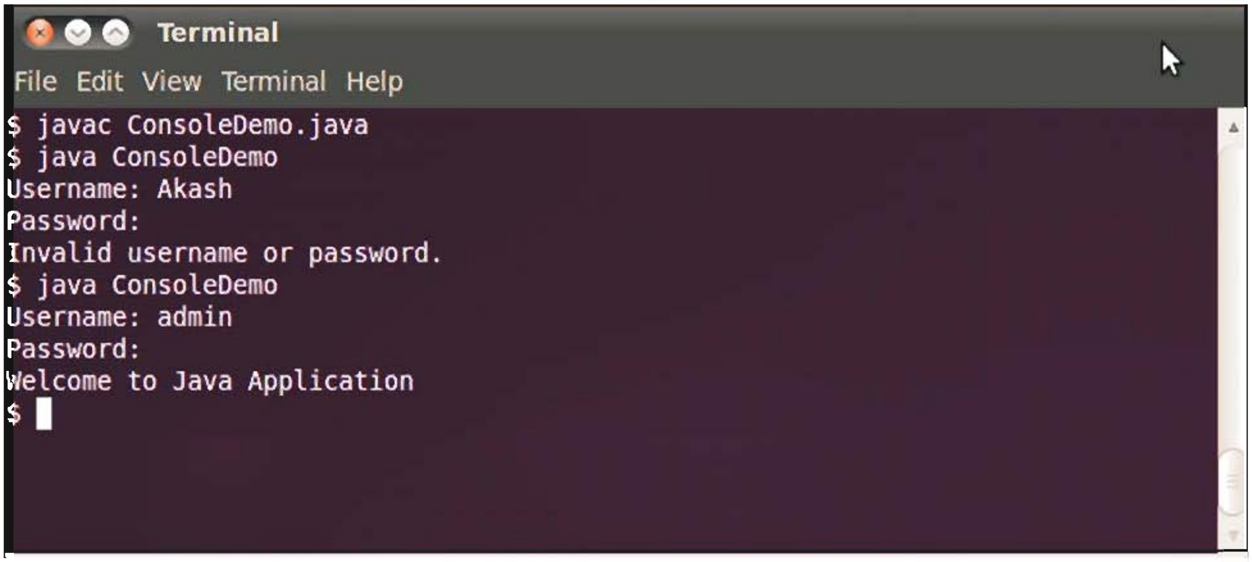
કોષ્ટક 11.8 : Console ક્લાસની પદ્ધતિઓ

કોડલિસ્ટિંગ 11.7માં દર્શાવેલ પ્રોગ્રામ, યૂઝરનેમ (username) અને પાસવર્ડ (password) વાંચવા કૉન્સોલક્લાસનો ઉપયોગ રજૂ કરે છે. વધુમાં, તે એ બાબતની ખાતરી પણ કરી લે છે કે, યૂઝરનેમ અને પાસવર્ડ સાચો છે કે નહીં.

```
// Program to reading passwords
import java.io.Console;
import java.util.Arrays;
public class ConsoleDemo {

    public static void main(String[] args) {
     Console console = System.console();
    String username = console.readLine("Username: ");
     char[] password = console.readPassword("Password: ");

   if (username.equals("admin") && String.valueOf(password).equals("secret")) {
      console.printf("Welcome to Java Application \n");
      } else {
      console.printf("Invalid username or password.\n");
      }
   }
}
```

કોડલિસ્ટિંગ 11.7 : યૂઝરનેમ અને પાસવર્ડ વાંચતો પ્રોગ્રામ

આકૃતિ 11.8માં પ્રોગ્રામનું પરિણામ દર્શાવાયું છે. યૂઝરનેમ અને પાસવર્ડ ઇનપુટ કરવા આપણે બે પ્રયત્નોનો ઉપયોગ કર્યો છે. પ્રથમ પ્રયત્નમાં યૂઝરનેમ ખોટું હતું અને તેથી આપણે invalid username or password સંદેશનો ઉપયોગ કર્યો. બીજા પ્રયત્નમાં, આપણે સાચો યૂઝરનેમ અને પાસવર્ડ દાખલ કર્યો.



11.8 : કોડલિસ્ટિંગ 11.7નું પરિણામ

અનેક ઉપયોગકર્તા પદ્ધતિના કિસ્સામાં ઉપયોગકર્તાની યાદી વાંચવા અને ફાઇલમાંથી તેમના પાસવર્ડ સરખાવવા માટે પ્રોગ્રામને વિસ્તારી શકાય.

આ પ્રકરણમાં ચર્ચેલા વિવિધ ક્લાસ ઉપરાંત, જાવાફાઇલનો ઉપયોગ કરી ઑબ્જેક્ટને સાચવવા અને પુનઃપ્રાપ્ત (retrieve) કરવા માટેના ક્લાસ પૂરા પાડે છે. તે ફાઇલનો યાદચ્છિક રીતે ઉપયોગ કરવા માટેના ક્લાસ અને પદ્ધતિઓ પણ પૂરી પાડે છે. અત્યાર સુધી, આપણે એવી પદ્ધતિઓ જોઈ કે જે ક્રમિક (sequential) ક્રિયાઓ પાર પાડી શકે. જોકે, તેમ છતાં એવા ક્લાસ પણ છે જે ક્રમિક રીતે ઉપયોગ કરવાને બદલે આપણને સીધા n-ક્રમ પરના રેકર્ડ પર જવાની સુવિધા આપે. આ બધા ક્લાસની ચર્ચા કરવી આ પાઠ્યપુસ્તકની મર્યાદાબહારનું હોઈ java.io પૅકેજની અન્ય વિવિધ રસપ્રદ લાક્ષણિકતાઓનો ખ્યાલ મેળવવાનું અમે વિદ્યાર્થીઓ પર છોડીએ છીએ.

સારાંશ

આ પ્રકરણમાં આપણે ફાઇલ સંબંધી ક્રિયાઓ વિશે શીખ્યા. ફાઇલ-ક્રિયાઓ કરવા માટે java.io.File ક્લાસનો કેવી રીતે ઉપયોગ કરવો તે આપણે જોયું. આપણે સ્ટ્રીમનો ખ્યાલ મેળવ્યો અને વિવિધ પ્રકારની ઇનપુટ અને આઉટપુટ-સ્ટ્રીમનો કેવી રીતે ઉપયોગ કરવો તે આપણે શીખ્યા. ફાઇલ કે કી-બોર્ડ પરથી ડેટાનો ઉપયોગ કરવા માટે સ્કૅનર-ક્લાસનો ઉપયોગ કેવી રીતે કરવો તે પણ આપણે શીખ્યા. અંતમાં, કી-બોર્ડ પરથી ડેટા ઇનપુટ કરવા માટે કૉન્સોલ-ક્લાસનો ઉપયોગ કેવી રીતે કરવો તે આપણે શીખ્યા.

સ્વાધ્યાય

1. જાવાપ્રોગ્રામિંગમાં ફાઇલ શા માટે અગત્યની છે ? કેવા સંજોગો હેઠળ તમે ડેટાને ફાઇલમાં સાચવશો ?
2. ફાઇલ અને ડિરેક્ટરી પર કરી શકાતી વિવિધ ક્રિયાઓ જણાવો.
3. જાવામાં શા માટે સ્ટ્રીમ (Stream)નો ખ્યાલ રજૂ કરવામાં આવ્યો છે ? સ્ટ્રીમનો ઉપયોગ કરવાના ફાયદા જણાવો.

4. નીચેના દરેક પ્રશ્નો માટે આપેલા વિકલ્પો પૈકી યોગ્ય વિકલ્પ પસંદ કરી ઉત્તર આપો :

(1) નીચેનામાંથી ક્યું વિધાન સાચું છે ?

(a) નાશવંત પ્રકારની સંગ્રહવ્યવસ્થા થોડીક સેકન્ડમાં નાશ પામે છે.

(b) નાશવંત પ્રકારનો સંગ્રહ જ્યારે કમ્પ્યૂટર બંધ કરવામાં આવે, ત્યારે નાશ પામે છે.

(c) કમ્પ્યૂટરડિસ્ક એ નાશવંત પ્રકારનું સંગ્રહસાધન છે.

(d) ઉપરનાં તમામ

(2) નીચેનામાંથી ક્યું કમ્પ્યૂટરસિસ્ટમમાં નાશવંત સાધન પર સાચવેલ ડેટાના સમૂહનો નિર્દેશ કરે છે ?

(a) ફાઇલ (b) વિનિયોગ (c) નાશવંત ડેટા (d) હાર્ડડિસ્ક

(3) નાનાથી મોટા ડેટા મુજબ ડેટા પદાનુક્રમ નીચેનામાંથી કયા ક્રમમાં આવે ?

(a) file:character:field:record (b) file:character:record:field

(c) character:field:file:record (d) character:field:record:file

(4) સ્ટ્રીમ બાબતે નીચેનામાંથી ક્યું વિધાન સાચું છે ?

(a) સ્ટ્રીમ હંમેશાં બે દિશામાં વહે છે.

(b) સ્ટ્રીમ એ માર્ગ (channel) છે, જેમાંથી ડેટા વહે છે.

(c) પ્રોગ્રામમાં કોઈ પણ એક સમયે ફક્ત એક જ સ્ટ્રીમ (પ્રવાહ) ખુલ્લી હોઈ શકે.

(d) ઉપરનાં તમામ

(5) રેકૉર્ડના ફિલ્ડની વચ્ચે નીચેનામાંથી ક્યું ચિહ્ન બે ફિલ્ડને જુદા પાડવા ઉપયોગમાં લેવાય છે ?

(a) પથ (b) ડેલિમીટર (c) ચલ (d) ખાલી જગ્યા

(6) નીચેનામાંથી કઈ ક્રિયાઓ પાર પાડવા માટે સ્કૅનરક્લાસનો ઉપયોગ કરી શકાય ?

(a) કી-બોર્ડ પરથી ઇનપુટ સ્વીકારવા (b) ફાઇલમાંથી વાંચવા

(c) ડેલિમીટર વડે જુદા પાડતી શબ્દમાળા વર્ણવવી (d) ઉપરનાં તમામ

પ્રાયોગિક સ્વાધ્યાય

1. આપેલી ડિરેક્ટરીમાં પડેલી ".txt" અનુલંબન ધરાવતી તમામ ફાઇલોની યાદી દર્શાવવા જાવાપ્રોગ્રામ લખો.

2. એક એવો જાવાપ્રોગ્રામ લખો, જે ફાઇલનું નામ સ્વીકારે અને ચકાસણી કરે કે આપેલ ફાઇલ અસ્તિત્વમાં છે કે નહીં ? જો તે ફાઇલ હયાત હોય, તો તે ફાઇલની લાક્ષણિકતાઓ દર્શાવો.

3. "friends.dat" નામની ફાઇલ બનાવવા માટે જાવાપ્રોગ્રામ લખો; writer ક્લાસનો ઉપયોગ કરી આ ફાઇલમાં તમારા મિત્રોનાં નામ લખો.

4. એક એવો જાવાપ્રોગ્રામ લખો, જેમાં ઉપયોગકર્તા ફાઈલનું નામ દાખલ કરે, એ પછી, એ ફાઈલની અન્ય ફાઈલમાં નકલ કરે.

5. /etc/passwd ની ફાઈલને તમારી ડિરેક્ટરીમાં mypasswd.dat નામ વડે નકલ કરવા જાવાપ્રોગ્રામ લખો.

6. આપેલ ફાઈલમાં અક્ષરોની સંખ્યા ગણવા માટેનો જાવાપ્રોગ્રામ લખો.

7. ફાઈલનું કદ બાઈટના માપથી દર્શાવવા જાવાપ્રોગ્રામ લખો.

8. જે ઉપયોગકર્તા પાસેથી મુદ્દલની રકમ, વ્યાજનો દર અને મુદતના વર્ષની વિગત ઈનપુટ મેળવીને સાદું વ્યાજ ગણી આપતો જાવાપ્રોગ્રામ લખો.

9. એક એવો પારસ્પરિક સંવાદ કરતો (interactive) જાવાપ્રોગ્રામ લખો. જે કોન્સોલ પર ડેટા દાખલ કરી આપેલા ઇંચની કિંમતને સેન્ટિમીટરમાં અને સેન્ટિમીટરની કિંમતને ઇંચમાં બદલી કરી આપે.

•

લેટેક્સની મદદથી દસ્તાવેજનું પ્રકાશન 12

OpenOffice.orgના રાઇટરનો ઉપયોગ કરીને દસ્તાવેજ બનાવતા આપણે શીખી ગયા. આ પ્રકરણમાં આપણે ટેક્સ અને લેટેક્સ ટાઇપસેટિંગ સૉફ્ટવેર વિશે ચર્ચા કરીશું તથા લેટેક્સના ફાયદાઓની પણ ચર્ચા કરીશું. ત્યાર બાદ ટેક્સલાઇવનો ઉપયોગ કરીને લેટેક્સ કઈ રીતે વપરાય તે ભણીશું, આ એક ટેક્સ અને લેટેક્સ સૉફ્ટવેરનું મિશ્રણ ધરાવતું પૅકેજ છે. તેમાં SciTE એડિટરનો પણ સમાવેશ થાય છે.

લેટેક્સનો ઉપયોગ (Use of LaTex)

લેટેક્સ વાપરવા માટે લેટેક્સ વિતરણ સૉફ્ટવેરની જરૂર પડે જેમાં ટેક્સ અને બીજા વધારાનાં સૉફ્ટવેરનો સમાવેશ થાય છે. ટેક્સલાઇવ એ પ્રમાણભૂત ઉબુન્ટુ રિપોઝિટરીમાં ઉપલબ્ધ ખૂબ જ લોકપ્રિય લેટેક્સ વિતરણ સૉફ્ટવેર છે. આઉટપુટ જોવા માટે બીજા સાદા એડિટર અને સૉફ્ટવેરની પણ જરૂર પડે, કારણકે લેટેક્સ જુદા-જુદા ઘણાંબધાં પ્રકારના ફાઇલમાળખાંમાં આઉટપુટ આપે છે. આ માટે ફાઇલમાળખાને આધારે આઉટપુટ જોવા માટે જુદાં-જુદાં સંબંધિત સૉફ્ટવેરની જરૂર પડે છે.

કોઈ પણ સાદા ટૅક્સ્ટ-એડિટરનો ઉપયોગ કરીને લેટેક્સનો દસ્તાવેજ બનાવી શકાય (જેવા કે gedit અથવા SciTE). દસ્તાવેજ બનાવવા માટે તેના જુદા-જુદા ભાગોને લેટેક્સના કમાન્ડ વડે ચિહ્નિત કરવામાં આવે છે. ઉદાહરણ તરીકે, દસ્તાવેજનું શીર્ષક આપવા માટે **\title** કમાન્ડનો ઉપયોગ થાય છે. દસ્તાવેજના લેખકનું નામ લખવા **\author** તેમજ દસ્તાવેજના સર્જનની તારીખ આપવા માટે **\date** કમાન્ડનો ઉપયોગ થાય છે. આ જ રીતે, **\chapter, \section, \subsection, \paragraph** વગેરે કમાન્ડ દસ્તાવેજનું લૉજિકલ માળખું બનાવવા માટે વપરાય છે.

આ દસ્તાવેજને વ્યાવસાયિક શૈલીમાં ગોઠવવા માટેના આંતરિક રસ્તાઓ લેટેક્સમાં ઉપલબ્ધ છે. જ્યારે આપણે દસ્તાવેજનું લખાણ લખીએ છીએ, ત્યારે આપણને કમાન્ડ સાથેનું ગોઠવણી વગરનું લખાણ જ દેખાય છે. ત્યાર પછી લેટેક્સ દસ્તાવેજ ઉપર પ્રક્રિયા કરીને એક આઉટપુટ ફાઇલ બનાવે છે. સાથે-સાથે કેટલીક વધારાની ફાઇલો પણ બનાવે છે. ઘણા બધા કિસ્સામાં આ વધારાની ફાઇલોને કોઈ પણ માહિતી ગુમાવ્યા વગર સુરક્ષિત રીતે (ડિલિટ) દૂર કરી શકાય છે.

જ્યારે આપણે યોગ્ય સૉફ્ટવેરની મદદથી આઉટપુટ ફાઇલ જોઈશું અથવા પ્રિન્ટર વડે પ્રિન્ટ કાઢીશું, ત્યારે આપણને એક વ્યવસ્થિત માળખામાં ગોઠવેલ દસ્તાવેજ જોવા મળશે. જો આપણે દસ્તાવેજના આ દેખાવથી સંતુષ્ટ ન હોઈએ તો, આપણે આંતરિક શૈલી અથવા આપણી પોતાની શૈલીનો ઉપયોગ કરીને દસ્તાવેજમાં ફેરફાર કરી શકીએ.

જ્યારે-જ્યારે દસ્તાવેજમાં આવો ફેરફાર કરીશું, ત્યારે ફેરફારની અસર આઉટપુટમાં જોવા માટે દસ્તાવેજને પુનઃ-કમ્પાઇલ કરવો પડશે. ટેક્સ અને લેટેક્સ બંને .tex ફાઇલ અનુલંબનનો ઉપયોગ કરે છે. લેટેક્સમાં હવે pdflatex કમાન્ડ ઉપલબ્ધ છે, જે PDF (Portable Document Format) માળખામાં આઉટપુટ આપે છે. PDF ફાઇલને સ્ક્રીન ઉપર જોઈ શકાય છે, તેમજ પ્રિન્ટરની મદદથી પ્રિન્ટ પણ કાઢી શકાય છે. જે પ્રિન્ટ મૉનિટર ઉપર દેખાય છે તેવી જ આબેહૂબ હશે. વેબસાઇટ ઉપર દસ્તાવેજનું વિતરણ (શેરિંગ) કરવા માટે PDF ફાઇલ ખૂબ જ પ્રચલિત છે. ઉબુન્ટુના evince ડૉક્યુમેન્ટ વ્યુવર સૉફ્ટવેરમાં PDF દસ્તાવેજને જોઈ શકાય છે. જ્યારે દસ્તાવેજને સંપાદિત-કમ્પાઇલ અને જોવાનો ક્રમ આ મુજબ છે :

- Gedit જેવા કોઈ સાદા ટેક્સ્ટ-એડિટરની મદદથી દસ્તાવેજ સંપાદિત કરો.
- જ્યાં ટેક્સફાઇલ સેવ કરી હોય તે ડિરેક્ટરીમાં ***pdflatex filename*** કમાન્ડ વડે પ્રૉમ્પ્ટ ઉપરથી દસ્તાવેજને કમ્પાઇલ કરો.

- GUIની મદદ વડે અથવા કમાન્ડ પ્રૉમ્પ્ટ ઉપરથી ***evince pdffilename*** કમાન્ડ આપીને બનેલી PDF ફાઇલને જુઓ. (જ્યારે તમે PDF ફાઇલ બંધ કરશો, ત્યાર પછી જ ટર્મિનલ તમને પ્રૉમ્પ્ટ દેખાડશે.)

લેટેક્સ ફાઇલમાં સુધારાવધારા કરવા માટે આપણે SciTE (સાઇટ) એડિટરનો ઉપયોગ કરી શકીએ છીએ. gedit અને SciTE બંનેમાં (સિન્ટેક્સને) વાક્યરચનાને અલગ રીતે પ્રદર્શિત કરવામાં આવે છે. (વાંચવામાં અને ઓળખવામાં સરળતા માટે બીજી ભાષાના ઘટકને અલગ રંગમાં દર્શાવવામાં આવે છે.), gedit એડિટર કરતા SciTEમાં એક ફાયદો એ છે કે તેમાં જ દસ્તાવેજને કમ્પાઇલ કરીને જોઈ શકાય છે. તમે બધા જ, SciTE એડિટરથી પરિચિત છો. આપણે લેટેક્સ ભણવા માટે અહીં SciTEનો જ ઉપયોગ કરીશું. પીડીએફ લેટેક્સ સાથે SciTEનો ઉપયોગ કરવા માટે પ્રકરણના અંતમાં આપેલ માહિતી મુજબ કન્ફિગ્યુરેશન ફાઇલમાં ફેરફાર કરવો પડે. જરૂરી સૉફ્ટવેર પ્રસ્થાપિત કર્યા પછી ઉપયોગકર્તાએ ફક્ત એક જ વખત આ કાર્ય કરવું પડે છે.

લેટેક્સ ભાષા (The LaTex Language)

લેટેક્સ એ મૂળભૂત એક માર્ક-અપ ભાષા છે. લેટેક્સના પ્રોગ્રામમાં સાદું લખાણ તેમજ માર્ક-અપ લખાણ હોય છે, જેને કમાન્ડ તરીકે ઓળખવામાં આવે છે. કેટલાક કમાન્ડ સ્વતંત્ર હોય છે, તે લખાણના કોઈ પણ ભાગને નિશાન કરતા નથી. આવા કમાન્ડ જ્યારે લેટેક્સ દસ્તાવેજની પ્રક્રિયા કરે છે ત્યારે ઘણાંબધાં મહત્ત્વનાં કાર્ય કરતા હોય છે. જેમકે લખાણ અથવા દસ્તાવેજ વિશેની માહિતી પૂરી પાડે છે, દસ્તાવેજના માળખામાં નિશાની કરેલ લખાણની ભૂમિકા દર્શાવે છે. (અને આના કારણે લેટેક્સ લખાણને ચોક્કસ માળખામાં ગોઠવે છે.) તે સીધેસીધું ગોઠવણીનું માળખું આપે છે અથવા લેટેક્સને દસ્તાવેજની ચોક્કસ પ્રક્રિયા કરવાની સૂચના આપે છે. (ઉદાહરણ તરીકે, નિશ્ચિત પાનાંના કદનો ઉપયોગ કરવો, એકી નંબરના પેજ ઉપરથી જ નવું પ્રકરણ શરૂ કરવું વગેરે....)

લેટેક્સના કમાન્ડની શરૂઆત \બૅકસ્લેશ અક્ષર પછી કમાન્ડનું નામ એ રીતે થાય છે. કમાન્ડનું નામ ફક્ત મૂળાક્ષરો અથવા મૂળાક્ષરો સિવાયના એક અક્ષરનું બનેલું હોય છે. લેટેક્સના કમાન્ડ કેસ-સૅન્સિટિવ છે. (કૅપિટલ (મુખ્ય) અને નાના અક્ષરને અલગ અલગ ગણે છે.) કેટલાક કમાન્ડ વધારાની માહિતી પણ સ્વીકારે છે. (ઉદાહરણ તરીકે \textcolor કમાન્ડમાં લખાણ ક્યા રંગમાં દર્શાવવું છે, તે રંગ પણ આપવો પડે છે.) આ વધારાની માહિતીને આર્ગ્યુમેન્ટ તરીકે ઓળખવામાં આવે છે.

આર્ગ્યુમેન્ટ મુખ્ય બે પ્રકારની હોય છે. વૈકલ્પિક આર્ગ્યુમેન્ટ, નામ પ્રમાણે આવી આર્ગ્યુમેન્ટ ફરજિયાત હોતી નથી. આવી આર્ગ્યુમેન્ટ આપણે આપી શકીએ અથવા ન આપીએ તોપણ ચાલે. જો આપણે આવી એક અથવા એક કરતા વધારે વૈકલ્પિક આર્ગ્યુમેન્ટ આપવી હોય, તો તેને કમાન્ડના નામ પછી [ ] (ચોરસ કૌંસમાં) અલ્પવિરામ કરીને આપી શકાય. આ આર્ગ્યુમેન્ટ પછી { } (છગડિયા કૌંસ)માં આપેલ ફરજિયાત આર્ગ્યુમેન્ટ (જો હોય તો) આપવામાં આવે છે. ઉદાહરણ તરીકે, \documentclass[12pt]{article}; કમાન્ડમાં documentclass એ કમાન્ડનું નામ છે. 12pt એ વૈકલ્પિક આર્ગ્યુમેન્ટ છે જ્યારે article એ ફરજિયાત આર્ગ્યુમેન્ટ છે.

લેટેક્સમાં બધા જ વ્હાઇટ સ્પેસ અક્ષરો (સ્પેસ, ટેબ અને નવી લાઇન) જેમ છે તેમજ રાખવામાં આવે છે. એક કરતાં વધારે વ્હાઇટ સ્પેસ અક્ષરોને એક અક્ષરમાં પરિવર્તિત કરવામાં આવે છે. લીટીની શરૂઆતમાં આપવામાં આવેલ વ્હાઇટ સ્પેસ અક્ષરને અવગણવામાં આવે છે અને એક કરતાં વધારે ખાલી લીટીને નવા ફકરાની શરૂઆત ગણવામાં આવે છે. એટલે આનો મતલબ એ થયો કે આપણે લખાણને એક કરતાં વધારે લીટીમાં લખીશું, તો પણ તે લખાણને જ્યાં સુધી ખાલી લીટી નહિ આવે, ત્યાં સુધી એ આઉટપુટમાં સતત લખાણ તરીકે જ દર્શાવશે. જો લખાણમાં નવી લીટી ઉમેરવી હોય, તો \\ (લાઇન બ્રેક કમાન્ડ) લીટીના અંતમાં આપવો પડે છે. પરંતુ ફકરાની છેલ્લી લાઇનમાં આ કમાન્ડ આપવાની જરૂર નથી. આકૃતિ 12.1માં આવું સતત લખાણ તથા લખાણમાં લાઇનબ્રેક ઉમેરતું ઉદાહરણ આપવામાં આવેલું છે. (\textsf કમાન્ડની ચર્ચા પછી કરીશું.) આકૃતિ 12.2 લેટેક્સ ફાઇલનું આઉટપુટ દર્શાવે છે.

```
\documentclass[12pt]{article}
\title{Line handling in \LaTeX}
\date{May 2013}
\begin{document}
  \maketitle
  \section{Continuous Text}\textsf{
  We have no wings,  we cannot fly
  But we have legs to sail and climb
  By slow degrees and by and by
  The cloudy summits of our time}
  \section{Text with Seperate Lines}\textsf{
  Heights by great men reached and kept \\
  Were not attained by a sudden flight \\
  But they, while their companions slept, \\
  Were toiling upwards in the night}
\end{document}
```

આકૃતિ 12.1 : લીટીના સંચાલનનું ઉદાહરણ દર્શાવતી ફાઈલ

1 Continuous Text

We have no wings, we cannot fly But we have legs to sail and climb By slow degrees and by and by The cloudy summits of our time

2 Text with Seperate Lines

Heights by great men reached and kept
Were not attained by a sudden flight
But they, while their companions slept,
Were toiling upwards in the night

આકૃતિ 12.2 : લીટી-સંચાલનના ઉદાહરણનું આઉટપુટ

નીચેના અક્ષરો એ લેટેક્સના અનામત (Reserved) અક્ષરો છે.

$ % & _ (અન્ડરસ્કોર) { } ^ ~ \

લેટેક્સમાં આ અક્ષરોનો ખાસ અર્થ છે. આ અક્ષરોનો સીધેસીધો ઉપયોગ લેટેક્સના લખાણમાં કરી શકાતો નથી. જો આ અક્ષરો આપણે લખાણમાં વાપરવા હોય, તો તે નીચે મુજબ લખીને વાપરી શકાય છે.

\# \$ \% \& _ \{ \} \^{} \~{} \textbackslash{}

છેલ્લા ત્રણ અક્ષરોની ખાસ નોંધ લો. ચિહ્ન < અને > જુદી જ રીતે છપાય છે. (મેથ મોડ સિવાય) તેને લખવા માટે **\textless** અને **\textgreater** લખવા પડે છે. ` (ગ્રેવ એસેન્ટ અથવા બેકકોટ) અને ' (એપોસ્ટ્રોફ અથવા સ્ટ્રેઇટ કોટ)નો ઉપયોગ લખાણને એક અવતરણચિહ્નમાં લખવા માટે થાય છે. દા.ત., `Book Code'. બે અવતરણચિહ્ન કરવા માટે આ ચિહ્ન બે વખત લખવાં પડે છે. દા.ત. ``Book Code''. (આ બે વખત લખાયેલા સ્ટ્રેઇટ કોટ છે, એક વખત લખાયેલ ડબલ કોટ નથી.) તે ફાઈલની અંદર વિચિત્ર રીતે દેખાય છે પરંતુ આઉટપુટમાં સરખી રીતે દેખાય છે.

લેટેક્સ લખાણના ભાગને ચિહ્નિત કરવા માટે જૂથનો ઉપયોગ કરે છે. આવાં જૂથ છગડિયા કૌંસ { અને }માં લખવામાં આવે છે. આવા જૂથમાં આપવામાં આવેલા કોઈ પણ કમાન્ડ ફક્ત જૂથમાં રહેલ કમાન્ડ પછીના લખાણને જ અસર કરે છે. અને કેટલાક કમાન્ડ જૂથમાં હોય છે જે આખા જૂથને લાગુ પડે છે. જૂથ એ ટૂંકું લખાણ જેવું કે લીટીનો ભાગ, થોડી લીટીઓ કે ફકરાને જૂજ – થોડા કમાન્ડ એકસાથે લાગુ પાડવા માટે ઉપયોગી છે.

જ્યારે મોટી સંખ્યામાં કમાન્ડ લાગુ પાડવા હોય (ઉદાહરણ તરીકે ટેબલની ગોઠવણી કરવી હોય અથવા ગાણિતિક સમીકરણ લખવું હોય) અથવા લખાણના કોઈ મોટા ભાગને ઘણા બધા કમાન્ડની અસર આપવી હોય (જેમ કે ઘણા બધા ફકરાઓ, આખો વિભાગ) ત્યારે લેટેક્સ એક અલગ સગવડ પૂરી પાડે છે. આ સગવડને એન્વાયર્નમેન્ટ તરીકે ઓળખવામાં આવે છે. આ એન્વાયર્નમેન્ટ \begin{environment-name} કમાન્ડથી શરૂ થાય છે અને \end{environment-name} કમાન્ડથી પૂરું થાય છે.

એન્વાયર્નમેન્ટમાં લખેલ બધા જ લખાણ ઉપર એન્વાયર્નમેન્ટની બધી જ (ફૉર્મેટિંગ) માળખાકીય લાક્ષણિકતા લાગુ પડે છે. એન્વાયર્નમેન્ટ નેસ્ટેડ પણ હોઈ શકે; એક એન્વાયર્નમેન્ટમાં બીજું એન્વાયર્નમેન્ટ હોય તેને નેસ્ટેડ એન્વાયર્નમેન્ટ કહેવાય. આવા કેટલાક વિષયવસ્તુના પ્રકાર પ્રમાણે જુદા જુદા પ્રમાણભૂત એન્વાયર્નમેન્ટ છે. જેવા કે સમીકરણ (equation), અવતરણ (quotation), ટેબલ અને યાદી (list). આ બધા જ એન્વાયર્નમેન્ટ તેના વિષયવસ્તુ પ્રમાણે તૈયાર ફૉર્મેટિંગ કમાન્ડ સાથે આવે છે.

લેટેક્સના દસ્તાવેજને છાપી શકાય છે તેમજ પ્રદર્શિત કરી શકાય છે. લેટેક્સમાં જુદાં-જુદાં ઘણાંબધાં ફિચર (લાક્ષણિકતા) છે. જેમકે પ્રોગ્રામિંગ કરવું, દસ્તાવેજનો કોઈક ભાગ સ્વયંસંચાલિત બનાવવો અથવા એક કરતા વધુ દસ્તાવેજો બનાવવા (મેઇલમર્જ) તેમજ અન્ય વ્યક્તિ અથવા ટીમ દ્વારા બનાવેલ ટેમ્પલેટ અથવા પૅકેજનો ઉપયોગ કરવો. કેટલીક વખત આ વ્યક્તિને પણ લેટેક્સનો કોડ ખેંચવાની જરૂર પડે છે.

આવા કિસ્સામાં જટિલ ભાગનું વિગતવાર વિશ્લેષણ આપવાથી આવા કોડને બીજી વ્યક્તિ સરળતાથી સમજી શકે છે. આવુ વિગતવાર વિશ્લેષણ કૉમેન્ટના ભાગ રૂપે આપી શકાય. લેટેક્સમાં % ચિહ્નથી કૉમેન્ટ શરૂ કરી શકાય છે. % પછીનું તમામ લખાણ, લીટીના અંત સુધીનું કૉમેન્ટ તરીકે ગણાય છે. કૉમેન્ટ એ એડિટરમાં રહેલ કોડને વાંચવા માટે, સમજવા માટે અને જરૂર જણાય ત્યાં સુધારાવધારા કરવા માટે ઉપયોગકર્તાને ઉપયોગી છે. કંપાઇલેશનમાં કૉમેન્ટને અવગણવામાં આવે છે તેમજ આઉટપુટમાં પણ કૉમેન્ટનું કોઈ સ્થાન હોતુ નથી.

લેટેક્સ દસ્તાવેજનું માળખું (The Structure of a LaTex Document)

લેટેક્સ દસ્તાવેજના બે ભાગ છે : પ્રસ્તાવના અને વિષયવસ્તુ. પ્રસ્તાવનામાં મેટાડેટા (માહિતીની માહિતી) હોય છે. મેટાડેટા એ દસ્તાવેજ વિશેની માહિતી છે. (ઉદાહરણ તરીકે, આ કેવા પ્રકારનો દસ્તાવેજ છે, દસ્તાવેજના લેખક કોણ છે, દસ્તાવેજ ક્યારે બનાવેલ છે) અને લેટેક્સ દસ્તાવેજની પ્રક્રિયા કઈ રીતે કરશે, તેની સૂચના હોય છે. મૂળ વિષયવસ્તુ હંમેશાં એન્વાયર્નમેન્ટ દસ્તાવેજમાં હોય છે, જે \begin{document} અને \end{document} કમાન્ડની વચ્ચે લખાયેલ હોય છે.

પ્રાસ્તાવિક ભાગ (The Preamble)

લેટેક્સમાં વિવિધતાસભર ઘણાબધા જુદી-જુદી લાક્ષણિક્તા અને માળખાવાળા દસ્તાવેજ બનાવી શકાય છે. લેટેક્સે એ જાણવું જરૂરી છે કે મૂળ ફાઇલ તરીકે ક્યા પ્રકારનો દસ્તાવેજ ઉપયોગમાં આવે છે. પ્રસ્તાવનાનું સૌથી પહેલું તત્ત્વ (એલિમેન્ટ) **\documentclass**{document-class-name} હોવું જ જોઈએ. જે દસ્તાવેજનો પ્રકાર જણાવે છે. કેટલાક સામાન્ય દસ્તાવેજના ક્લાસ કોષ્ટક 12.1માં આપેલ છે. કેટલાક દસ્તાવેજ-ક્લાસને વિકલ્પો પણ હોય છે. કોષ્ટક 12.2માં આવા સામાન્ય વિકલ્પ આપેલ છે.

દસ્તાવેજ-ક્લાસ	ઉદ્દેશ (ઉપયોગ)
article	સ્વતંત્ર લેખ લખવા માટે
book	આખું પુસ્તક લખવા માટે
slides	પ્રદર્શન માટેની સ્લાઇડ બનાવવા માટે, આમાં ફૉન્ટનું કદ આપોઆપ મોટું થઈ જાય છે
letter	પત્ર લખવા માટે
beamer	બીમર પૅકેજનો ઉપયોગ કરીને ઑફિસસ્યુટ જેવું જ પ્રદર્શન બનાવવા માટે

કોષ્ટક 12.1 : કેટલાક સામાન્ય દસ્તાવેજ-ક્લાસ

વિકલ્પ	કાર્ય
10pt, 11pt, 12pt	દસ્તાવેજના મુખ્ય ફૉન્ટનું કદ 10 પૉઇન્ટ (પૂર્વનિર્ધારિત), 11 પૉઇન્ટ કે 12 પૉઇન્ટ રાખવા માટે.
a4paper, letterpaper, legalpaper	પાનાંનું કદ નક્કી કરવા માટે. આ બધા જુદાં-જુદાં આંતરરાષ્ટ્રીય પાનાંનાં કદ છે. આમાંનું ઑફિસના ઉપયોગ માટેનું A4, લેટર અને લીગલ એ સામાન્ય કદ છે.
fleqn	સમીકરણો અને રચનાઓને વચ્ચે રાખવાને બદલે ડાબી બાજુ (લેફ્ટ અલાઇન) દર્શાવવા માટે
landscape	દસ્તાવેજને લૅન્ડસ્કેપમાં છાપવા માટે (આડા પાનાંમાં છાપવા માટે)

કોષ્ટક 12.2 : કેટલાક દસ્તાવેજ-ક્લાસના કેટલાક સામાન્ય વિકલ્પ

વૈકલ્પિક પૅકેજની જાહેરાત પછી દસ્તાવેજના ક્લાસને જાહેર કરવામાં આવે છે. લેટેક્સ પદ્ધતિ પોતે જ કેટલીક સામાન્ય ટાઇપસેટિંગની જરૂરિયાત પૂરી પાડે છે, પરંતુ ઉપયોગકર્તાને જોઈતું જરૂરી બધું જ પૂરું પાડતું નથી. આથી વધારાની સગવડ માટે લેટેક્સ ઉપયોગકર્તાને પોતાનું પૅકેજ બનાવવાની સવલત આપે છે. લેટેક્સના ઉપયોગકર્તાનો મોટો સમૂહ છે, જે આવા નવાં પૅકેજ બનાવે છે અથવા અસ્તિત્વમાં હોય એવા પૅકેજમાં પોતાની જરૂરિયાત મુજબ ફેરફારો કરીને તેને Comprehensive Tex Archive Network (CTAN) ઉપર વહેંચે છે.

લેટેક્સમાં આવાં ઘણાંબધાં પૅકેજ પહેલેથી જ પ્રસ્થાપિત કરેલાં હોય છે. આપણા દસ્તાવેજમાં આવાં એક અથવા એકથી વધારે પૅકેજનો ઉપયોગ કરવો હોય, તો તેને પ્રાસ્તાવિક ભાગ (પ્રસ્તાવનામાં) \usepackage{package-name} કમાન્ડ વડે જાહેર (ડિક્લેર) કરવાં પડે છે. કેટલાંક પૅકેજને તેનું વર્તન નક્કી કરવા માટે વિકલ્પ પણ હોય છે. જો આમાંના કોઈ પણ વિકલ્પનો ઉપયોગ આપણે ન કરવો હોય, તો એક જ લાઇનમાં એક જ \usepackage કમાન્ડ વડે એક કરતાં વધારે પૅકેજને અલ્પવિરામ આપીને જાહેર (ડિક્લેર) કરી શકાય. કોષ્ટક 12.3માં કેટલાંક સામાન્ય ઉપયોગમાં આવતાં પૅકેજ દર્શાવવામાં આવ્યા છે.

પૅકેજ	વર્ણન
amsmath	આ પૅકેજમાં લેટેક્સમાં ગણિત માટેનાં વિસ્તરણ હોય છે. મૂળભૂત રીતે અમેરિકન ગણિત સોસાયટી માટે બનાવવામાં આવેલ.
color	લખાણના કલર ઉમેરવા માટે
easylist	એક કરતાં વધુ સ્તરની યાદી ઉમેરવા માટે
geometry	પાનાંની રચના માટે, જેમકે પાનાનું કદ નક્કી કરવું, સ્થાપન (ઓરિએન્ટેશન) નક્કી કરવું માર્જિન નક્કી કરવું વગેરે
listings	દસ્તાવેજમાં પ્રોગ્રામિંગ કોડ ઉમેરવા માટે
setspace	લીટી વચ્ચેની જગ્યા બદલવા માટે (લાઇનસ્પેસિંગ)

કોષ્ટક 12.3 : સામાન્ય રીતે ઉપયોગમાં આવતાં પૅકેજ

વધુમાં માહિતીના ત્રણ ઘટકો પૂરા પાડવામાં આવે છે, જે નીચે દર્શાવેલ છે :

\title{the-title-of-the-document} – દસ્તાવેજનું શીર્ષક

\author{author(s) of the document} – દસ્તાવેજના લેખક

\date{date of creation / last update of the document, in any format} – દસ્તાવેજ બનાવ્યાની તારીખ

જો લેટેક્સ તમારા માટે સ્વયંસંચાલિત રીતે શીર્ષકનું સર્જન કરે તેમ તમે ઇચ્છતા હો, તો તમારે શીર્ષક અને લેખકની માહિતી પૂરી પાડવી જરૂરી છે, પરંતુ તારીખ દર્શાવવી મરજિયાત છે. જો આ માહિતી પૂરી પાડવામાં નહિ આવે તો, કમ્પાઇલેશનની તારીખને શીર્ષક તરીકે લેવામાં આવશે. આ માહિતી પ્રસ્તાવનામાં અથવા તો દસ્તાવેજ એન્વાયર્નમેન્ટમાં પ્રથમ વસ્તુ તરીકે આપવામાં આવે છે.

દસ્તાવેજ એન્વાયર્નમેન્ટ (The Document Environment)

લેખ અને સ્લાઇડ માટેના દસ્તાવેજ એન્વાયર્નમેન્ટને ફક્ત દસ્તાવેજના મુખ્ય વિષયવસ્તુને અનુસરતું શીર્ષક હશે. આ શીર્ષક લેટેક્સ દ્વારા સ્વયંસંચાલિત રીતે અપાશે. જ્યારે તે \maketitle કમાન્ડ મેળવશે, અલબત્ત \title, \author અને \date કમાન્ડ્સ \maketitle પહેલાં અપાઈ જવા જોઈએ. કારણકે તે માહિતી શીર્ષકના સર્જનમાં ઉપયોગી બને છે.

એક પુસ્તકમાં ઘણા વિસ્તૃત વિભાગ હોઈ શકે અલબત્ત તેમાંના મોટા ભાગના સ્વૈચ્છિક હોઈ શકે. પુસ્તકના દસ્તાવેજ એન્વાયર્નમેન્ટને મુખ્ય ત્રણ ભાગમાં વિભાજિત કરી શકાય - આગળની વિગત, મુખ્ય વિગત અને પાછળની વિગત. આ માટે અનુક્રમે \frontmatter, \mainmatter અને \backmatter કમાન્ડ આપવામાં આવે છે. આ વિવિધ વિભાગોમાં અલગ-અલગ વસ્તુઓ હોઈ શકે, જેમકે આગળની વિગતનું શીર્ષક, વિષયવસ્તુનું કોષ્ટક અને પ્રસ્તાવના તેમજ પાછળની વિગતની ગ્રંથસૂચિ, અનુક્રમણિકા અને સંદર્ભ સૂચિ અને મુખ્ય વિગતમાં પ્રકરણોના સ્વરૂપમાં પ્રાથમિક વિષય વસ્તુ, વિભાગો અને પેટા વિભાગો.

પુસ્તકના મુખ્ય વિષયવસ્તુને સ્તરીકરણ માળખુ હોય છે. જ્યાં પુસ્તકનું વિભાગોમાં, વિભાગોનું પ્રકરણમાં, પ્રકરણોનું મુદ્દાઓમાં, મુદ્દાઓનું પેટા મુદ્દાઓમાં, પેટા મુદ્દાઓનું પેટા-પેટા મુદ્દાઓમાં, પેટા-પેટા મુદ્દાઓનું ફકરાઓમાં, ફકરાઓનું પેટા ફકરાઓમાં વિભાજન કરવામાં આવે છે. આ માટે અનુક્રમે \part, \chapter, \section, \subsection, \subsubsection, \paragraph અને \subparagraph કમાન્ડ આપવામાં આવે છે.

દરેક કમાન્ડમાં ફરજિયાત એક આર્ગ્યુમેન્ટ આપવામાં આવે છે. શીર્ષક તથા એક વૈકલ્પિક આર્ગ્યુમેન્ટ વિષયવસ્તુના કોષ્ટકનું શીર્ષક, જે નવો વિભાગ, પ્રકરણ કે મુદ્દો શરૂ કરે છે. મુખ્ય લખાણમાં દર્શાવવામાં આવતું શીર્ષક લાંબું હોઈ શકે તેમજ તેને આંતરિક ફૉર્મેટિંગ પણ હોઈ શકે (જેમકે અક્ષરોને ઘાટા કરવા (બોલ્ડ) અથવા અક્ષરોને ત્રાંસા કરવા (ઇટાલિક)) જ્યારે વિષયવસ્તુના કોષ્ટકને આપવામાં આવેલ શીર્ષક ટૂંકું હોય તેમજ લખાણની સાતત્યતા જાળવવા માટે તેને ખાસ પ્રકારનું કાંઈ ફૉર્મેટિંગ આપવામાં આવેલું હોતું નથી. આ 7 ઘટકો એકબીજાની અંદર આવેલા હોય છે તેમજ દરેક સ્તરને પૂર્ણાંકનંબર આપવામાં આવેલ હોય છે, જેમકે વિભાગને સ્તર 1, પ્રકરણને સ્તર 0, મુદ્દાને સ્તર 1 વગેરે.

લેટેક્સ દ્વારા વિભાગ, પ્રકરણ અને મુદ્દાઓને સ્વયંસંચાલિત અનુક્રમનંબર આપવામાં આવે છે. આ માટે લેખકે ચિંતા કરવાની હોતી નથી. લેખક પ્રકરણ, મુદ્દા-પેટા મુદ્દાઓના ક્રમને ગમે ત્યારે ફેરવી શકે છે. આ માટે ફરી વખત અનુક્રમનંબર આપવાની તસ્દી લેવાની જરૂર રહેતી નથી. વિભાગોના અનુક્રમ રોમનમાં અપાય છે (I, II, III, વગેરે), જ્યારે પ્રકરણ, મુદ્દાઓ અને પેટા મુદ્દાઓ વગેરેના અનુક્રમ અરેબિકમાં અપાય છે (1, 2, 3, વગેરે).

\appendix કમાન્ડ પછીનાં (જે એક જ વખત આપી શકાય છે) બધાં જ પ્રકરણો પરિશિષ્ટો તરીકે સમજવામાં આવે છે અને તેમને બધાને અનુક્રમનંબર મોટા મૂળાક્ષર વડે આપવામાં આવે છે (A, B, C વગેરે). આગળના મુદ્રણનાં પાનાંઓને રોમનમાં અનુક્રમનંબર આપવામાં આવે છે, જ્યારે મુખ્ય મુદ્રણ અને પાછળના મુદ્રણમાં અરેબિકમાં અનુક્રમનંબર આપવામાં આવે છે, જે 1થી શરૂ થશે. આગળના મુદ્રણ અને પાછળના મુદ્રણ બંનેમાં પ્રકરણ હોય છે (જેવા કે પ્રસ્તાવના, કબૂલાતનામું, ગ્રંથસૂચિ). આવાં પ્રકરણોને સામાન્ય રીતે મુદ્દાઓ કે પેટાઘટકો હોતા નથી. દસ્તાવેજના જુદા ઘટકને દર્શાવતા કમાન્ડમાં નંબરની જગ્યાએ તારો (*) પણ હોઈ શકે. ઉદાહરણ તરીકે \section* કમાન્ડ અનુક્રમનંબર વગરનો મુદ્દો બનાવવા માટે વાપરી શકાય.

મૂળભૂત રીતે અનુક્રમનંબર 2 સ્તર સુધી આપી શકાય છે. એટલે કે પેટા મુદ્દા સુધી આપી શકાય છે. પેટા-પેટા મુદ્દાઓ અને ત્યાર પછીના ભાગને અનુક્રમનંબર આપવામાં આવતા નથી. આને બદલવા માટે લેટેક્સની પ્રસ્તાવનાના આંતરિક ગણકમાં ફેરફાર કરીને બદલી શકાય. ઉદાહરણ તરીકે \setcounter{secnumdepth}{3} કમાન્ડ ઘટકને સ્તર 3 (પેટા-પેટા મુદ્દા) સુધી અનુક્રમનંબર આપે છે. પૂર્ણવિરામ (.) અને ઘટકના નંબર વડે મુખ્ય ઘટકના ઘટકોને અનુક્રમનંબર આપી શકાય. આમાં પ્રકરણ એક અપવાદ છે. તેને આપેલ અનુક્રમનંબરને કોઈ વિભાગનંબર કે પૂર્ણવિરામ હોતું નથી. પ્રકરણને સાદો અરેબિક નંબર આપેલો હોય છે. જો એક પુસ્તકના વિભાગ IIમાં પ્રકરણ 5 અને તેમાં મુદ્દા 4ના પેટા મુદ્દા 1 હોય, તો આ પેટા મુદ્દાને 5.4.1 અનુક્રમનંબર આપવામાં આવે છે.

\tableofcontents કમાન્ડ આપવામાં આવે છે, ત્યારે એક સુવ્યવસ્થિત ગોઠવેલ વિષયવસ્તુનું કોષ્ટક (TOC) શીર્ષક ઘટક દ્વારા લેટેક્સમાં આપમેળે બને છે. ફરી વખત, વિષયવસ્તુના કોષ્ટકનું સ્તર 2 સુધી હોય છે (પેટા મુદ્દા સુધી). પરંતુ આને tocdepth આંતરિક ગણક વડે બદલી શકાય છે.

અહીં એક નોંધવા જેવો મહત્ત્વનો મુદ્દો એ છે કે લેટેક્સ સૉર્સફાઇલની શરૂથી અંત સુધી ક્રમશઃ એક જ વખતમાં પ્રક્રિયા કરે છે અને ક્રમશઃ આઉટપુટ ફાઇલ પણ બનાવે છે. તે ફાઇલમાં આગળ અને પાછળ હરતું-ફરતું નથી. આ એક સમસ્યા છે. દસ્તાવેજમાં વિષયવસ્તુનું કોષ્ટક પહેલાં અને ત્યાર પછી પાનાંનંબર સાથે પ્રકરણ અને મુદ્દાઓ આઉટપુટમાં જોવામાં મળે છે. જોકે આ તબક્કે લેટેક્સને પ્રકરણ કે મુદ્દાઓ વિશે માહિતી હોતી નથી. આવી પરિસ્થિતિમાં લેટેક્સને ઘણી વખત રન કરવું પડે છે.

પ્રથમ રનમાં લેટેક્સ દસ્તાવેજના માળખા વિશેની માહિતી ભેગી કરે છે અને એક પૂરક ફાઇલમાં સંગ્રહ કરે છે. આ તબક્કે આઉટપુટ ફાઇલમાં TOC ખાલી હોય છે અથવા જૂની પૂરક ફાઇલની માહિતી દર્શાવે છે. બીજા રનમાં તે પૂરક ફાઇલમાંથી સાચી માહિતી એકઠી કરીને શરૂઆતમાં સાચું TOC (વિષયવસ્તુનું કોષ્ટક) બનાવે છે.

TOCની જેમજ લેટેક્સ આકૃતિની યાદી, કોષ્ટકની યાદી, અન્યોન્ય સંબંધ (એક પુસ્તકમાં બીજા ફકરાનો નિર્દેશ), ગ્રંથ-સૂચિ, પારિભાષિક શબ્દકોશ અથવા અનુક્રમણિકા રાખે છે. આ બધી જ વસ્તુ લેખકનો ભાર હળવો કરે છે અને આના જ કારણે લેટેક્સ પ્રખ્યાત છે.

ઉદાહરણ

લેટેક્સ પુસ્તકને કઈ રીતે ગોઠવે છે, તે જોવા માટે આપણે એક સૉર્સફાઇલ બનાવીએ અને તેનું આઉટપુટ જોઈએ. આ ઉદાહરણમાં આપણે લેટેક્સ સાથે SciTE એડિટરનો ઉપયોગ કરીશું.

- SciTE એડિટરમાં **File → New** મેનુવિકલ્પ પસંદ કરી નવી ફાઇલ બનાવો.
- લિસ્ટિંગ 12.1માં આપેલ માહિતી SciTE એડિટરમાં (ટાઇપ કરો) લખો.

```
\documentclass[12pt]{book}
\usepackage{amsmath}
\title{\huge Mathematics \\[3\baselineskip]
\Large Standard 12}
\author{Gujarat State Board of School Textbooks}
\date{2013}
\setcounter{secnumdepth}{2}
\setcounter{tocdepth}{1}
 \begin{document}
 \frontmatter
  \maketitle
  \chapter{\MakeUppercase{Fundamental Duties}}
  \tableofcontents
  \chapter{\MakeUppercase{About This Textbook...}}
\mainmatter
 \part{Semester I}
 \chapter{Set Operations}
 \section{Introduction}
 \section*{Exercise 1.1}
 \section{Properties of the Union Operation}
 \subsection{Union is a Binary Operation}
 \section{Properties of the Intersection Operation}
 \subsection{Intersection is a Binary Operation}
 \subsection{Associative Law}
 \chapter{Number Systems}
 \section{Introduction}
 \section*{Exercise 2.1}
 \section{Irrational Numbers}
 \chapter{Polynomials}
 \chapter{Coordinate Geometry}
 \chapter{Some Primary Concepts in Geometry : 1}
 \chapter*{Answers}
 \markboth{\MakeUppercase{Answers}}{}
 \addcontentsline{toc}{chapter}{Answers}
 \part{Semester II}
 \chapter{Quadrilaterals}
```

```
\section{Introduction}
\section{Plane Quadrilateral}
\chapter{Areas of Parallelograms and Triangles}
\section{Introduction}
\section{Interior of a Triangle}
\chapter{Circle}
\chapter{Surface Area and Volume}
\chapter*{Answers}
\markboth{\MakeUppercase{Answers}}{}
\addcontentsline{toc}{chapter}{Answers}
\appendix
\chapter{Terminology}
\backmatter
\end{document}
```

લિસ્ટિંગ 12.1 : લેટેક્સમાં પુસ્તકનો એક નમૂનો

- **File → Save** મેનુવિકલ્પ પસંદ કરી ફાઇલને સેવ કરો. અહીં એ નોંધી લેશો કે ફાઇલનું અનુલંબન .tex હોવું જોઈએ.
- લેટેક્સ ફાઇલને કમ્પાઇલ કરવા માટે **Tools → Build** વિકલ્પ પસંદ કરો. (શૉર્ટકટ કી : F7) આઉટપુટ વિન્ડોમાં ઘણાબધા સંદેશાઓ દેખાશે. જો આ સંદેશાઓમાં છેલ્લી લીટી (વાદળી કલરમાં) Exit code: 0 હોય તો આપણું કમ્પાઇલેશન સફળ થયું છે. નહિ તો આપણને ભૂલસંદેશ મળશે. પરંતુ ઘણી વાર તેનું અર્થઘટન કરવું મુશ્કેલ છે.
- જો કમ્પાઇલેશન સફળ થયું હોય તો, ડૉક્યુમેન્ટ વ્યૂઅરમાં ફાઇલ જોવા માટે **Tools → Go** મેનુવિકલ્પ પસંદ કરો. (શૉર્ટકટ કી : F5) SciTEમાં પરત ફરતા પૂર્વે ડૉક્યુમેન્ટ વ્યૂઅર બંધ કરવાનું ભૂલશો નહિ.

અહીં એ નોંધી લેશો કે આપણે લિસ્ટિંગ 12.1માં મૂળભૂત બુકશૈલીમાં કેટલાક ફેરફારો કર્યા છે, જેમાંની કેટલીક લાક્ષણિકતાઓની ચર્ચા અહીં કરી નથી. લેટેક્સ ટાઇપસેટિંગ ભારે વૈવિધ્યપૂર્ણ છે.

લખાણ ફૉર્મેટિંગ (Text Formatting)

આપણે લેટેક્સ દસ્તાવેજમાં આખેઆખો ફકરો એન્ટર કી દબાવ્યા વગર લખી શકીએ છીએ. લેટેક્સ ત્યાર પછી આ લખાણને ગોઠવે છે. લેટેક્સ પાનાની પહોળાઈ, ફૉન્ટનું કદ અને ગોઠવણીના વિકલ્પના આધારે નક્કી કરે છે કે કેટલું લખાણ પહેલી લીટીમાં, કેટલું લખાણ બીજી લીટીમાં અને કેટલું લખાણ ત્યાર પછી આવશે. લેટેક્સ એક શબ્દને બે ભાગમાં વિભાજિત કરવાનું ટાળે છે. જો ફરજિયાત રીતે એક શબ્દને બે ભાગમાં વિભાજિત કરવો પડે, તો તે બંને ભાગોને જોડવા માટે (-) સંયોગ ચિહ્નનો ઉપયોગ કરે છે. ઉદાહરણ તરીકે formatting નામનો શબ્દ એક લીટીમાં ન સમાઈ શકતો હોય, તો પહેલી લીટીમાં formatt– અને ત્યાર પછીની લીટીમાં ingથી શરૂ થાય છે.

જ્યારે બીજી તરફ, કેટલીક પરિસ્થિતિ કે લખાણમાં કેટલાક તકનિકી બહુવિધ શબ્દોને અલગ-અલગ લીટીમાં વિભાજિત કરી શકાતા હોતા નથી. ઉદાહરણ તરીકે up to બે શબ્દો છે, પરંતુ એક લીટીના અંતમાં up અને બીજી લીટીની શરૂઆતમાં to લખવું અપેક્ષિત નથી. કારણ કે બંને શબ્દો સ્વતંત્ર છે અને બંનેનો અર્થ અલગ છે. જો આમ કરવામાં આવે, તો વાચક બીજી લીટીની શરૂઆતમાં to શબ્દ જોઈને આશ્ચર્ય પામશે.

આ એક સમતલ વાંચનમાં વિઘ્નરૂપ અનુભવ છે. આ રીતને અવગણવા માટે up અને to બંને શબ્દો કોઈ એક લીટીમાં જ હોવા જોઈએ. કાં તો પહેલી લીટીમાં અથવા બીજી લીટીમાં લેટેક્સમાં આવું કરવા માટે બે શબ્દો વચ્ચે મૂકી ન શકાય તેવી જગ્યા ~ (ટાઇલ્ડ) અક્ષર વડે દર્શાવવામાં આવે છે.

લેટેક્સમાં ફૉન્ટને ત્રણ વર્ગમાં વિભાજિત કરવામાં આવે છે. રોમન (સેરિફ પણ કહી શકાય), સેન્સ સેરિફ અને મોનોસ્પેસ. રોમન ફૉન્ટને રેખાના અંતમાં ટૂંકી આડી લીટી હોય છે. સેન્સ સેરિફને આવી ટૂંકી આડી લીટી હોતી નથી જ્યારે મોનોસ્પેસ ફૉન્ટ બધા જ અક્ષરો માટે સમાન પહોળાઈ ધરાવે છે. સામાન્ય રીતે મોનોસ્પેસ ફૉન્ટનો ઉપયોગ કમ્પ્યૂટરમાં કોડ લખવા માટે થાય છે. જ્યારે રોમન ફૉન્ટ એ મૂળભૂત ફૉન્ટ છે. આ ત્રણેય પ્રકારના ફૉન્ટનો ઉપયોગ કરવા માટે \textrm{text}, \textsf{text} અને \texttt{text} કમાન્ડ વાપરી શકાય. આકૃતિ 12.2માં તમે સેરિફ અને સેન્સ સેરિફ ફૉન્ટ વચ્ચેનો તફાવત જોઈ શકશો, જેમાં શીર્ષકનો વિભાગ મૂળભૂત સેરિફ ફૉન્ટમાં અને બોડીનો વિભાગ સેન્સ સેરિફ ફૉન્ટમાં છે.

\tiny, \scriptsize, \footnotesize, \small, \normalsize, \large, \LARGE, \huge અને \Huge કમાન્ડ વડે ફૉન્ટના કદમાં ફેરફાર કરી શકાય. અહીં એ નોંધી લેશો કે કમાન્ડ કેસ-સેન્સિટિવ છે. \textbf, \textit અને \emph કમાન્ડનો ઉપયોગ લખાણમાં ઘાટું (બોલ્ડ), ત્રાંસા (ઇટાલિક) કે ભાર દર્શાવતી (સામાન્ય રીતે ત્રાંસા જેવી જ) અસર આપવા માટે થાય છે. \textsc કમાન્ડનો ઉપયોગ નાના કદમાં મોટા અક્ષરો (કૅપિટલ અક્ષર) દર્શાવવા થાય છે જ્યારે fixltx2e પૅકેજના \textsuperscript અને \textsubscript કમાન્ડ વડે લખાણને સુપરસ્ક્રિપ્ટ (એક ચિહ્ન ઉપર કરેલું બીજું ચિહ્ન) અને સબસ્ક્રિપ્ટ (એક ચિહ્નની નીચે કરેલું બીજું ચિહ્ન)માં લખી શકાય છે.

ફકરાની ગોઠવણી (Paragraph Formatting)

લેટેક્સમાં બે લીટીની વચ્ચે અંતર રાખવા માટે setspace પૅકેજમાં singlespace, onehalfspace, doublespace અને spacing{amount-of-spacing} એન્વાયર્નમેન્ટ ઉપલબ્ધ છે. લેટેક્સમાં મૂળભૂત રીતે બોડીનું લખાણ સંપૂર્ણ ઉચિત ગોઠવાયેલ (જસ્ટિફાઈ) હોય છે. જો લખાણ ડાબી તરફ, જમણી તરફ અથવા વચ્ચે સીધી લીટીમાં ગોઠવવું હોય, તો flushleft, flushright અને center એન્વાયર્નમેન્ટનો ઉપયોગ કરવો પડે. મથાળાની નીચે આપેલ ફકરાને બાદ કરતાં બાકીના ફકરાની પહેલી લીટી આરંભમાં જગ્યા છોડીને લખેલ હોય છે.

ફકરાની શરૂઆતમાં બાહ્ય રીતે \indent અને \noindent કમાન્ડનો ઉપયોગ કરીને, પહેલી લીટીના આરંભમાં જગ્યા છોડવી કે ન છોડવી તે નક્કી કરી શકાય છે. verbatim એન્વાયર્નમેન્ટ પ્રક્રિયા કર્યા વગર આઉટપુટ જેમ છે તેમ (ખાસ અક્ષર, જગ્યા, નવી લાઇન અને લેટેક્સ કમાન્ડ સહિત) જ આપે છે. moreverb પૅકેજ-પ્રોગ્રામ કોડ સુધીમાં લીટીને નંબર આપવા માટે એક ફરજિયાત આર્ગ્યુમેન્ટ-પહેલી લીટીનો અનુક્રમનંબર-સાથે એક સૂચિકરણ (લિસ્ટિંગ) એન્વાયર્નમેન્ટ પૂરું પાડે છે.

પૃષ્ઠ લે-આઉટ (Page Layout)

લેટેક્સમાં geometry પૅકેજનો ઉપયોગ કરીને પાનાંને લેઆઉટ આપી શકાય છે. \usepackage કમાન્ડમાં પાનાંનું કદ અને માર્જિન વૈકલ્પિક આર્ગ્યુમેન્ટ તરીકે પાસ કરી શકાય છે. દા.ત., નીચે આપેલો કમાન્ડ પાનાનું કદ A4 ઉપરનું માર્જિન 1 ઇંચ, નીચેનું માર્જિન 2 ઇંચ, ડાબી બાજુનું માર્જિન 1.5 ઇંચ અને જમણી બાજુનું માર્જિન 1 ઇંચ ગોઠવે છે.

```
\usepackage[a4paper, top=1in, bottom=2in, left=1.5in, right=1in]{geometry}
```

પાનાંનું કદ આંતરરાષ્ટ્રીય પ્રમાણિત હોય છે. સામાન્ય રીતે નિયમિત વપરાતા પ્રિન્ટરમાં પાનાનું કદ A4, લેટર અથવા લીગલ હોય છે. તેમને અનુક્રમે a4paper, letterpaper અને legalpaper વડે દર્શાવી શકાય છે. portrait (મૂળભૂત) અને landscape વિકલ્પ વડે પાનાનું ઓરિએન્ટેશન ગોઠવી શકાય છે.

દસ્તાવેજો એક બાજુ અથવા બે બાજુ છપાયેલા હોય છે. લેખ મૂળભૂત એક બાજુ, જ્યારે પુસ્તકો બંને બાજુ છપાયેલ હોય છે. બંને બાજુ છપાયેલ દસ્તાવેજને ડાબા પાનાં (બેકી) તથા જમણા પાના (એકી)માં અલગ અલગ માર્જિન છોડીને

જુદા પાડી શકાય છે. બાઈન્ડિંગ (ચોપડીનું વેષ્ટન) કરવા માટેની ખાલી જગ્યા પણ છોડવામાં આવે છે તેમજ એક એવો નિયમ પણ છે કે પ્રકરણની શરૂઆત એકી પાના ઉપરથી થવી જોઈએ.

લેટેક્સમાં ગણિતશાસ્ત્રની સામગ્રીનું ટાઈપસેટિંગ (Typesetting Mathematical Content in LaTeX)

લેટેક્સમાં જટિલ ગાણિતિક સામગ્રીને આપોઆપ ગોઠવી આપવાની ક્ષમતા છે. અમેરિકન મેથેમેટિકલ સોસાયટીએ બનાવેલ amsmath, amssymb અને amsfonts પૅકેજનો ઉપયોગ કરીને ગાણિતિક સામગ્રીને ગોઠવવાનો એક સર્વસામાન્ય રસ્તો લેટેક્સમાં છે.

amsmath પૅકેજ ગાણિતિક સામગ્રી માટેના ઘણાંબધાં એન્વાયર્નમેન્ટ વ્યાખ્યાયિત કરે છે. ટાઈપસેટિંગ માટેના બે રસ્તાઓ છે. સૂત્ર અને સમીકરણ. જેને આપણે એક લીટીના ભાગમાં અથવા તો સ્વતંત્ર લીટીમાં છાપવાના હોય છે. math એન્વાયર્નમેન્ટનો ઉપયોગ કરીને પહેલા આ સ્વરૂપ મેળવી શકાય છે, ત્યાર પછીથી displaymath એન્વાયર્નમેન્ટનો ઉપયોગ કરવો પડે છે.

સમીકરણ (equation) એન્વાયર્નમેન્ટ એ એક display એન્વાયર્નમેન્ટ છે. જે સમીકરણના ક્રમ આપોઆપ દર્શાવે છે. લખાણમાં math એન્વાયર્નમેન્ટનો ઉપયોગ કરવાનો સરળ રસ્તો એ છે કે ગાણિતિક લખાણ સામગ્રીને $...$ની વચ્ચે લખવી. ગાણિતિક એન્વાયર્નમેન્ટમાં દરેક શબ્દને ગાણિતિક ચલ તરીકે ગણવામાં આવે છે. જે ટેક્સ્ટ (લખાણનું) એન્વાયર્નમેન્ટ કરતાં જુદુ છે. આ એન્વાયર્નમેન્ટને સરખી રીતે સમજવા માટે લિસ્ટિંગ 12.2માં આપેલ કોડની એક .tex ફાઈલ બનાવો.

- SciTEમાં **File → New** મેનુવિકલ્પની મદદ વડે નવી ફાઈલ બનાવો.
- લિસ્ટિંગ 12.2માં આપેલ લખાણને SciTE એડિટરમાં લખો.

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\title{Introduction to \LaTeX}
\date{May 2013}
\begin{document}
\section*{math environment}
The quadratic equation, in its general form, is
\begin{math}
ax^2 + bx + c = 0
\end{math}.
You learnt about them in class X.

The quadratic equation, in its general form, is $ax^2 + bx + c = 0$. You learnt about
them in class X.

\section*{displaymath environment}
The quadratic equation, in its general form, is
\begin{displaymath}
ax^2 + bx + c = 0
\end{displaymath}. You learnt about them in class X.

\end{document}
```

લિસ્ટિંગ 12.2 : Math એન્વાયર્નમેન્ટનું પ્રદર્શન

- **File → Save** મેનુવિકલ્પનો ઉપયોગ કરીને ફાઈલને સેવ કરો.
- હવે લેટેક્સ ફાઈલને કમ્પાઈલ કરવા માટે **Tools → Build** મેનુવિકલ્પ પસંદ કરો. (શૉર્ટકટ કી : F7)
- જો કમ્પાઈલેશન સફળતાપૂર્વક થયું હોય તો, મૂળભૂત ડૉક્યુમેન્ટવ્યૂઅરમાં ફાઈલ જોવા માટે **Tools → Go** મેનુવિકલ્પ પસંદ કરો. (શૉર્ટકટ કી : F5). આકૃતિ 12.3માં આ ફાઈલનું આઉટપુટ દર્શાવેલ છે.

math environment

The quadratic equation, in its general form, is $ax^2 + bx + c = 0$. You learnt about them in class X.

The quadratic equation, in its general form, is $ax^2 + bx + c = 0$. You learnt about them in class X.

displaymath environment

The quadratic equation, in its general form, is

$$ax^2 + bx + c = 0$$

. You learnt about them in class X.

આકૃતિ 12.3 : લિસ્ટિંગ 12.2માં આપેલ કોડના આઉટપુટનો એક ભાગ

ગાણિતિક ચિહ્નો (સંજ્ઞા)નો ઉપયોગ (Using Mathematical Symbols)

ગણિતશાસ્ત્ર ઘણાંબધાં ચિહ્નોનો ઉપયોગ કરે છે. અહીં આપણે લેટેક્સમાં ઉપયોગમાં આવતાં આવાં ગાણિતિક ચિહ્નોની ચર્ચા કરીશું. ગ્રીક મૂળાક્ષરના અક્ષરને તેના કમાન્ડ છે, જેવા કે **\alpha, \beta, \gamma, \pi** વગેરે નાના અક્ષર બનાવે છે, જ્યારે આ જ કમાન્ડનો પહેલો અક્ષર મોટો અક્ષર હોય, તો ઉદાહરણ તરીકે **\Alpha**નો તે ગ્રીક ભાષાનો મોટો અક્ષર બનાવે છે.

આ ઉપરાંત ગાણિતિક ચિહ્નો માટેના પણ કેટલાક કમાન્ડ છે. AMS પૅકેજનો ઉપયોગ કરીને આકૃતિ 12.4 અને આકૃતિ 12.5માં આવા કેટલાક લેટેક્સના કમાન્ડ અને તેનાં ગાણિતિક ચિહ્નો દર્શાવવામાં આવ્યા છે. ત્રિકોણમિતીનાં વિધેયો sin, cos અને બીજા બધા માટે કમાન્ડ દર્શાવવાનું કારણ એ છે કે તેને સ્વતંત્ર ચલ કરતાં વિધેય તરીકે ઓળખી શકાય એ રીતે ગોઠવવાની જરૂર હોય છે.

$<$	$<$	$>$	$>$
=	=	\leq	≤
\geq	≥	\neq	≠
\times	×	\div	÷
\pm	±	\mp	∓
\in	∈	\notin	∉
\supset	⊃	\subset	⊂
\supseteq	⊇	\subseteq	⊆
\cup	∪	\cap	∩

આકૃતિ 12.4 : લેટેક્સમાં વપરાતી કેટલીક ગાણિતિક સંજ્ઞાઓ

\cong	$\cong$	\propto	$\propto$
\rightarrow	$\rightarrow$	\parallel	$\parallel$
\leftrightarrow	$\leftrightarrow$	\leftarrow	$\leftarrow$
\angle	$\angle$	\bigodot	$\bigodot$
\triangle	$\triangle$	\overleftrightarrow{AB}	$\overleftrightarrow{AB}$
\stackrel\frown{AB}	$\stackrel\frown{AB}$	\overrightarrow{AB}	$\overrightarrow{AB}$
\overline{AB}	$\overline{AB}$	\perp	$\perp$
45^{\circ}	45°	\implies	$\implies$
\iff	$\iff$	\therefore	$\therefore$
\because	$\because$	\sin	$\sin$
\cos	$\cos$	\tan	$\tan$
\sec	$\sec$	\csc	$\csc$
\cot	$\cot$	\theta	θ

આકૃતિ 12.5 : લેટેક્સમાં વપરાતી વધુ કેટલીક ગાણિતિક સંજ્ઞાઓ

ગાણિતિક પ્રક્રિયકોનો ઉપયોગ (Using Mathematical Operators)

લેટેક્સમાં ઘણા બધા ગાણિતિક પ્રક્રિયકોનો ઉપયોગ થાય છે. પરંતુ પાવર શોધવા માટેનો પ્રક્રિયક (x2) અને ઇન્ડેક્સનો પ્રક્રિયક (x1) અલગથી આપવામાં આવતો નથી. આ માટે અનુક્રમે સામાન્ય સુપરસ્ક્રિપ્ટ પ્રક્રિયક ^ (કેરેટ અક્ષર) અને સામાન્ય સબસ્ક્રિપ્ટ પ્રક્રિયક _ (અન્ડરસ્કોર અક્ષર)નો ઉપયોગ થાય છે. સુપરસ્ક્રિપ્ટ પ્રક્રિયક લખાણને ઉપર તરફ અને સબસ્ક્રિપ્ટ લખાણને નીચે તરફ દર્શાવે છે. બંને પ્રક્રિયક લખાણના કદને ઘટાડે છે. સંપૂર્ણ કિંમત દર્શાવવા માટે પદાવલીને બે ઊભી લીટી | વચ્ચે લખવામાં આવે છે.

વિધેય બનાવવા માટે \frac{numerator}{denominator} કમાન્ડનો ઉપયોગ થાય છે, જ્યારે x સંખ્યાનું વર્ગમૂળ શોધવા માટે \sqrt{x} કમાન્ડનો ઉપયોગ થાય છે. આ પ્રક્રિયકો નેસ્ટેડ હોઈ શકે. એટલે કે વર્ગમૂળમાં ભાજકનો એક અપૂર્ણાંક બીજા અપૂર્ણાંકમાં અને ઉપરનો અંક બીજા અપૂર્ણાંકમાં હોઈ શકે વગેરે. આ માટે કોઈ મર્યાદા નથી. આ માટે ઘટકના કદ અને સ્થાનની કાળજી લેટેક્સ પોતે જ રાખે છે. આ માટે સામાન્ય રીતે કૌંસનો ઉપયોગ થાય છે. લિસ્ટિંગ 12.3માં ગાણિતિક પ્રક્રિયકોનો ઉપયોગ દર્શાવતું ઉદાહરણ આપેલ છે.

- **File → New** મેનુવિકલ્પનો ઉપયોગ કરી SciTEમાં નવી ફાઇલ બનાવો.
- લિસ્ટિંગ 12.3માં આપેલ લખાણ SciTE એડિટરમાં લખો.

```
\documentclass[12pt]{article}
\usepackage{amsmath}
\setlength{\parindent}{0pt}
\title{Introduction to \LaTeX}
\date{May 2013}
\begin{document}
\begin{math}
x^2 \\[6pt]
x_1 \\[6pt]
x_i \\[6pt]
x_i^2 \\[6pt]
x^y \\[6pt]
a^bc \\[6pt]
{a^b}^c \\[6pt]
a^{b^c} \\[6pt]
\frac{x}{y} \\[6pt]
\sqrt{b} \\[6pt]
\frac{\frac{1}{3}+\frac{3}{2}}{\frac{2}{3}+\frac{1}{2}} \\[6pt]
\frac{-b\pm\sqrt{b^2-4ac}}{2a} \\[6pt]
\frac{-b\pm\sqrt{\Delta}}{2a} \\[6pt]
\sqrt{1+\frac{1}{\sqrt{1+\frac{1}{\sqrt{1+\frac{1}{2}}}}}} \\[6pt]
\sqrt{(x_1-x_2)^2+(y_1-y_2)^2} \\[6pt]
|x-y| \\[6pt]
(\frac{x_1+x_2}{2},\frac{y_1+y_2}{2})
\end{math}
\end{document}
```

લિસ્ટિંગ 12.3 : ગાણિતિક પ્રક્રિયકોનું ઉદાહરણ

- **File → Save** મેનુવિકલ્પ વડે ફાઇલને સેવ કરો.
- હવે **Tools → Build** મેનુવિકલ્પ (શૉર્ટકટ કી : F7) પસંદ કરી તમારી લેટેક્સ ફાઇલને કમ્પાઇલ કરો.
- જો સફળતાપૂર્વક કમ્પાઇલેશન થયું હોય, તો **Tools → Go** મેનુ વિકલ્પ (શૉર્ટકટ કી : F5) વડે ડૉક્યુમેન્ટ વ્યૂઅરમાં ફાઇલ જુઓ. આકૃતિ 12.6 ડૉક્યુમેન્ટ વ્યૂઅરમાં ફાઇલનું આઉટપુટ દર્શાવે છે.

x^2	$\sqrt{b}$
x_1	$\frac{\frac{1}{3}+\frac{3}{2}}{\frac{2}{3}+\frac{1}{2}}$
x_i	$\frac{-b\pm\sqrt{b^2-4ac}}{2a}$
x_i^2	$\frac{-b\pm\sqrt{\Delta}}{2a}$
x^y	$\sqrt{1+\frac{1}{\sqrt{1+\frac{1}{\sqrt{1+\frac{1}{2}}}}}}$
a^bc	$\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$
a^{b^c}	$\lvert x-y\rvert$
a^{b^c}	$(\frac{x_1+x_2}{2},\frac{y_1+y_2}{2})$
$\frac{x}{y}$	

આકૃતિ 12.6 : લિસ્ટિંગ 12.3નું બે વિભાગમાં વિભાજિત આઉટપુટ

સમીકરણોનો ઉપયોગ (Using Equations)

લેટેક્સ સમીકરણો માટે એક ખાસ સમીકરણ (equation) એન્વાયર્નમેન્ટ પૂરું પાડે છે. દરેક સમીકરણને સમીકરણ (equation) એન્વાયર્નમેન્ટમાં લખવામાં આવે છે, તેને ગણિત (Math) એન્વાયર્નમેન્ટમાં જોડવામાં આવતાં નથી. સમીકરણોને આપમેળે અનુક્રમ નંબર આપવામાં આવે છે અને કેન્દ્રમાં ગોઠવવામાં આવે છે (center-aligned). લિસ્ટિંગ 12.4માં સમીકરણ (equation) એન્વાયર્નમેન્ટનું ઉદાહરણ દર્શાવવામાં આવ્યું છે. તેમજ આકૃતિ 12.7માં આઉટપુટનો ભાગ દર્શાવવામાં આવ્યો છે.

```
\documentclass[12pt]{article}
\setlength{\parindent}{0pt}
\usepackage{amsmath}
\title{Introduction to \LaTeX}
\date{May 2013}
\begin{document}
\begin{equation}
\sin^2\theta + \cos^2\theta=1
\end{equation}
\begin{equation}
```

```
\sec^2\theta - \tan^2\theta=1
\end{equation}
\begin{equation}
\csc^2\theta - \cot^2\theta=1
\end{equation}
\end{document}
```

લિસ્ટિંગ 12.4 : સમીકરણ (Equation) એન્વાયર્નમેન્ટનો ઉપયોગ

$$\sin^2\theta + \cos^2\theta = 1 \qquad (1)$$

$$\sec^2\theta - \tan^2\theta = 1 \qquad (2)$$

$$\csc^2\theta - \cot^2\theta = 1 \qquad (3)$$

આકૃતિ 12.7 : લિસ્ટિંગ 12.4નું આઉટપુટ

લેટેક્સના ફાયદાઓ (Advantages of LaTeX)

લેટેક્સના ઘણાબધા લાભ છે. ટેક્સને લેટેક્સમાં વિસ્તૃત કરવામાં આવેલ છે. લેટેક્સને પણ વિસ્તૃત કરી શકાય. લેટેક્સમાં નવાં લક્ષણો ઉમેરવા માટે અથવા વૈકલ્પિક અમલીકરણ માટે, લેટેક્સની સુવિધાઓ વધારવા માટે કોઈ પણ વ્યક્તિ વધારાના પૅકેજ બનાવી શકે છે. સમગ્ર વિશ્વમાં લેટેક્સ વપરાશકર્તાઓ દ્વારા વિવિધ જરૂરિયાતો પૂરી પાડવા માટે આવાં હજારો પૅકેજ બનાવવામાં આવ્યાં છે. આમાંનાં ઘણાંબધાં પૅકેજ નિઃશુલ્ક છે. તેને CTAN (The Comprehensive TeX Archive Network)ની વેબસાઈટ website at www.ctan.org ઉપર હોસ્ટ કરવામાં આવ્યાં છે.

લેટેક્સ જટિલ ગાણિતિક સૂત્રોને સરસ અને યોગ્ય રીતે ગોઠવવા માટે ખૂબ જ સારું છે. આ કારણે જ તે લેખકો, ગણિતશાસ્ત્રના પ્રકાશકો, ઇજનેરી, કમ્પ્યૂટરવિજ્ઞાન અને બીજા તકનીકી વિસ્તારમાં ખુબજ લોકપ્રિય છે. બીજું કારણ એ છે કે તે એક ખૂબ જ વ્યવસ્થિત ઑપનસૉર્સ છે અને જ્ઞાનવહેંચણીની શૈક્ષણિક ભાવના અને સહયોગથી વિકસાવવામાં આવ્યું છે, તેથી તે શિક્ષણશાસ્ત્રીઓ અને વિદ્વાનોમાં પણ લોકપ્રિય છે. આ ક્ષેત્રના લોકો લેટેક્સનો ઉપયોગ કરે છે, પોતાનાં મંતવ્યો અને અનુભવો વહેંચે છે, એકબીજાને મદદ કરે છે અને લેટેક્સના વધુ વિકાસ માટે નવાં પૅકેજ બનાવે છે અને વહન કરે છે.

લેટેક્સમાં આપમેળે અનુક્રમનંબર આપવાની અને સંદર્ભ આપવાની તેમજ આપમેળે વિષયવસ્તુનું કોષ્ટક, અનુક્રમણિકા અને બીજી આવી જરૂરિયાત પૂરી પાડવાની સગવડ છે, જે લેખકનો માનસિક બોજો હળવો કરે છે.

સારાંશ

આ પ્રકરણમાં આપણે લેટેક્સનો ઉપયોગ કરીને તક્નિકી અને ગાણિતિક દસ્તાવેજ કઈ રીતે ગોઠવાય તે શીખ્યા. લેટેક્સનાં આવશ્યક તત્ત્વો ભણ્યાં. આપણે અહીં લેટેક્સ દસ્તાવેજના માળખા અને વાક્યરચનાની ચર્ચા કરી. ત્યાર પછી લેટેક્સમાં મૂળભૂત દસ્તાવેજ કઈ રીતે બનાવાય તેમજ કઈ રીતે ગોઠવાય, તેની ચર્ચા કરી અને છેલ્લે લેટેક્સની મુખ્ય તાકાત - ગાણિતિક સામગ્રી કઈ રીતે ગોઠવાય તે જોઈ ગયા. તે શબ્દપ્રક્રિયકથી કઈ રીતે જુદું પડે છે, તેની પણ ચર્ચા કરી અને અંતમાં લેટેક્સના ફાયદાઓની ચર્ચા કરી.

SciTE અને લેટેક્સ કન્ફિગ્યુર કરવું (ફક્ત શિક્ષકો માટે)

- જો SciTE પહેલેથી ખૂલેલું હોય તો બંધ કરો.
- ટર્મિનલ ઑપન કરો.
- નીચેનો કમાન્ડ રન કરો :

  sudo gedit /usr/share/scite/tex.properties&

  આ કમાન્ડ આપવાથી gedit એડિટરમાં ફાઇલ ખૂલશે. તેમાં નીચે પ્રમાણે ફેરફાર કરો :
- નીચેની લીટીઓ કાઢી નાખો :

  file.patterns.tex=*.tex;*.sty

  file.patterns.context=*.tex;*.tui;*.tuo;*.sty

  ખાતરી કરો કે તમે નીચે આપેલી લીટી કાઢી નથી :

  file.patterns.latex=*.tex;*.sty;*.aux;*.toc;*.idx
- લીટી બદલો :

  command.go.$(file.patterns.latex)=gv $(FileName).pdf

  ના સ્થાને

  command.go.$(file.patterns.latex)=evince $(FileName).pdf લીટી લખો
- ફાઇલને સેવ કરી gedit બંધ કરો.
- SciTE શરૂ કરો. યોગ્ય લેટેક્સ ફાઇલ ખોલી F7 અને F5 દબાવો. લેટેક્સ ફાઇલ કમ્પાઇલ થઈને એક પીડીએફ ફાઇલ ખૂલશે. SciTEમાં પરત ફરતા પૂર્વે પીડીએફ ફાઇલ અવશ્ય બંધ કરો.

સ્વાધ્યાય

1. શબ્દપ્રક્રિયક સાથે લેટેક્સની સરખામણી કરી બંનેનાં સારાં અને નબળાં પાસાંની યાદી તૈયાર કરો.
2. લેટેક્સની લોકપ્રિયતાનાં મુખ્ય કારણોની યાદી તૈયાર કરો.
3. લેટેક્સમાં સીધાં ઉપયોગ ન કરી શકાય તેવા અક્ષરો અને અનામત અક્ષરોની યાદી તૈયાર કરો.
4. લેટેક્સ-દસ્તાવેજનું માળખું સમજાવો.
5. \frac કમાન્ડ અને નેસ્ટેડ \frac કમાન્ડ ઉદાહરણ સહિત સમજાવો.
6. **નીચે આપેલા વિકલ્પોમાંથી યોગ્ય વિકલ્પ પસંદ કરો :**

   (1) આધુનિક શબ્દપ્રક્રિયા સૉફ્ટવેર નીચેનામાંથી કઈ પદ્ધતિ પ્રમાણે ચાલે છે ?

   (a) WIGIWIS (b) WISYWIG (c) WYSIWYG (d) WISYWYG

   (2) નીચેનામાંથી ક્યો અનામત અક્ષર લેટેક્સમાં નથી ?

   (a) @ (b) % (c) $ (d) ^

(3) નીચેનામાંથી ક્યું \begin{name}થી શરૂ થાય છે અને \end{name}થી પૂરું થાય છે ?

(a) જૂથ (b) મુદ્દો (c) એન્વાયર્નમેન્ટ (d) પ્રસ્તાવના

(4) નીચેનામાંથી ક્યો અક્ષર લેટેક્સમાં કૉમેન્ટ (ટિપ્પણી) માટે વપરાય છે ?

(a) $ (b) % (c) # (d) &

(5) નીચેનામાંથી ક્યો ભાગ લેટેક્સમાં મેટાડેટા ધરાવે છે ?

(a) પ્રાસ્તાવિક ભાગ (b) TOC (c) પ્રસ્તાવના (d) એન્વાયર્નમેન્ટ

(6) નીચેનામાંથી કઈ વેબસાઇટ લેટેક્સનાં પૅકેજ ધરાવે છે ?

(a) CTAN (b) CLAN (c) CTEN (d) CLEN

(7) નીચેનામાંથી ક્યું આપમેળે અનુક્રમનંબર આપતું નથી ?

(a) \section (b) \subsection (c) \chapter* (d) \part

(8) નીચેનામાંથી ક્યું એન્વાયર્નમેન્ટ લખાણની લીટીમાં ગાણિતિક વિષયવસ્તુ દર્શાવે છે ?

(a) displaymath (b) math (c) equation (d) text

(9) નીચેનામાંથી ક્યો કમાન્ડ સમૂહ-યુનિયનનું ચિહ્ન બનાવે છે ?

(a) \cup (b) \setunion (c) \cap (d) \union

(10) અનુક્રમણિકા અથવા સબસ્ક્રિપ્ટ દર્શાવવા માટે નીચેનામાંથી ક્યો પ્રક્રિયક વપરાય છે ?

(a) _ (અન્ડરસ્કોર) (b) ^ (કૅરેટ) (c) – (બાદબાકી) (d) < (ની કરતાં ઓછું)

(11) નીચેનામાંથી ક્યું પ્રસ્તાવનામાં પ્રથમ લીટી તરીકે વપરાય છે ?

(a) \usepackage (b) \title (c) \maketitle (d) \documentclass

પ્રાયોગિક સ્વાધ્યાય

1. લેટેક્સમાં આ પુસ્તકના માળખાની રચના કરો. તેમાં પ્રસ્તાવના, વિષયવસ્તુનું કોષ્ટક, પ્રકરણ, મુદ્દાઓ, પેટા મુદ્દાઓ વગેરેનો સમાવેશ કરો. દરેક વિભાગમાં થોડાક શબ્દો લખો.

2. લખાણ-ગોઠવણ (ફૉર્મેટિંગ) દર્શાવતો લેટેક્સ દસ્તાવેજ બનાવો.

3. ફકરાની ગોઠવણ (ફૉર્મેટિંગ) દર્શાવતો એક લેટેક્સ-દસ્તાવેજ તૈયાર કરો.

4. એક લેટેક્સ-દસ્તાવેજ બનાવી પૃષ્ઠ લેઆઉટ સાથે તેનો પ્રયોગ કરો.

5. નીચે આપેલ વિષયવસ્તુનો એક લેટેક્સ-દસ્તાવેજ તૈયાર કરો :

(a) If $A = \{x \mid x \in N, x \leq 4\}$, $B = \{-1, 0, 1, 2, 3\}$ and $C = \{0, 1, 2\}$, then $(A \cup B) \cap (A \cup C) = \{0, 1, 2, 3, 4\}$.

(b) If $A = \{x \mid x \in N, x \leq 7\}$ and $B = \{2, 4, 6\}$ then $B \subset A$.

(c) $\dfrac{(81)^{\frac{1}{4}}}{(625)^{\frac{1}{4}}} + \dfrac{(216)^{\frac{1}{3}}}{(8)^{\frac{1}{3}}} - (729)^{\frac{1}{6}}$

(d) $X \subset Y$ and $Y \subset X \Rightarrow X = Y$

(e) $\overrightarrow{PQ} \cap \overrightarrow{PR} = P$

(f) $BE \cap EG = \{E\}$

(g) $\sqrt{s(s-a)(s-b)(s-c)}$

(h) $Area of \odot(O,r) = \pi r^2$

(i) $\sqrt{a+2\sqrt{b}} = \sqrt{x} + \sqrt{y}$

(j) $x = \dfrac{a+\sqrt{a^2-4b}}{2}, y = \dfrac{a-\sqrt{a^2-4b}}{2}$

(k) $\dfrac{h(\tan\alpha - \tan\beta)}{\tan\alpha\tan\beta}$

અન્ય ઉપયોગી નિઃશુલ્ક ટૂલ્સ અને સેવાઓ 13

આ પ્રકરણમાં આપણે કેટલાંક નિઃશુલ્ક ટૂલ્સ અને સેવાઓની ચર્ચા કરીશું. અહીં આપણે ઝડપથી માહિતી-સંકોચન તક્નિક્નો પરિચય મેળવીશું. આ તક્નિક વડે આપણે માહિતી સંગ્રહ કરવા માટે સ્મૃતિસંચય (Memory) અને તક્તિ (Disk) ઉપરની જગ્યામાં ઘટાડો કરી શકીએ છીએ. અહીં આપણે ફાઇલ અને ડિરેક્ટરી ઉપર આ તક્નિક્નો ઉપયોગ કરવા માટેના આર્ચિવ મૅનેજર ટૂલ વિશે ચર્ચા કરીશું. એક પ્રતિષ્ઠિત મલ્ટિમીડિયા ટૂલ, VLC મીડિયા પ્લેયરની ટૂંકમાં ચર્ચા કરીશું. અહીં આપણે ગુગલમૅપની સેવાની પણ ચર્ચા કરીશું, જે આપણને જુદી જુદી જગ્યાના નકશાઓ પૂરી પાડતી ઑનલાઇન સેવા છે. આપણે બે નાના પણ ઉપયોગી વિનિયોગ (Application) જોઈશું. કૅરેક્ટરમૅપ, જે લખાણમાં આપણને જુદી-જુદી ભાષાનાં ચિહ્નો અને અક્ષરો ઉમેરવા આપે છે. ત્યાર પછી આપણે આંકડાકીય ગણતરી માટેનું 'આર' (R) એન્વાયરમેન્ટનું સામાન્ય નિરીક્ષણ કરીશું, જેમાં આપણે સામાન્ય ગાણિતિક પ્રક્રિયાઓ જેવી કે મધ્યક અને મધ્યસ્થ શોધવા વિશે ભણીશું. તેમજ Rમાં બારચાર્ટ અને હિસ્ટોગ્રામ દોરતા શીખીશું. છેલ્લે આપણે બે ટૂલ્સ, રેશનલ પ્લાન અને સ્કાઇપ જોઈશું. આમાંના કેટલાક વિનિયોગ (Application) ઉબન્ટુ 10.04 LTSમાં પહેલેથી જ પ્રસ્થાપિત કરેલ હોય છે અને કેટલાક ઉબુન્ટુના ભંડાર (Repositories)માંથી પ્રસ્થાપિત કરવાની જરૂર પડે છે.

માહિતી-સંકોચન (Data Compression)

અદ્યતન કમ્પ્યૂટર સિસ્ટમમાં લાખો ફાઇલો હોય છે. ઘણી વખત એક કમ્પ્યૂટરમાંથી બીજા કમ્પ્યૂટર અથવા સંગ્રહ ઉપકરણમાં ઘણી બધી ફાઇલો અથવા આખી ડિરેક્ટરી પરિવહન કરવાની જરૂર પડે છે. જ્યારે આપણને આવી કમ્પ્યૂટરફાઇલ (કે જેમાં માહિતી, પ્રોગ્રામ અથવા બંને હોય) એક કમ્પ્યૂટરમાંથી બીજા કમ્પ્યૂટર અથવા સંગ્રહ-ઉપકરણમાં સ્થાનાંતરિત કરવાની જરૂર પડે, ત્યારે સ્થાનાંતરિત અથવા સંગ્રહ થતો માહિતીનો જથ્થો ચિંતાનો વિષય બની શકે છે. કમ્પ્યૂટર નેટવર્ક અને બાહ્ય સંગ્રહ ઉપકરણો બંને સામાન્ય રીતે કમ્પ્યૂટરના આંતરિક ઘટકો જેટલા ઝડપી નથી. આમ, મોટા જથ્થામાં માહિતીનું પરિવહન વધુ સમય લે છે. જો પરિવહન માટે ઇન્ટરનેટનો ઉપયોગ કરવામાં આવે, તો ઇન્ટરનેટની ધીમી ગતિના કારણે એ પરિવહન માટે ઘણો વધુ સમય લે છે. આ પરિવહન ઇન્ટરનેટના જોડાણ ઉપર પણ ભાર મૂકે છે. જો કોઈ વપરાશકર્તા અથવા સંસ્થા અમર્યાદિત ઇન્ટરનેટ યોજના ન ધરાવતા હોય, તો મોટા જથ્થાની માહિતીનું પરિવહન ઊંચી કિંમતમાં પરિણમે છે.

એવી જ રીતે ફાઇલ અને ડિરેક્ટરીઓનું સંગ્રહ-ઉપકરણમાં પરિવહન કરવામાં આવે, તો સંગ્રહ-ઉપકરણની મર્યાદિત ક્ષમતા અને વિવિધ ઉપયોગના કારણે પરિવહન કરવાનો માહિતીજથ્થો ફરીથી એક સમસ્યા બની જાય છે. આ સમસ્યાને ધ્યાનમાં લઈને શક્ય હોય ત્યાં કમ્પ્યૂટર ફાઇલ અને આખી ડિરેક્ટરી દ્વારા રોકવામાં આવતી જગ્યાને ઘટાડવાની જરૂર પડે છે. અનુકૂળતાની દૃષ્ટિએ, ઘણા બધા કિસ્સામાં ફાઇલના વિશાળ જથ્થા અથવા જટિલ ડિરેક્ટરી બંધારણ કરતાં એક અલગ ફાઇલને નિયંત્રિત કરવી ઇચ્છનીય છે.

કમ્પ્યૂટર વૈજ્ઞાનિકોએ આખી ડિરેક્ટરીને એક ફાઇલમાં મૂકવા માટેની તક્નિક વિકસાવી છે. આવી ફાઇલને 'આર્ચિવ' ફાઇલ તરીકે ઓળખવામાં આવે છે. તેઓએ કમ્પ્યૂટર ફાઇલ અને ડિરેક્ટરી માટેની સંગ્રહસ્થાનની જરૂરિયાતો ઘટાડવા માટે ઘણી બધી તક્નિકો વિકસાવી છે. આ તક્નિકોને માહિતી-સંકોચન કહેવામાં આવે છે.

સામાન્ય રીતે માહિતી-સંકોચન, માહિતીના સંકેતલેખન દ્વારા માહિતીના પુનરાવર્તનને ઓળખવા અને આવા પુનરાવર્તનને ઘટાડવા અથવા દૂર કરવાનું કામ કરે છે. આવી કેટલીક તક્નિકો, જગ્યાસંરક્ષણ માટે ઓછી મહત્ત્વની માહિતીને ઓળખે છે અને તેને દૂર કરે છે. ઉદાહરણ તરીકે આકૃતિ 13.1માં બતાવ્યા પ્રમાણે બાળકોની કવિતા ધ્યાને લો. તેમાં ચોક્કસપણે

શબ્દોનાં ઘણાં પુનરાવર્તન છે. એમાં આ પ્રક્રિયાનો લાભ લેવા માટે દરેક શબ્દને એક અંક અથવા અક્ષર દ્વારા રજૂ કરી શકાય. સાંકેતિક લિપિમાં લખાયેલી ફાઇલની શરૂઆતમાં ટેબલની અંદર દરેક અંક / અક્ષર ક્યા શબ્દને રજૂ કરે છે, તે માહિતી દર્શાવવામાં આવે છે. (આ માહિતી ફાઇલને તેના મૂળ સ્વરૂપમાં ફેરવવા માટે જરૂરી છે.) ત્યાર પછી એક શબ્દ માટે એક અંક / અક્ષરનો ઉપયોગ કરવામાં આવે છે. આ રીતે, લાંબા શબ્દો ફક્ત એક વખત અને ત્યાર પછી તે શબ્દોને એક અક્ષર વડે દર્શાવવામાં આવે છે અને ચિહ્નો જેમ છે તેમજ છોડી દેવામાં આવે છે. અહીં ટેબલની શરૂઆતમાં ^ (કૅરેટ) ચિહ્ન અને અંતમાં $ ચિહ્નનું નિશાન કરવામાં આવેલ છે. આકૃતિ 13.1ના બીજા ભાગમાં સાંકેતિક લિપિમાં લખેલ ફાઇલ પણ દર્શાવવામાં આવેલ છે.

One little, two little, three little Indians
Four little, five little, six little Indians
Seven little, eight little, nine little Indians
Ten little Indian boys.

Ten little, nine little, eight little Indians
Seven little, six little, five little Indians
Four little, three little, two little Indians
One little Indian boy.

Actual Contents

^0:One 1:little 2:two 3:three 4:Indians 5:Four 6:five 7:six 8:Seven 9:eight A:nine B:Ten C:Indian D:boys E:boy$0 1, 2 1, 3 1 4
5 1, 6 1, 7 1 4
8 1, 9 1, A 1 4
B 1 C D.

B 1, A 1, 9 1 4
8 1, 7 1, 6 1 4
5 1, 3 1, 2 1 4
0 1 C E.

Encoded Contents

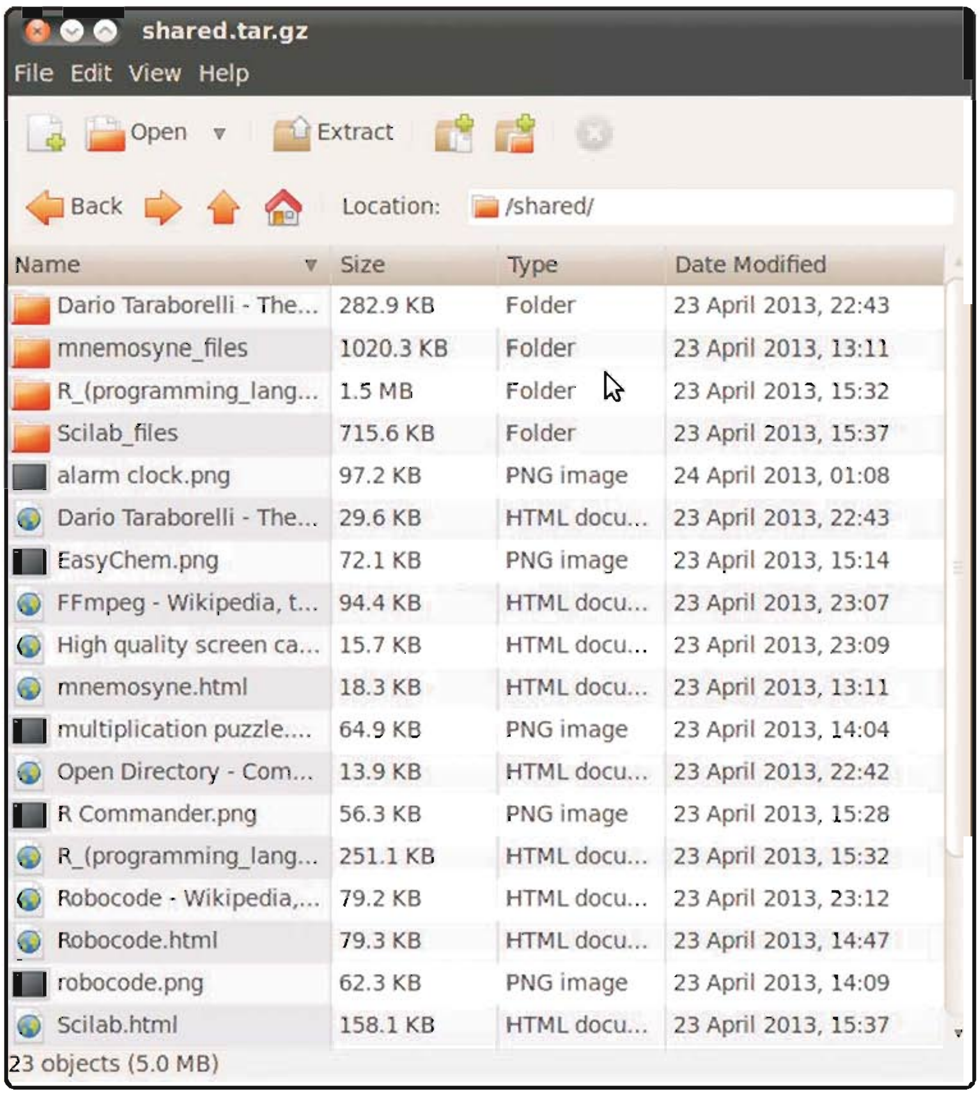
આકૃતિ 13.1 : માહિતી-સંકોચનનું ઉદાહરણ

મૂળ ફાઇલનું કદ 324 બાઇટ હતું અને સાંકેતિક લિપિમાં ફેરવેલ ફાઇલનું કદ ફક્ત 226 બાઇટ થયું. આમ થવાનું કારણ દરેક શબ્દ પૂરેપૂરો એક જ વખત વાપરવામાં આવ્યો અને ત્યાર પછી તે શબ્દને ફક્ત એક અક્ષર વડે દર્શાવવામાં આવ્યો. ઊલટી પ્રક્રિયા વડે સાંકેતિક લિપિમાં લખેલ ફાઇલને ગમે ત્યારે તેના મૂળ રૂપમાં ફેરવી શકાય છે.

અહીં આપેલી પદ્ધતિ એકદમ સરળ છે. મૂળ માહિતી-સંકોચનનો પ્રોગ્રામ વ્યવહારુ-અલ્ગોરિધમનો ઉપયોગ કરે છે, જે ફાઇલના કદમાં મોટા પ્રમાણમાં ઘટાડો કરે છે અને તે માહિતીના સિદ્ધાંત ઉપર આધારિત છે. સમૃદ્ધ લિનક્સ આર્કિવના સંચાલન માટે તૈયાર નિ:શુલ્ક અને ઑપનસોર્સ સૉફ્ટવેર પૂરાં પાડે છે, જેને આર્કિવ મેનેજર તરીકે ઓળખવામાં આવે છે.

આર્કાઇવ મૅનેજર (Archive Manager)

આકૃતિ 13.2માં દર્શાવ્યા પ્રમાણે આર્કાઇવ મૅનેજર એક કરતાં વધુ ફાઇલ અથવા આખી ડિરેક્ટરીને એક ફાઇલમાં જોડી દે છે, જે આર્કાઇવ (ફાઇલોનો ભંડાર) તરીકે ઓળખાય છે. લિનક્સમાં TAR (ટેપ આર્કાઇવર) એ સર્વસામાન્ય આર્કાઇવ-માળખું છે. તે આર્કાઇવને સંકોચનની સગવડ પણ પૂરી પાડે છે, જે તેની ફાઇલનું કદ ઘટાડે છે. તે એક કરતાં વધુ સંકોચન અલ્ગોરિધમ અને સંકોચિત ફાઇલમાળખાને સમર્થન આપે છે. ઝીપ (zip) ફાઇલમાળખું અને tar.gz ફાઇલમાળખું એ સર્વસામાન્ય સંકુચિત ફાઇલમાળખું છે. જ્યારે ઝિપ ફાઇલમાળખું એક કરતાં વધુ ફાઇલને એક ઝીપ ફાઇલમાં સંગ્રહ કરે છે. યુનિક્સ / લિનક્સ પદ્ધતિમાં સામાન્ય રીતે એક કરતાં વધુ ફાઇલોને એક ટાર (tar) માળખાની સંકોચનરહિત ફાઇલમાં ભેગી કરવામાં આવે છે અને ત્યાર બાદ gzip પ્રોગ્રામ (GNU ઝીપ, એક ઓપનસૉર્સ આર્કાઇવ પ્રોગ્રામ છે.)નો ઉપયોગ કરીને ઝીપમાળખાની ફાઇલમાં સંકોચન કરવામાં આવે છે. આવી ફાઇલને સામાન્ય રીતે tar.gz અનુલંબન હોય છે અને તેને ટારબૉલ (Tar Ball) તરીકે ઓળખવામાં આવે છે. લિનક્સમાં ફાઇલોના જથ્થાને અથવા સૉફ્ટવેરને વિતરણ કરવા માટે 'ટારબૉલ' એ ખૂબ જ સામાન્ય માળખું છે. ઝીપમાળખું પણ જુદાં-જુદાં નામ હેઠળ જુદા-જુદા વિનિયોગ માટે ઉપયોગમાં આવે છે. ઉદાહરણ તરીકે જાવા (JAVA)માં JAR ફાઇલ અને ઑફિસમાં OpenOffice.org ફાઇલમાળખું ઝીપસંકોચનનો ઉપયોગ કરે છે. નવી આર્કાઇવ બનાવવા માટે અને આર્કાઇવ ખોલવા તેમજ ફાઇલમાળખાં પસંદ કરવા માટે આર્કાઇવ મૅનેજર ફાઇલ નામનો અનુલંબન ભાગ ફાઇલમાળખાને ઓળખવા માટે ઉપયોગમાં લે છે.



આકૃતિ 13.2 : આર્કાઇવ મૅનેજર

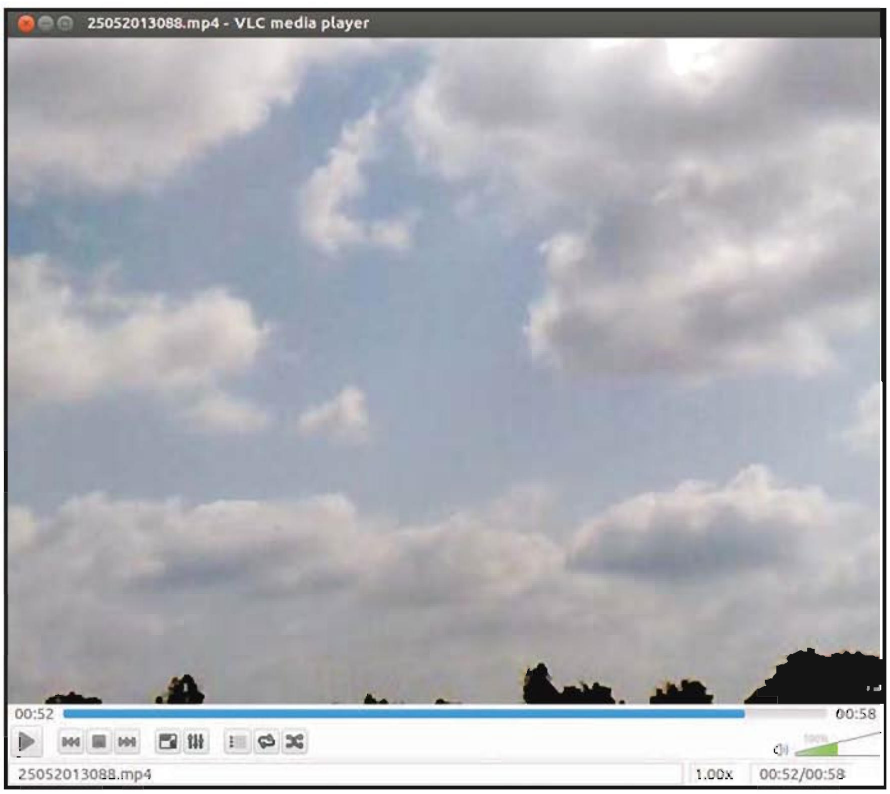
આર્કિવ ફાઈલની વિષયવસ્તુના અન્વેષણ માટે, આર્કિવમાંથી ફાઈલોને બહાર કાઢવા માટે, આર્કિવમાં નવી ફાઈલોને ઉમેરવા માટે, આર્કિવમાંથી ફાઈલોને દૂર કરવા માટે અને બીજાં કેટલાંક કાર્ય કરવા માટે પણ આર્કિવ મૅનેજરનો ઉપયોગ થાય છે. સામાન્ય રીતે આર્કિવ મૅનેજર શરૂ કરવા માટે કોઈ મેનુવિકલ્પ નથી. ફાઈલબ્રાઉઝરમાં આર્કિવ ફાઈલ ઉપર ડબલ ક્લિક કરવાથી આર્કિવ મૅનેજર શરૂ થાય છે. ફાઈલ અથવા ડિરેક્ટરી ઉપર જમણી (રાઈટ) ક્લિક કરી મેનુમાંથી Compress વિકલ્પ પસંદ કરીને નવી આર્કિવ ફાઈલ બનાવી શકાય છે. જે આ ફાઈલ અથવા ડિરેક્ટરીના વિષયવસ્તુની આર્કિવ બનાવે છે. આર્કિવ ફાઈલ માટે આ વિકલ્પ ઉપલબ્ધ હોતો નથી. જો આપણે ઇચ્છીએ તો Application મેનુના પેટામેનુ Accessoriesમાં આર્કિવ મૅનેજરનો ઉમેરો કરી શકીએ. આવું કરવા માટે ઉપરના ડાબા ખૂણાના ઉબુન્ટુ આઇકન ઉપર જમણી (રાઈટ) ક્લિક કરો. લિસ્ટમાંથી Edit મેનુ પસંદ કરો અને ડાયલોગબૉક્સમાં એસેસરીઝ ઉપર ક્લિક કરો. આર્કિવ મૅનેજર માટેના ચેકબૉક્સ પસંદ કરીને ડાયલોગબૉક્સ બંધ કરો.

જ્યારે આર્કિવ મૅનેજરમાં આર્કિવ ખૂલેલી હોય ત્યારે આકૃતિ 13.2માં દર્શાવ્યા મુજબ ફાઈલબ્રાઉઝરમાં ખોલેલી ફાઈલ સમાન દેખાય છે. તેમનાં કાર્ય અને કી-બોર્ડના ટૂંકા કમાન્ડ (Shortcuts) પણ સરખા જ છે. ડિરેક્ટરી ઉપર ડબલક્લિક કરવાથી તેજ ડિરેક્ટરી ખૂલે છે અને તેમાં રહેલી વિષયવસ્તુ દૃશ્યમાન થાય છે. ટૂલબારમાં રહેલ Up અને Back બટન દ્વારા પૅરેન્ટ ડિરેક્ટરી (જો આર્કિવમાં ઉપલબ્ધ હોય તો)માં અને પહેલાંની ડિરેક્ટરીમાં (જો હોય તો) જઈ શકાય છે. આમાંનું જ્યારે લાગુ પડે, ત્યારે જ શક્ય બને છે.

આ રીતે, આર્કિવ મૅનેજર આપણને આર્કિવ ફાઈલનો ઉપયોગ કરીને બૅકઅપ લેવા, બાહ્યસંગ્રહ ઉપકરણ વડે સ્થાનાંતરિત કરવા નેટવર્ક વડે સ્થાનાંતરિત કરવા, તક્તી (Disk) ઉપરની જગ્યા બચાવવા વગેરે માટે ઉપયોગી છે.

VLC મીડિયાપ્લેયર (The VLC Media Player)

આપણે આગળ ચર્ચા કર્યા મુજબ ઉબન્ટુમાં મલ્ટિમીડિયા ઑડિયો, વીડિયો વગેરે (શ્રાવ્ય અને દૃશ્ય) વગાડવા માટેનું ટૂલ આપે છે. તેમ છતાં બીજું મહત્ત્વનું ઓપનસૉર્સ ટૂલ VLC (મૂળ સ્વરૂપે તે VideoLAN Clientનું ટૂંકાક્ષરી) મીડિયાપ્લેયર આકૃતિ 13.3માં દર્શાવ્યું છે. તે તેની લાક્ષણિક્તા અને સાર્વત્રિક્તાને કારણે ખૂબ જ પ્રખ્યાત છે.



આકૃતિ 13.3 : VLC મીડિયાપ્લેયર

પૅરિસમાં વિશ્વવિદ્યાલયના સંગીતપ્રિય વિદ્યાર્થીઓએ અભ્યાસલક્ષી પ્રોજેક્ટ (યોજના) માટે શરૂ કરેલ તે હાલમાં જાહેરજનતા માટે ઘણી બધી ઑપરેટિંગ સિસ્ટમમાં ઉપલબ્ધ છે અને કરોડો વખત એ ડાઉનલોડ થયેલ છે. મલ્ટિમીડિયાના વિષયવસ્તુની એક સમસ્યા એ છે કે હાર્ડવેર ઉપકરણ (audio/video streams) પરથી આવતી મલ્ટિમીડિયા માહિતીને કમ્પ્યૂટરની માહિતીમાં ફેરવવા અને કમ્પ્યૂટર ઉપરની માહિતીને હાર્ડવેર ઉપકરણ ઉપર વગાડવા, ફરીથી ઑડિયો/વીડિયોમાં ફેરવવાના ઘણા બધા માર્ગ ઉપલબ્ધ છે.

આ પરિવર્તન (સાંકેતીકરણ) અને ઊલટું પરિવર્તન (અસાંકેતીકરણ) એક સૉફ્ટવેર વડે કરવામાં આવે છે. જેને કોડેક (કોડર-ડીકોડર) તરીકે ઓળખવામાં આવે છે. દરેક મલ્ટિમીડિયા માહિતીના માળખાને પોતાનું કોડેક જરૂરી હોય છે. VLCનો ફાયદો એ છે કે તે દરેક પ્રખ્યાત કોડેક અને દરેક પ્રખ્યાત માળખાને સમર્થન આપે છે. તે મોટા ભાગનાં દરેક ઉપકરણો જેવાં કે વેબકૅમેરા, એચડી મૉનિટર, દરેક પ્રકારનાં સ્પીકર, માઇક્રોફોન, હેડફોન વગેરેને સમર્થન આપે છે. ઑડિયો અને વીડિયો વગાડવા માટે VLC ઘણા બધા વિકલ્પો આપે છે. તે મલ્ટિમીડિયા ફાઇલને એક માળખામાંથી બીજા માળખામાં પરિવર્તિત પણ કરી શકે છે. તે નેટવર્કમાં રહેલ બીજા કમ્પ્યૂટરમાંથી ઑડિયો અને વીડિયો મેળવી પણ શકે છે અને ઑડિયો અને વીડિયો મોકલી પણ શકે છે.

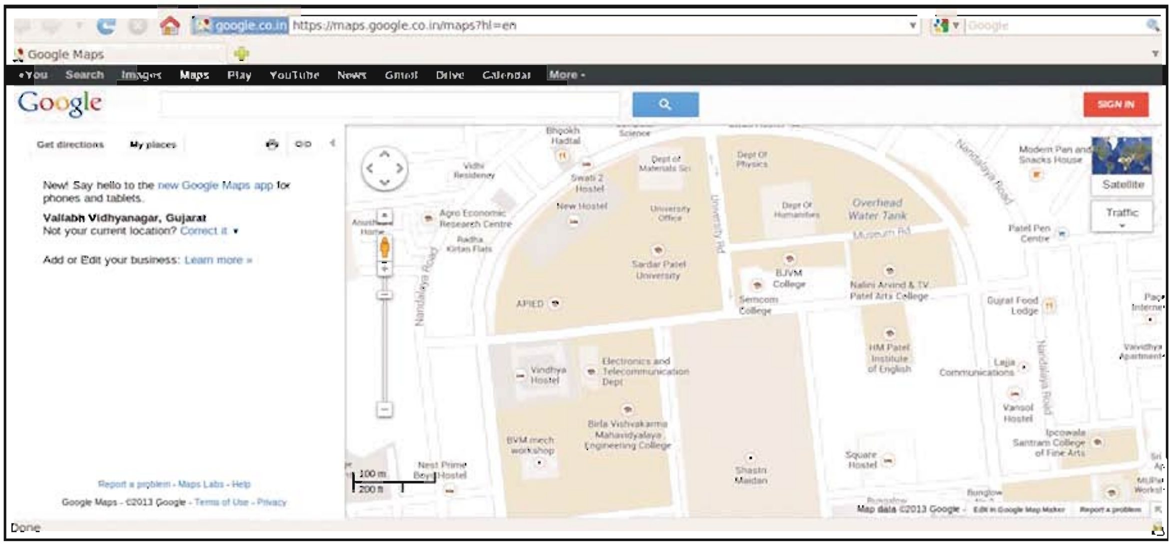
Applications → Sound & Video મેનુ વડે VLC શરૂ કર્યા પછી, **Media → Open File...** મેનુવિકલ્પ વડે આપણે એક અથવા વધારે ફાઇલને ખોલી શકીએ છીએ. **Media → Open Directory...** વિકલ્પ વડે આપણે એકસાથે આખી ડિરેક્ટરી ખોલી શકીએ છીએ. જો આપણે એક ડિરેક્ટરી ખોલીએ તો, હકીકતમાં VLC તે ડિરેક્ટરીમાં રહેલી વગાડી શકાય તેવી તમામ મીડિયા ફાઇલને ખોલે છે. જ્યારે એક અથવા એક કરતાં વધારે ફાઇલ ખૂલેલી હોય ત્યારે, તે ફાઇલ પ્લેલિસ્ટમાં ઉમેરાય છે. પ્લેલિસ્ટ એ મીડિયા ફાઇલ વગાડવાની યાદી છે. પ્લેલિસ્ટમાં રહેલ મીડિયાને આપણે અનુક્રમ પ્રમાણે અથવા આડા-અવળા ક્રમમાં વગાડી શકીએ છીએ. આપણે આગળ અથવા પાછળના માર્ગ (TRACK) પર જઈ શકીએ છીએ. બીજી વખત આજ ટ્રેક વગાડવા માટે આપણે પ્લેલિસ્ટને સેવ (SAVE) પણ કરી શકીએ છીએ. **View → Playlist** વિકલ્પની મદદ વડે આપણે પ્લેલિસ્ટને ખોલી શકીએ છીએ અને **Media → Save Playlist to File** વિકલ્પ વડે પ્લેલિસ્ટ સેવ કરી શકીએ છીએ. VLC પ્લેલિસ્ટના ઘણાં બધાં માળખાંને સમર્થન આપે છે પણ M3U માળખું એ સર્વસામાન્ય માળખું છે.

VLC મીડિયાપ્લેયર વિન્ડોના નીચેના ભાગમાં એક પ્રગતિ લીટી (પ્રોગ્રેસ બાર) દર્શાવવામાં આવે છે, જે ચાલુ ટ્રેકનો કુલ સમય અને ટ્રેક કેટલો વાગી ચૂક્યો છે તે સમય દર્શાવે છે. બાર ઉપર રહેલા નાના એવા સ્લાઇડરને સરકાવવામાં આવે, તો ચાલુ ટ્રેકમાં આપણે આગળ અને પાછળ હરીફરી શકીએ છીએ. તેની નીચે પ્લેબટન હોય છે. (જમણી તરફ દર્શાવેલું ત્રિકોણ જેવું નિશાન) જ્યારે ટ્રેક વાગતો હોય, ત્યારે આ બટન પોઝબટન (અટકવા માટેનું બટન) બની જાય છે. (બે સમાંતર ઊભી લીટી જેવું નિશાન). જ્યારે આપણે વગાડવું અટકાવી દઈએ, ત્યારે પાછું આ બટન પ્લેમાં ફેરવાય જાય છે. ત્યાર પછીનાં ત્રણ બટનમાંનું વચ્ચેનું બટન એ ટ્રેકને વગાડાતો સંપૂર્ણ બંધ કરવા માટેનું છે. જ્યારે ડાબી અને જમણી તરફના ડબલ તીરવાળાં બટન આપણને આગળ અને પાછળના ટ્રેક ઉપર લઈ જાય છે. ત્યાર પછીનું બટન વીડિયોને વિન્ડોમાં જોવા અથવા આખી સ્ક્રીનમાં જોવાના વિકલ્પમાં ફેરવવા માટેનું કાર્ય કરે છે. આનો મતલબ એ થયો કે જ્યારે આપણે બટન ઉપર ક્લિક કરીએ, ત્યારે તે આપણને એક સ્થિતિમાંથી બીજી સ્થિતિમાં ફેરવે છે. આખી સ્ક્રીનમાં વીડિયો જોતી વખતે ચાલુ કરવા, અટકાવવા કે બંધ કરવા માટેનાં નિયમન અદશ્ય હોય છે. વીડિયો ઉપર માઉસકર્સરને ફેરવવાથી કામચલાઉ રીતે અસ્થાયી વિન્ડોમાં તેને જોઈ શકાય છે. ત્યાર પછીનું બટન એ પ્લેલિસ્ટ દર્શાવવા માટેનું છે. આ બધા જ વિકલ્પ મેનુમાં પણ ઉપલબ્ધ છે.

VLCમાં બીજી ઘણી બધી કાર્યપ્રણાલી છે, અહીં આપણે મલ્ટિમીડિયા ફાઇલને એક માળખામાંથી બીજા માળખામાં પરિવર્તિત કરવા માટે VLC નો કઈ રીતે ઉપયોગ થાય, તેની ચર્ચા કરીશું. આ માટે મેનુમાંથી **Media → Convert / Save** વિકલ્પ પસંદ કરો. એડ (Add) બટનનો ઉપયોગ કરી, જે ફાઇલને પરિવર્તિત કરવાની છે, તેને પસંદ કરો અને ઉમેરો. **Convert / Save** બટનની બાજુની યાદીમાંથી પરિવર્તિત (Convert) વિકલ્પ પસંદ કરો. આથી એક અન્ય ડાયલોગબૉક્સ ખુલશે. અહીં, આપણે નિર્ધારિત ફાઇલનું નામ પૂરું પાડવું પડશે. (જેમાં પરિવર્તિત ટ્રેકને સેવ કરવાનો છે તે). આપણે, આપેલ ડ્રૉપડાઉનમાંથી ઇચ્છિત ફાઇલમાળખું (ઑડિયો અથવા વીડિયો) પસંદ કરવું જરૂરી રહેશે. સ્ટાર્ટ (શરૂ) બટન ઉપર ક્લિક કરવાથી પ્રક્રિયા શરૂ થશે અને પ્રગતિ દર્શાવવામાં આવશે. ક્યારેક આપણને પરિવર્તનની પ્રક્રિયામાં ભૂલનો સંદેશ પણ મળે છે.

ગુગલ નકશો (Google Map)

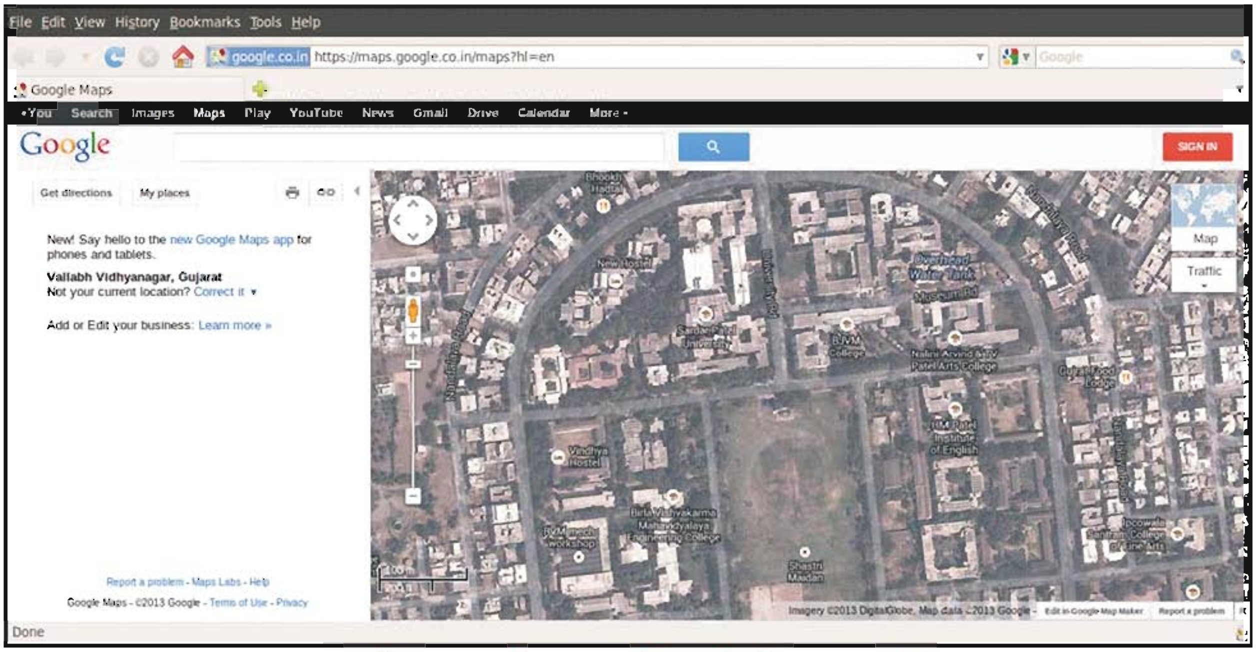
આકૃતિ 13.4 માં દર્શાવેલ ગુગલ નકશો એ ગુગલ ઈનકૉર્પોરેશનની ઈન્ટરનેટ આધારિત નિઃશુલ્ક સેવા છે. ગુગલે ઉપગ્રહની મદદ વડે આકૃતિ મેળવીને, કાર ઉપર કૅમેરા બાંધીને, બીજી સંસ્થાઓ પાસેથી માહિતી ખરીદીને અને દુનિયાના હજારો વ્યક્તિગત ઉપયોગકર્તાઓ પાસેથી માહિતી મેળવીને વર્ષોના સમયગાળા પછી આખી પૃથ્વીના વિશાળ નકશાની માહિતી ભેગી કરી છે. ગુગલ નકશો કોઈ પણ વ્યક્તિને નકશામાં ફેરફાર કરવા અને જમીનની નિશાની, ઈમારત વગેરે ઓળખવા માટે પરવાનગી આપે છે. તે કોઈ પણ જગ્યાનો ફોટોગ્રાફ અપલોડ કરવા અને કોઈ પણ જગ્યાની પરીક્ષણ-માહિતી આપવાની પરવાનગી પણ આપે છે. આ સેવાનો ઉપયોગ હરતાંફરતાં ઉપકરણો જેવા કે સ્માર્ટફોન અને ટેબ્લેટમાં ખૂબ જ થાય છે.



આકૃતિ 13.4 : ગુગલ નકશાની સેવા

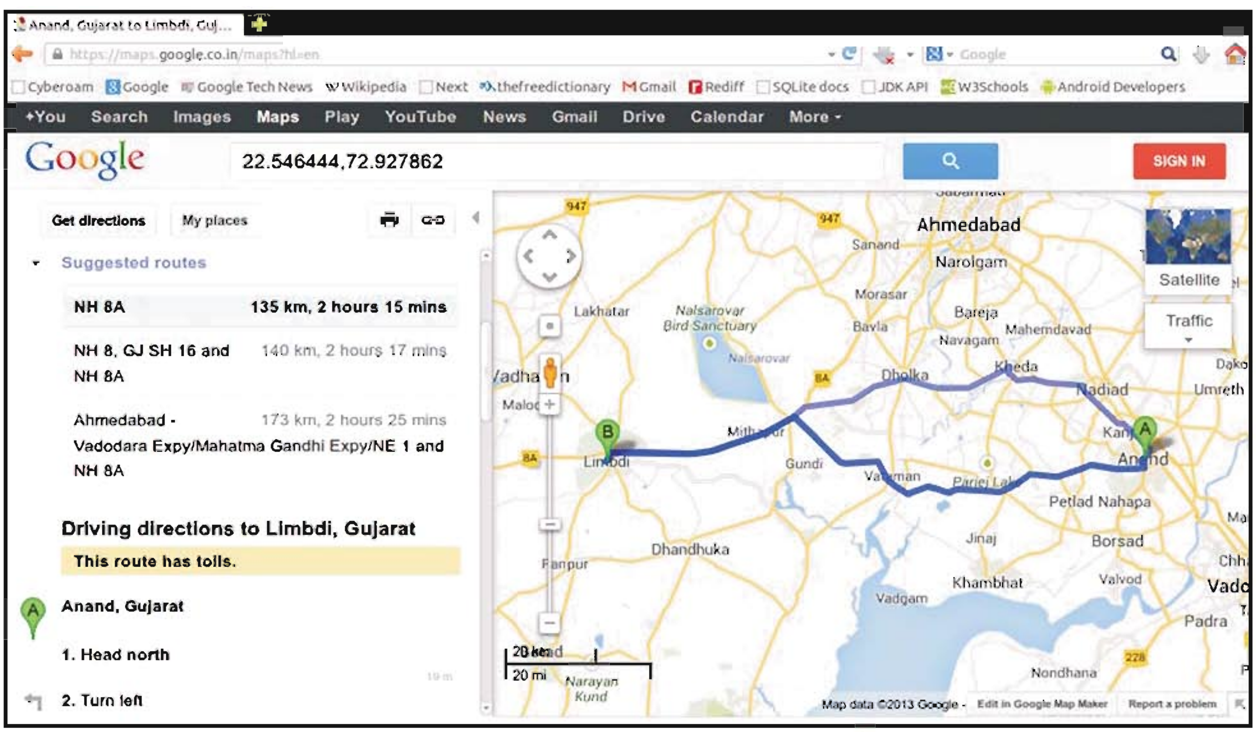
આ સેવા વેબબ્રાઉઝરમાં http://www.google.co.in વેબસાઇટ ખોલીને ઉપરના મેનુમાંથી મૅપ (નકશાઓ) વિકલ્પ પસંદ કરીને મેળવી શકાય છે અથવા સીધેસીધું http://maps.google.co.in. આપીને પણ મેળવી શકાય છે. મોબાઇલ ફોનમાં આ સેવા મોબાઇલ ફોનના વેબબ્રાઉઝર વડે તદ્ઉપરાંત ગુગલ નકશાના વિનિયોગ (Application) વડે મેળવી શકાય છે. જ્યારે આ સેવા આપણે ચાલુ કરીએ છીએ, ત્યારે તે પ્રથમ આપણા હાલના વિસ્તારને જાણવાનો પ્રયત્ન કરે છે. જ્યારે કમ્પ્યૂટર ઉપર તે ઉપયોગકર્તાના વિસ્તારનો ખ્યાલ તે ઉપયોગકર્તાના ઈન્ટરનેટના જોડાણ ઉપરથી મેળવે છે. મોબાઇલ ફોન ઉપર, GPS (Global Positioning System) સગવડનો ઉપયોગ હોય, તો ઉપયોગકર્તાના વિસ્તારનો ખ્યાલ ખૂબ જ ચોકસાઈથી મળે છે. આ પદ્ધતિ ઉપગ્રહોના તરંગોનો ઉપયોગ કરીને અંદાજિત થોડા મીટરની ભૂલથી ઉપયોગકર્તાનો વિસ્તાર શોધી કાઢે છે. ગુગલ નકશો જો તે અચોક્કસ હોય, તો વિસ્તારને ચોક્કસ કરવાની / બરાબર કરવાની પરવાનગી આપે છે અને પછી તે હાલના વિસ્તારનો નકશો દર્શાવે છે. આપણે નકશાને જુદી દિશામાં ઘસડીને અથવા ડાબી બાજુના વર્તુળમાં રહેલ તીરના નિશાન ઉપર ક્લિક કરીને આમતેમ ફેરવી શકીશું. ડાબી બાજુનું ઊભું સ્લાઇડરબાર આપણને નકશાને દૂર લઈ જવા અને નજીક લાવવાની પરવાનગી આપે છે.

ગુગલ નકશાઓ કોઈ એક જગ્યાની માહિતી જુદાં જુદાં સ્વરૂપે આપી શકે છે. આકૃતિ 13.4માં દર્શાવ્યા પ્રમાણે તેનો સામાન્ય દેખાવ એ નકશો (Map View) છે. સેટેલાઇટ (ઉપગ્રહ) બટન ઉપર ક્લિક કરવાથી આકૃતિ 13.5માં દર્શાવ્યા પ્રમાણે આપણા આ દેખાવને ઉપગ્રહ ચિત્રના દેખાવમાં ફેરવી શકીએ છીએ. આ દેખાવમાં આકાશના ઉપગ્રહ દ્વારા લેવાયેલ તસવીર દર્શાવવામાં આવી છે. આ તસવીર પરિચિત ઈમારતો અને રસ્તાઓ ઓળખવા માટે પર્યાપ્ત અને સ્પષ્ટ હોય છે. આ સેવા એક જગ્યા Aથી બીજી જગ્યા B સુધી પહોંચવા માટેનો દિશાનિર્દેશ પણ કરે છે. આ બે જગ્યાઓ બે જુદાં-જુદાં શહેર અથવા એક શહેરનાં જુદાં-જુદાં સ્થળ પણ હોઈ શકે. આ સેવા જ્યાં શક્ય હોય, ત્યાં રસ્તાઓની પસંદગી પણ પૂરી પાડે છે. આકૃતિ 13.6 ઉદાહરણ પૂરું પાડે છે. જો આપણે GPS રિસીવરવાળું મોબાઇલ ઉપકરણ વાપરીએ તો મુસાફરી દરમિયાન આ સેવા આપણને (એક વળાંકથી બીજા વળાંક) એક તરફથી બીજી તરફ જવાનું માર્ગદર્શન પણ પૂરું પાડે છે. રસ્તા ઉપર અથવા ગીચ શહેરી વિસ્તારમાં ચાલતાં અથવા વાહન ચલાવતા આ સેવા હાલની જગ્યાએથી નિશ્ચિત જગ્યા સુધી પહોંચવાનો રસ્તો નક્કી કરી આપે છે અને આપણને સ્ક્રીન ઉપર અથવા અવાજ વડે (બોલીને સૂચના આપીને) માર્ગદર્શન આપે છે.



આકૃતિ 13.5 : ઉપગ્રહ તસવીર દર્શાવતો ગુગલ નકશો

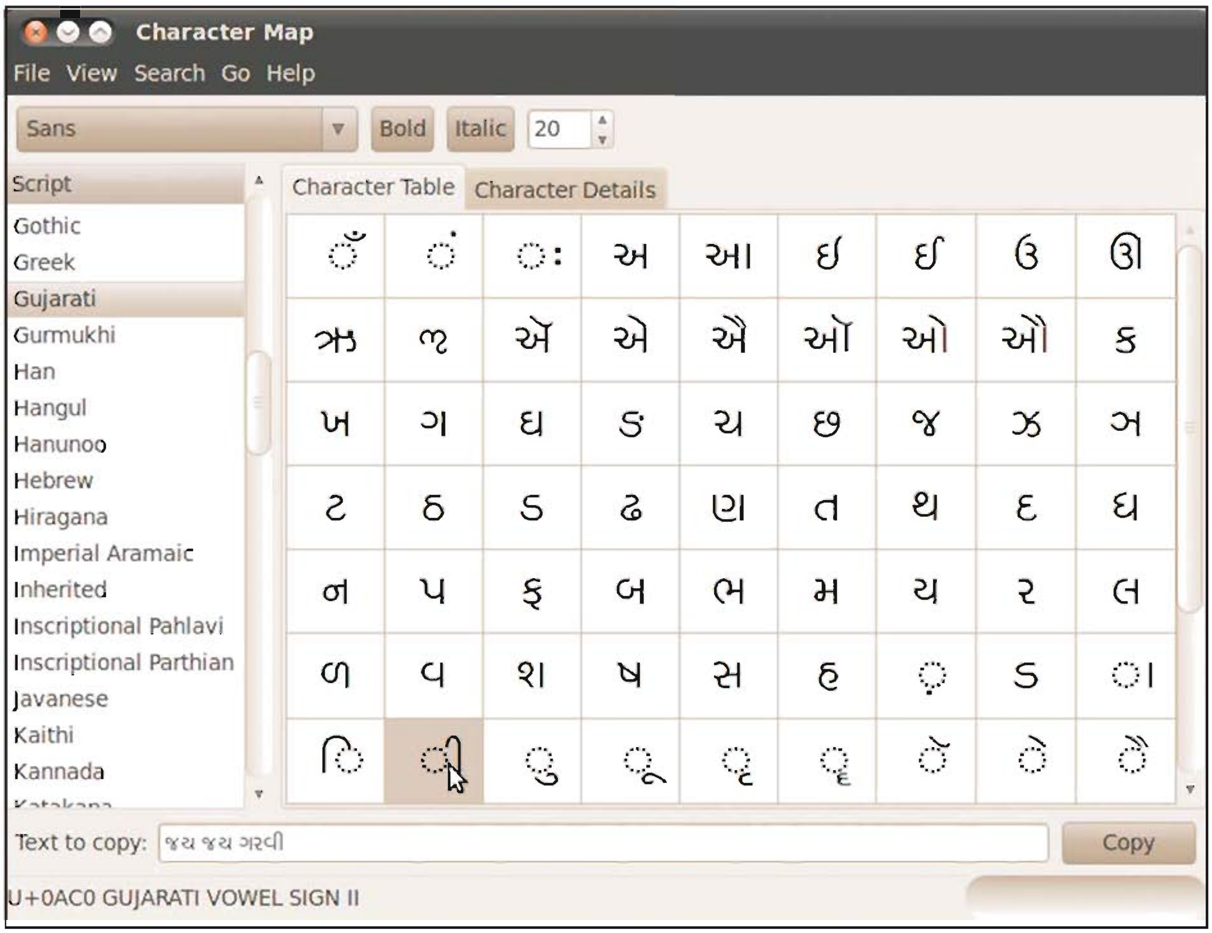
આ સેવાના જુદા-જુદા ઘણા ઉપયોગ છે. આ સેવાનો ઉપયોગ સ્થળની ચોક્કસ જગ્યા શોધવા માટે પણ થાય છે. આ સેવાનો ઉપયોગ કરી આપણે જે શહેરથી પરિચિત ન હોઈએ, તેવા શહેર અથવા અજાણી જગ્યા પર પહોંચવા માટેનો દિશાનિર્દેશ પણ મેળવી શકીએ છીએ. સરકારી સંસ્થાઓ અને વેપારીઓ, લોકોને પોતાની ઓફિસે કઈ રીતે પહોંચવું તેની માહિતી પણ આ સેવા વડે પૂરી પાડે છે. આ સેવાનો ઉપયોગ કરીને કોઈ પણ વ્યક્તિ કોઈ પણ જગ્યાનો નકશો વેબસાઇટમાં દર્શાવી શકે છે. ઘણી બધી સંસ્થાઓ આવા નકશાઓ પોતાની વેબસાઇટમાં દર્શાવે છે. પ્રવાસીની માહિતી દર્શાવતી વેબસાઇટ અને સરકારનો પ્રવાસન વિભાગ પણ આ સેવાનો ઉપયોગ કરે છે. બસનો માર્ગ દર્શાવવા, ચાલુ ટ્રેનનો હાલનો વિસ્તાર દર્શાવવા વગેરે કાર્યો માટે પણ આનો ઉપયોગ થાય છે. આ સેવા નજીકનું ATM, બેન્ક, ભોજનાલય, બસસ્ટૉપ અથવા જરૂરી કોઈ સ્થળ શોધવાની સગવડતા પણ આપે છે.



આકૃતિ 13.6 : ગુગલ નકશાનો ઉપયોગ કરીને દિશાનિર્દેશ શોધવો

અક્ષરનકશો (Character Map)

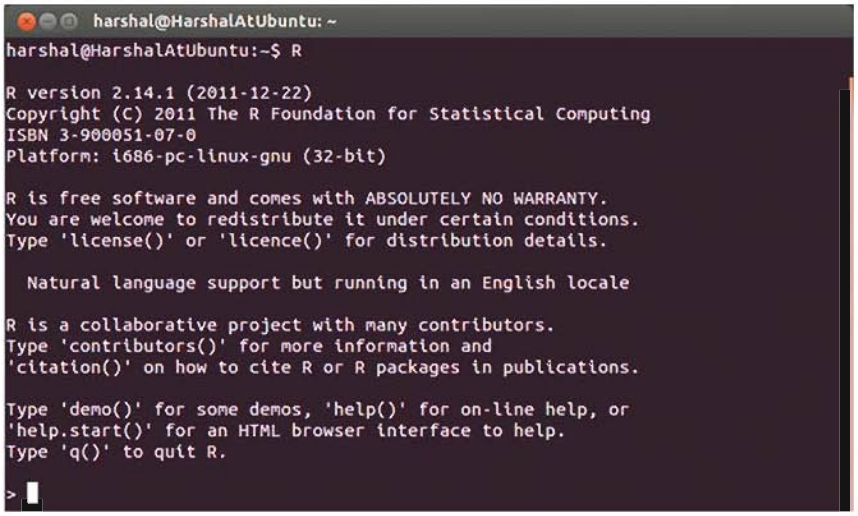
આકૃતિ 13.7માં દર્શાવેલ (અક્ષરનકશો) કૅરેક્ટરમૅપ પ્રોગ્રામનો ઉપયોગ કોઈ પણ વિનિયોગમાં યુનિકોડ (Unicode) અક્ષર દાખલ કરવા માટે થાય છે. પ્રથમ ડાબી તકતી (પૅન)માંથી લિપિ પસંદ કરવાની અને ત્યાર પછી જે અક્ષરને લખાણ અથવા નકલના વિસ્તારમાં ઉમેરવાનો હોય, તેના ઉપર ડબલ ક્લિક કરવાની. ક્લિક કર્યા પછી ખાલી જગ્યા અને ચિહ્નોને કી-બોર્ડ વડે સીધા જ દાખલ કરી શકાય છે. અક્ષરો ભેગા કરવા, જેમકે ગુજરાતીમાં 'દીર્ઘ ઈ' ઉમેરવા માટે અનુરૂપ યોગ્ય અચલને ભેગા કરવા પડે છે. નીચે સ્ટેટસ લાઇનમાં એક ક્લિક વડે પસંદ કરેલ અક્ષર અથવા ડબલ ક્લિક વડે દાખલ કરેલ અક્ષરની ટૂંકી માહિતી દર્શાવે છે. જ્યારે અક્ષરો વિશેની વધુ માહિતી Character Details નામના ટૅબમાં આપેલ છે. આવશ્યક જથ્થામાં વિષયવસ્તુ મેળવ્યા પછી, આપણે આ વિષયવસ્તુની નકલ (Copy) કરી અને કોઈ વિનિયોગમાં પેસ્ટ (Paste) કરી શકીએ છીએ. બીજી કોઈ લિપિમાં પ્રાસંગિક ધોરણે તમારે થોડા અક્ષરો જ ટાઈપ કરવા હોય, તો આ માટે કૅરેક્ટરમૅપ એ એક સારો ઉકેલ છે.



આકૃતિ 13.7 : કૅરેક્ટરમૅપ પ્રોગ્રામ

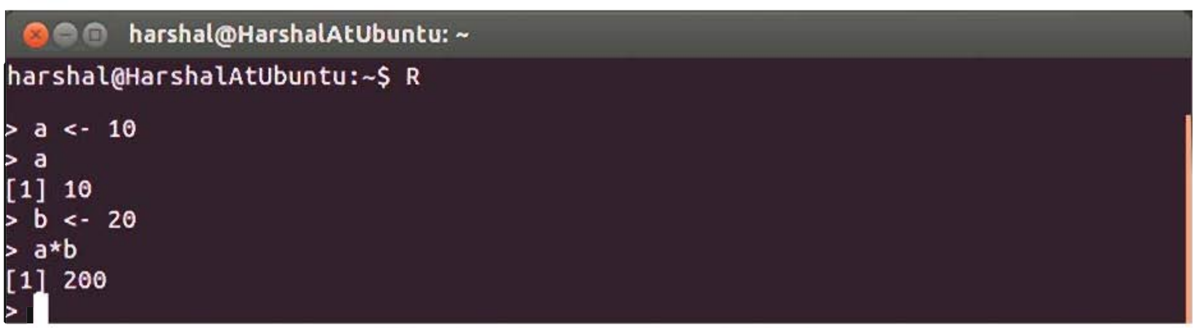
'આર' સૉફ્ટવેર (The 'R' Software)

'આર' એ એક આંકડાકીય ગણતરી માટેનું નિઃશુલ્ક સૉફ્ટવેર છે. તે એક GNU યોજના છે. તેનો આંકડાકીય વિશ્લેષણ માટે બહોળા પ્રમાણમાં ઉપયોગ થાય છે. તેને પોતાની સ્ક્રિપ્ટિંગ ભાષા છે. તે એક કેસ સંવેદનશીલ (કેસ સૅન્સિટિવ) ભાષા છે. 'આર'ને કમાન્ડલાઇન અને ગ્રાફિક્સ નામના બે કાર્યપ્રદેશ છે. જો આપણે GUI પ્રદેશનો ઉપયોગ કરવો હોય, તો આપણે ઉબુન્ટુ સૉફ્ટવેર કેન્દ્રમાંથી આર કમાન્ડર અથવા આર સ્ટુડિયો નામનું ગ્રાફિકલ એડિટર પ્રસ્થાપિત કરવું પડે. 'આર'ને પ્રસ્થાપિત કરવા માટેના કમાન્ડ આ પ્રકરણના અંતમાં આપેલ છે. ટર્મિનલ વિન્ડોમાંથી આર લિપિને બોલાવવા માટે, ટર્મિનલ ખોલીને કમાન્ડપ્રૉમ્પ્ટ ઉપર આર (R) લખો અને એન્ટર કી દબાવો. આકૃતિ 13.8માં દર્શાવ્યા પ્રમાણે તમને આર પ્રૉમ્પ્ટની સાથે સ્વાગતસંદેશ મળશે.



આકૃતિ 13.8 : આર કમાન્ડ પ્રૉમ્પ્ટ

લિનક્સ શેલની જેમ જ R કોમેન્ટની નિશાની તરીકે # નો ઉપયોગ કરે છે. # પછીની કોઈ પણ લખાણ લાઇનના અંત સુધી કોમેન્ટ (ટિપ્પણી) તરીકે ગણવામાં આવે છે. નંબર અને સ્ટ્રિંગ બે મૂળભૂત ડેટા ટાઇપ છે. (માહિતીના પ્રકાર છે.) સ્ટ્રિંગને એક અવતરણ અથવા ડબલ (બે) અવતરણ ચિહ્નોમાં બાંધવામાં આવે છે. બીજી પ્રોગ્રામિંગ ભાષાની જેમ જ સામાન્ય પ્રક્રિયકો અને વિધેયો પણ ઉપલબ્ધ છે. વસ્તુઓની ક્રમિક યાદી (ઍરે અથવા લિસ્ટ) સામાન્યપણે ઉપયોગમાં આવે છે અને તેનો વેક્ટર તરીકે પણ ઉલ્લેખ થાય છે. યાદી (લિસ્ટ) બનાવવા માટે C વિધેયનો ઉપયોગ થાય છે, જે જુદાં-જુદાં નંબરને એક યાદીમાં ભેગા કરે છે. પ્રક્રિયકો જેવા કે +, -, *, / અને બીજા બધા એક નંબર તેમજ યાદી ઉપર એકસરખી રીતે જ કાર્ય કરે છે. તેમ છતાં, જ્યારે દ્વિઅંકી (બાઇનરી) પ્રક્રિયકનાં બંને સંકાર્ય (Operands) યાદી (લિસ્ટ) હોય ત્યારે, તેની લંબાઈ (તેની અંદરનાં તત્ત્વોની સંખ્યા) સરખી હોવી જોઈએ, અચલનું નામ દાખલ કરી અને એન્ટર દબાવવાથી તેનું મૂલ્ય દેખાશે. આ વિભાવનાનું ઉદાહરણ આકૃતિ 13.9માં દર્શાવેલ છે.



આકૃતિ 13.9 : R 'આર'માં અચલનો ઉપયોગ

અહીં > ની નિશાની આરનો પ્રૉમ્પ્ટ દર્શાવે છે. વિધાન a ← 10 'a' નામનો અચલ બનાવે છે અને તેને કિંમત 10 આપે છે. વિધાન > a પછી એન્ટર કી દબાવતા અચલ aની કિંમત દેખાશે. આ જ રીતે વિધાન b ← 20 'b' નામનો અચલ બનાવે છે અને તેને 20 કિંમત આપે છે. વિધાન > a*b પછી એન્ટર કી દબાવતા અચલ 'a' અને 'b' ની કિંમતનો ગુણાકાર કરી, તેનું પરિણામ પ્રૉમ્પ્ટ ઉપર દર્શાવે છે.

કેટલાક સામાન્ય R કમાન્ડ q() આરમાંથી બહાર નીકળવા માટે, help() ઑનલાઇન મદદ મેળવવા માટે, demo() કંઈક પ્રદર્શન જોવા માટે અને help.start() બ્રાઉઝરમાં ઑનલાઇન મદદ ખોલવા માટે વપરાય છે. જો કોઈ નિશ્ચિત વિધેય માટે મદદ જોઈતી હોય, તો આપણે help(function name) વાક્યરચનાનો ઉપયોગ કરવાની જરૂર પડે છે. જ્યારે આપણે Rની બહાર નીકળીએ ત્યારે આપણને પૂછવામાં આવે છે કે આપણે માહિતીને (અચલની કિંમતને) સેવ કરવી છે ? જો આપણે 'હા' કહીએ અથવા "y" કહીએ ત્યારે, વર્તમાન ડિરેક્ટરીની એક ફાઇલમાં આ માહિતી સેવ કરવામાં આવે છે અને જ્યારે આ જ ડિરેક્ટરીમાં ફરી વખત આપણે R શરૂ કરીએ, ત્યારે આ માહિતીને ઉપલબ્ધ કરે છે. આકૃતિ 13.10માં દર્શાવ્યા મુજબ વિધેય ls() બનાવેલા તમામ અચલની યાદી પ્રસિદ્ધ કરે છે.

```
harshal@HarshalAtUbuntu: ~
harshal@HarshalAtUbuntu:~$ R

> a <- 10
> a
[1] 10
> b <- 20
> a*b
[1] 200
> ls()
[1] "a" "b"
>
```

આકૃતિ 13.10 : 'આર'માં ls()નો ઉપયોગ

ક્રમાંકની હારમાળા બનાવવા માટે 'શરૂનો નંબર : અંતનો નંબર' વાક્યરચનાનો ઉપયોગ થાય છે. ઉદાહરણ તરીકે 1:5 એ c(1, 2, 3, 4, 5)ની બરાબર છે.

'આર' આંકડાશાસ્ત્રનાં કાર્યોને તદ્દન નિખાલસ સ્વરૂપે પૂરા પાડે છે. ઉદાહરણ તરીકે આકૃતિ 13.11માં દર્શાવ્યા પ્રમાણે યાદીમાંથી ન્યૂનતમ કિંમત, મહત્તમ કિંમત, મધ્યક અને મધ્યસ્થ શોધવા માટે આપણે min(list), max(list), mean(list) અને median(list) વિધેયનો ઉપયોગ કરી શકશે.

```
harshal@HarshalAtUbuntu: ~
harshal@HarshalAtUbuntu:~$ R
> ll <- c(3,6,8,3,4,6,9,4,7,3,9)
> min(ll)
[1] 3
> max(ll)
[1] 9
> mean(ll)
[1] 5.636364
> median(ll)
[1] 6
>
```

આકૃતિ 13.11 : Rમાં ગાણિતિક વિધેયો

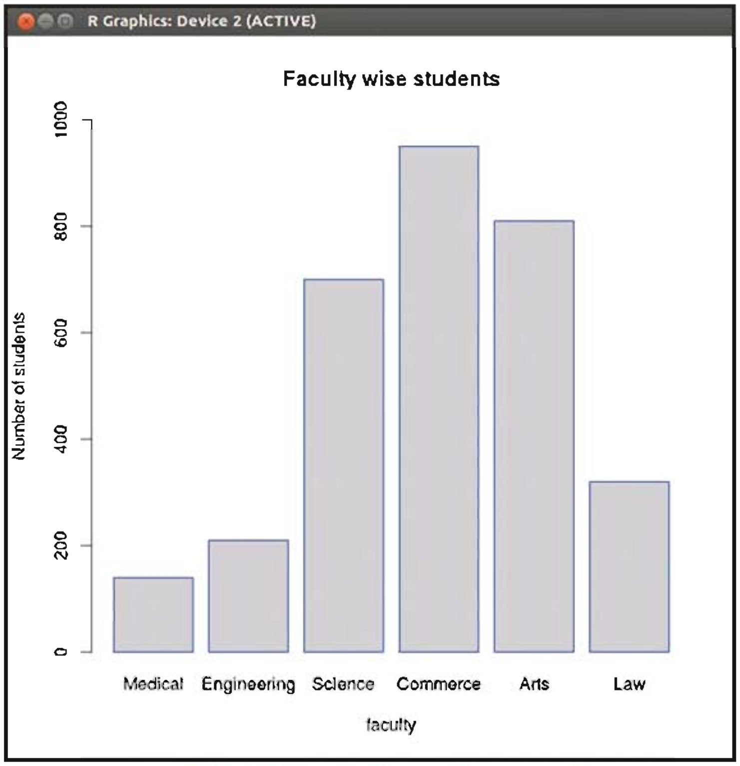
'આર'માં આલેખ દોરવા

ધારો કે આપણે એક બારચાર્ટ દોરવો છે, જે યુનિવર્સિટીમાં વિદ્યાશાખા પ્રમાણે વિદ્યાર્થીઓની સંખ્યા દર્શાવે છે. જે માટે આપણે વિદ્યાર્થીઓની સંખ્યાની વિભાગ અનુસાર એક યાદી અને એક વિદ્યાશાખાની યાદી બનાવીશું. ત્યાર પછી આપણે આકૃતિ 13.12 માં બતાવ્યા પ્રમાણે barplot() વિધેયનો ઉપયોગ કરીને તેમાં જુદી-જુદી આર્ગ્યુમેન્ટ પાસ કરીને આલેખ દોરીશું.

```
harshal@HarshalAtUbuntu: ~
harshal@HarshalAtUbuntu:~$ R
> no_students <- c(140,210,700,950,810,320)
> faculty_names <- c("Medical","Engineering","Science","Commerce","Arts","Law")
> barplot(no_students,main="Faculty wise students",xlab="faculty",
+ ylab="Number of students",names.arg=faculty_names,
+ ylim=c(0,1000),border="blue")
>
```

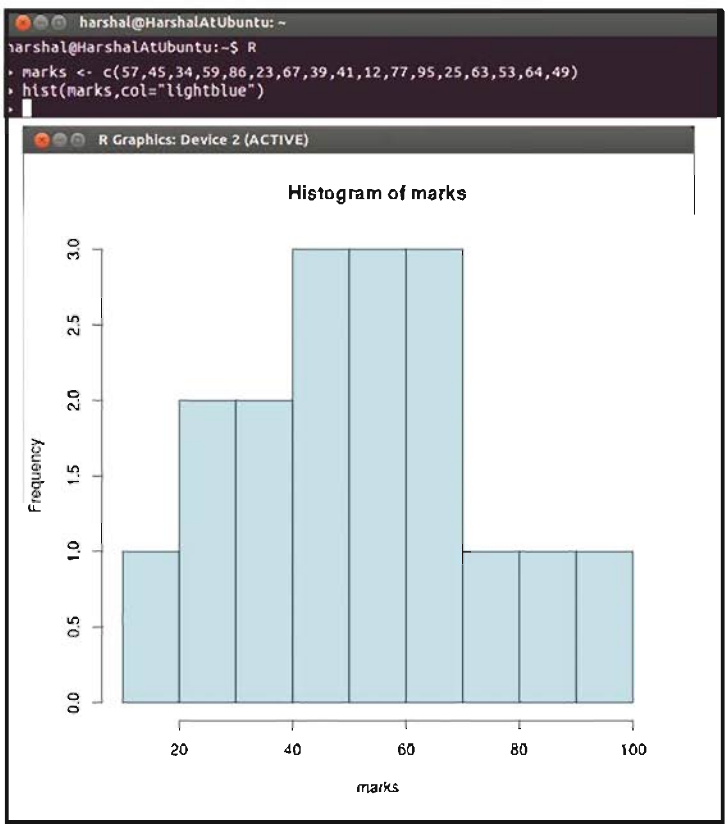
આકૃતિ 13.12 : Rમાં આલેખ દોરવા

પહેલી આર્ગ્યુમેન્ટ માહિતીની યાદી છે અને બાકીની તમામ આર્ગ્યુમેન્ટનું સ્વરૂપ નામ=કિંમત છે. આપણે આર્ગ્યુમેન્ટ તરીકે main, xlab, ylab, names.arg, ylim અને border નો ઉપયોગ કર્યો છે, જે મુખ્ય મથાળું (Title), એક્સ ધરીનું લેબલ, વાય ધરીનું લેબલ, બાર માટેની કિંમત, વાય ધરી ઉપરની કિંમતની રેન્જ અને બોર્ડરનો કલર દર્શાવે છે. નિરીક્ષણ કરો કે જ્યારે કમાન્ડ લાંબો હોય અને કમાન્ડ પૂરો થયા પહેલા આપણે એન્ટર કી દબાવીએ, તો જ્યાં સુધી કમાન્ડ પૂરો ન થાય ત્યાં સુધી R આપણને + પ્રૉમ્પ્ટ આપે છે. આકૃતિ 13.13માં દર્શાવ્યા પ્રમાણે સામાન્યતઃ આલેખને ગ્રીડલાઇન હોતી નથી.



આકૃતિ 13.13 : Rમાં બનાવેલ બારચાર્ટ

હિસ્ટોગ્રામ બનાવવો પણ કઠિન નથી. ધારો કે આપણી પાસે જુદા-જુદા વિદ્યાર્થીઓના કોઈ એક વિષયના 100માંથી મેળવેલા ગુણ છે. આકૃતિ 13.14માં આપેલ કોડનો ઉપયોગ કરીને હિસ્ટોગ્રામ દોરી શકાય, જે આ જ આકૃતિમાં દર્શાવ્યા પ્રમાણેનો દેખાશે.



આકૃતિ 13.14 : Rમાં બનાવેલ હિસ્ટોગ્રામ

રેશનલ પ્લાન (Rational Plan)

જુદા જુદા કાર્યક્ષેત્રમાં કામ કરતા લોકો જેવા કે બાંધકામ, ઇજનેરી સેવા અને સલાહકાર વેપાર ધંધા અથવા સોફ્ટવેર બનાવવા વગેરેને યોજના (Project) ઉપર કામ કરવાનું હોય છે અને આવી યોજનામાં સમય નાણાં અને સાધનોની મુશ્કેલી હોય છે. કોઈ પણ કારણોસર આવી યોજના મોડી પૂરી થાય તો યોજનાનો ખર્ચ વધી જાય છે. આ તબક્કે આવી યોજનાના સંચાલન માટે એક સુવ્યવસ્થિત ગોઠવણી કરવી પડે છે. જે ગોઠવણ નિર્ધારિત સમયમાં અને નાણાંમાં યોજના પૂરી કરવામાં મદદ કરે.

રેશનલ પ્લાન એ એક આવો જ ઓપન સોર્સ સોફ્ટવેર છે જે યોજનાના સંચાલકને આવો પ્લાન બનાવવામાં, નિર્વાહ કરવામાં મદદ કરે છે. તે યોજનાના જીવનચક્ર દરમ્યાન યોજનાના સંચાલકને મદદ કરે છે. આમાં ત્રણ ડેસ્કટોપ ઉત્પાદનનો સમાવેશ થાય છે. રેશનલ પ્લાન સિંગલ, મલ્ટિ અને વ્યૂઅર.

રેશનલ પ્લાન સિંગલ એ સ્વતંત્ર યોજનાના સંચાલન માટે ઉપયોગી છે. તે બીજી યોજના સાથે જોડાયેલ કે કોમન સાધનોનો વપરાશ કરતી અન્ય યોજના સાથે જોડાયેલ માટે હોતો નથી. તે સંચાલકને સામાન્ય યોજના માટેની માહિતી જેવી કે નામ, નોંધ, લીંક, ધારણાઓ, અવરોધ અને જોખમો વગેરે માહિતી યોજના સાથે જોડવા આપે છે. તે વિગતવાર નોંધપત્રક બનાવવા, તેમાં સુધારાવધારા કરવા તથા તેને દૂર કરવાની સગવડ આપે છે. એક વખત સાધન સામગ્રી બની ગયા પછી તેને આપણે કાર્યની ફાળવણી કરી શકીએ છીએ. જે આપણને યોજનાનું ટ્રેકિંગ ટૂલ પૂરું પાડે છે. જેવા કે નિર્ણાયક રસ્તો (Critical Path), કાર્ય માટેની લક્ષ્ય પૂર્ણતાની કિંમત, માહિતીના સમયસર તબક્કાનું કામ અને કિંમત વગેરે. આપણે છાપી શકાય તેવો અહેવાલ બનાવી શકીએ છીએ અને બીજી યોજના સંચાલનના ટૂલમાંથી માહિતી આયાત કરી શકીએ છીએ અને બીજા માળખામાં માહિતી નિકાસ પણ કરી શકીએ છીએ.

રેશનલ પ્લાન મલ્ટિ એ એવી યોજનાનું સંચાલન કરવામાં મદદ કરે છે કે જેમાં એક કંપનીના સાધનો જુદી-જુદી યોજનામાં વહેંચાયેલાં હોય. તે એકબીજા ઉપર આધારિત યોજનાનું સંચાલન પણ કરે છે. તેમાં રેશનલ પ્લાન સિંગલમાં આપેલ તમામ લક્ષણ (ફિચર્સ)નો સમાવેશ થાય છે. વધારેમાં તે, દરેક યોજનામાં રોકાયેલ સાધનોની માહિતી (કામ, કિંમત અને વધારે ફાળવણી)ની ગણતરી પણ કરી આપે છે. તે જુદી જુદી યોજનાઓ સાથે સંબંધિત કાર્યનું જોડાણ, યોજનાની માહિતીનું વિશ્લેષણ અને યોજનાનો પૉર્ટફોલિયો બનાવવાની પરવાનગી આપે છે.

રેશનલ પ્લાન વ્યૂઅર એ એક વધારાનું ટૂલ છે જે મૂળ ફાઇલના માળખાં (.xrp)માં રહેલ યોજનાને વહેંચવા (શેર કરવા) માટે બનાવેલ છે. પણ તેનો ઉપયોગ માઇક્રોસૉફ્ટ યોજનાની ફાઇલ ખોલવા માટે પણ થાય છે. સાધનોને ફાળવેલી કામગીરી જોવા માટે તેમજ ગોઠવેલ કાર્યપદ્ધતિમાં કોઈ પણ પ્રકારનો ફેરફાર કર્યા વગર યોજનાનો વિકાસ જોવા માટે તે ખૂબજ ઉપયોગી છે.

ઉબુન્ટુ 10.04માં રેશનલ પ્લાન આવતું નથી અને સમર્થન પણ કરતું નથી તે ફક્ત ઉબુન્ટુ 12.04 અને ત્યાર પછીના વર્ઝનમાં જ ઉપલબ્ધ છે.

સ્કાઇપ (Skype)

તમે કેટલીય સંદેશા માટેની જુદી જુદી સેવાઓ જેવી કે યાહૂ મૅસેન્જર, ગુગલટોક અથવા રેડિફબોલનો ઉપયોગ કર્યો હશે, જેનો ઉપયોગ આપણે વાસ્તવિક સમયમાં ગપસપ (ચેટિંગ) માટે કરીએ છીએ અને તેમાં લખાણ, દૃશ્ય અને શ્રાવ્યમાહિતીનો ઉપયોગ પણ કરીએ છીએ. સ્કાઇપ આવું જ એક સૉફ્ટવેર છે, જે આપણને કમ્પ્યૂટરમાં ઇન્ટરનેટનો ઉપયોગ કરીને કોલ કરવા માટેની સગવડ આપે છે.

સ્કાઇપ ઉપયોગકર્તાને અવાજ, લખાણ અને વીડિયોનો ઉપયોગ કરીને બીજા સાથે સંપર્ક કરવાની પરવાનગી આપે છે. આપણે આપણા ટેલીફોન નેટવર્કમાં ફોન કરવા માટે પણ સ્કાઇપનો ઉપયોગ કરી શકીએ. ટેલીફોન અથવા મોબાઇલ ફોન ઉપર કરેલા ફોનનો ખર્ચ ઉપયોગકર્તાના ખાતામાં ઉધાર થાય છે અને સ્કાઇપમાં કરેલ ફોન કોલ નિઃશુલ્ક હોય છે. તે ફાઇલને એક જગ્યાએથી બીજી જગ્યાએ ફેરવવાની તેમજ વીડિયો-કોન્ફરન્સની વધારાની સગવડ પણ પૂરી પાડે છે.

સ્કાઇપ સૉફ્ટવેરને નિઃશુલ્ક ડાઉનલોડ કરી શકાય છે, પરંતુ તેનો સૉર્સકોડ માલિકીનો છે, તેમાં કોઈ પણ પ્રકારના સુધારાવધારા કરી શકાતા નથી. સ્કાઇપનો ઉપયોગ કરવો હોય, તો અવાજ(સાઉન્ડ)નું ઇનપુટ અને આઉટપુટ કાર્યરત હોવું જોઈએ.

આધુનિક કમ્પ્યૂટરમાં આ સગવડ હોય જ છે. લેપટોપમાં તો અંદર જ આપેલ હોય છે જ્યારે ડેસ્કટૉપ કમ્પ્યૂટરમાં આપણે હેડફોન, સ્પીકર અને વેબકેમેરાને બહારથી જોડવાં પડે છે. આપણે સ્કાઇપને સૉફ્ટવેર કેન્દ્રમાંથી પ્રસ્થાપિત કરી શકીએ છીએ.

સ્કાઇપ શરૂ કરવા માટે, **Applications → Internet → Skype** વિકલ્પ પસંદ કરો. જો તે તમે પ્રથમ વખત જ ખોલતા હશો, તો એક ઉપયોગકર્તા માટે પરવાના માટેનાં કરારની વિન્ડો ખોલશે. તેનું લખાણ વાંચી અને "I agree" બટન ઉપર ક્લિક કરો. હવે આપણે આકૃતિ 13.15માં આપેલ વિન્ડો જોઈ શકીશું.

આકૃતિ 13.15 : સ્કાઇપની શરૂઆતની વિન્ડો

સ્કાઇપસેવાનો ઉપયોગ કરવા માટે ઉપયોગકર્તાનું ખાતું હોવું જોઈએ. જો ન હોય તો, "Don't have a Skype Name yet?" ઉપર ક્લિક કરો અને સ્કાઈપ નામ બનાવવા માટેનાં પગલાંને અનુસરો. એક વખત તમારી પાસે સ્કાઇપ નામ અને પાસવર્ડ આવી ગયા પછી આકૃતિ 13.15માં બતાવેલ ટેક્સ્ટબૉક્સમાં માહિતી ભરો અને "Sign in" બટન ક્લિક કરો. જો બધું જ બરાબર હશે તમે બીજા સ્કાઇપના ઉપયોગકર્તા સાથે સંવાદ કરી શકશો અથવા સ્કાઇપનો ઉપયોગ કરીને ફોનકૉલ કરી શકશો.

સારાંશ

આ પ્રકરણમાં આપણે કેટલાંક ઉપયોગી નિઃશુલ્ક ટૂલ્સ અને સેવાઓની ચર્ચા કરી. આપણે માહિતી-સંકોચનની તકનિક વિશે ભણ્યા જે માહિતીસંગ્રહ કરવા માટેની જગ્યામાં ઘટાડો કઈ રીતે કરી શકાય તે શીખવે છે. આ તકનિકનો ઉપયોગ ફાઇલ અને ડિરેક્ટરી ઉપર કરવા માટેનું ટૂલ આર્ચિવ (આર્કાઇવ્ઝ) મેનેજર પણ જોયું. આપણે મલ્ટિમીડિયા પ્લેયર VLCની પણ ચર્ચા કરી. આપણે ગુગલ નકશા સેવાની પણ ચર્ચા કરી, જે આપણને જુદા-જુદા વિસ્તારના નકશાઓ ઑનલાઇન પૂરા પાડે છે. લખાણમાં જુદી-જુદી ભાષાનાં અક્ષરો અને ચિહ્નોને ઉમેરવા માટેના અક્ષરનકશો (કૅરેક્ટર મૅપ) વિશે પણ અભ્યાસ કર્યો. ગાણિતિક ગણતરી માટેના 'આર' એન્વાયરમેન્ટ ઉપર પણ ચર્ચા કરી, જેમાં આપણે સામાન્ય ગણતરી જેવી કે મધ્યસ્થ અને મધ્યક શોધવા વિશે ચર્ચા કરી અને ''આર''માં બારચાર્ટ અને હિસ્ટોગ્રામ કેવી રીતે બનાવાય તે વિશે પણ ભણ્યા અને છેલ્લે આપણે રેશનલ પ્લાન અને સ્કાઇપનો ઉપયોગ પણ જોઈ ગયા.

શિક્ષક માટે સૂચના

આ પ્રકરણમાં આપેલ ટૂલ્સ ફક્ત પ્રદર્શન માટે જ છે. શિક્ષકે વિદ્યાર્થીને આ ટૂલ્સ ડેમો સ્વરૂપે દર્શાવવાના છે. પ્રાયોગિક સ્વાધ્યાયમાં આપેલા ઉદાહરણનો ઉપયોગ કરીને શિક્ષકે વિદ્યાર્થીઓને આ ટૂલ્સનો ઉપયોગ દેખાડવાનો છે. શિક્ષકે કમ્પ્યૂટરમાં 'આર' સૉફ્ટવેર પ્રસ્થાપિત કરવું પડશે. R પ્રસ્થાપિત કરવા માટે નીચે પ્રમાણેનો કમાન્ડ આપો :

sudo apt-get install r-base r-base-dev

સ્વાધ્યાય

1. માહિતી-સંકોચન વિશે ટૂંકનોંધ લખો.

2. આપણને માહિતી-સંકોચન ટૂલની જરૂરત શા માટે છે ?

3. આર્ચિવ (આર્કાઇવ્ઝ) મૅનેજરનાં મુખ્ય લક્ષણોની યાદી બનાવો.

4. VLC મીડિયાપ્લેયરનાં મુખ્ય લક્ષણો જણાવો.

5. ગુગલ નકશાના વિનિયોગની યાદી બનાવો.

6. કૅરેક્ટરમૅપ (અક્ષરનકશો) પ્રોગ્રામનો ઉપયોગ સમજાવો.

7. **આપેલા વિકલ્પમાંથી સાચો વિકલ્પ પસંદ કરો :**

(1) નીચેનામાંથી કઈ ફાઇલ તેમાં રહેલ આખા ડિરેક્ટરી-માળખાનો ઉલ્લેખ કરે છે ?

(a) અપાચે (b) આર્ચી (c) આર્થિ (d) આર્ચિવ (આર્કાઇવ્ઝ)

(2) ટાર (TAR)નું આખું નામ શું છે ?

(a) ટેપ આર્ચિવર (b) ટેક આર્ચિવર (c) ટેસ્ટ આર્ચિવર (d) ટાઇટ આર્ચિવર

(3) ક્યા આર્ચિવ પ્રકારમાં પાસવર્ડ વિકલ્પ ઉપલબ્ધ હોય છે ?

(a) ઝીપ (b) ટાર (c) tar.gz (d) ઝીપ અને tar.gz બંને

(4) નીચેનામાંથી ક્યું મીડિયાપ્લેયર લાક્ષણિકતાની દૃષ્ટિએ સમૃદ્ધ છે ?

(a) VAC (b) VEC (c) VLC (d) VNC

(5) VLCનું આખું નામ શું છે ?

(a) વીડિયોલેન ક્લાયન્ટ (b) વીડિયો લાઇનકોડર

(c) વીડિયો લેન્થકોડર (d) વીડિયો લિસ્ટ ક્રિએટર

(6) કઈ તકનિક આપણા સ્થાનનો ચોક્કસ ખ્યાલ આપે છે ?

(a) GRS (b) GPRS (c) GRPS (d) GPS

(7) કોઈ પણ વિનિયોગમાં અક્ષરો દાખલ કરવા ક્યા પ્રોગ્રામનો ઉપયોગ થાય છે ?

(a) કૅરેક્ટર ડિસ્પ્લે (b) કૅરેક્ટર ઇન્સર્ટ

(c) કૅરેક્ટર મૅપ (અક્ષરનકશો) (d) કૅરેક્ટર સિલેક્ટ

(8) Rમાંથી બહાર નીકળવા માટે ક્યો કમાન્ડ વપરાય છે ?

(a) quit() (b) q() (c) exit() (d) close()

(9) Rમાં બારગ્રાફ બનાવવા માટે ક્યા વિધેયનો ઉપયોગ થાય છે ?

(a) bar() (b) plot() (c) bargraph() (d) barplot()

(10) નીચેનામાંથી રેશનલ પ્લાનનાં જુદાં-જુદાં સ્વરૂપ ક્યા છે ?

(a) સિંગલ, મલ્ટિ, વ્યૂઅર (b) સિંગ્યુલર, મલ્ટિપલ

(c) વ્યૂ, પ્રિવ્યૂ (d) સર્વર, ક્લાયન્ટ

પ્રાયોગિક સ્વાધ્યાય

1. ઝીપફાઇલ બનાવી તેમાં આખી ડિરેક્ટરીનો સંગ્રહ કરો.
2. tar.gz ફાઇલ બનાવી, તેમાં આખી ડિરેક્ટરીનો સંગ્રહ કરો.
3. સબડિરેક્ટરી ફાઇલ1 (files1)માં ઝીપ આર્ચિવમાંથી બધી જ ફાઇલો બહાર કાઢો.
4. સબડિરેક્ટરી ફાઇલ2 (files2)માં tar.gz આર્ચિવમાંથી બધી જ ફાઇલો બહાર કાઢો.
5. ગુગલ નકશામાં તમારી શાળા શોધો. સેટેલાઇટ તસવીરમાં તમે તમારી ઇમારત ઓળખી શકો છો ?
6. ગુગલ નકશામાં તમારું ઘર શોધો. તમારા ઘરેથી શાળા સુધી પહોંચવા માટેનો માર્ગ શોધો. આ રસ્તો એ જ છે, જેનો તમે દરરોજ ઉપયોગ કરો છો ?
7. કેરેક્ટરમેપ (અક્ષરનકશો)નો ઉપયોગ કરીને geditમાં सत्यमेवजयते લખો.
8. કેરેક્ટરમેપ (અક્ષરનકશો)નો ઉપયોગ કરીને geditમાં $\sin(\alpha - \beta) = \sin\alpha \ \cos\beta - \ \cos\alpha \sin\beta$ લખો.
9. વર્ષના મહિનાના દરેક દિવસોનો બારગ્રાફ બનાવો.
10. પાઠ્યપુસ્તકના પાનાંનો બારગ્રાફ બનાવો.
11. અઠવાડિયાનાં દિવસોનાં નામમાં (રવિવાર, સોમવાર...... શનિવાર) વારંવાર આવતા અક્ષરો માટેનો હિસ્ટોગ્રામ દોરો. ફક્ત એવા અક્ષરોનો જ સમાવેશ કરો જે ઓછામાં ઓછા એક વખત આવતા હોય.
12. તમારું પોતાનું સ્કાઇપ ઉપયોગકર્તાખાતું બનાવો અને સ્કાઇપસેવા દ્વારા આપવામાં આવતી સગવડનું નિરીક્ષણ કરો.

• • •