ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ અને પ્રોગ્રામિંગ

પરિચય

આપણે અગાઉ અલ્ગોરિધમનો ઉપયોગ કરીને સમસ્યાનું નિરાકરણ કેવી રીતે કરવું તે વિશે શીખ્યા અને સમજ્યા કે અલ્ગોરિધમ કમ્પ્યૂટર પ્રોગ્રામ્સ જેવા હોય છે. સમસ્યાનું નિરાકરણ શીખ્યા પછી, આપણે C પ્રોગ્રામિંગ વિષે અભ્યાસ કર્યો, જેમાં આપણે C ભાષાના સિન્ટેક્સનો ઉપયોગ કરીને પ્રોગ્રામ લખ્યા. C એક પ્રોસિજરલ ભાષા (procedural language) છે જે પ્રોગ્રામ લખવા માટે એક પછી એક સૂચનાઓનો ઉપયોગ કરે છે. પ્રોસિજરલ પ્રોગ્રામિંગ ભાષા પ્રોગ્રામને ક્રમબદ્ધ પગલામાં આપેલ સૂચનાઓના સંદર્ભમાં ગોઠવે છે, જેના કારણે પ્રોગ્રામ મોટા અને જટિલ બનતા જાય છે. તેનું વ્યવસ્થાપન કરવું અને તેને ફરીથી ઉપયોગમાં લેવા મુશ્કેલ બની જાય છે. તે વાસ્તવિક-દુનિયાની એન્ટિટી (વસ્તુઓ)ની આસપાસ પ્રોગ્રામનો વિકાસ કે ગોઠવણ કરતું નથી અને તેના એક ભાગમાં કરવામાં આવેલો ફેરફાર અન્ય ભાગોને અનપેક્ષિત રીતે અસર કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ (Object Oriented Programming-OOP) પ્રોગ્રામને ઑબ્જેક્ટ નામની વાસ્તવિક-દુનિયાની એન્ટિટીની આસપાસ ગોઠવે છે. ઉદાહરણ તરીકે, કાર (car) એક ઑબ્જેક્ટ છે જેમાં રંગ, મોડેલ વગેરે જેવા એટ્રિબ્યુટ્સ (attributes) હોય છે અને તે ચાલી (move) શકે છે અને અટકી (stop) શકે છે. ઑબ્જેક્ટ ક્લાસ (class)માંથી બનાવવામાં આવે છે, જે ઑબ્જેક્ટ માટે ટેમ્પ્લેટ અથવા બ્લુપ્રિન્ટ તરીકે કાર્ય કરે છે. ઉદાહરણ તરીકે, કારની એક ડિઝાઇનમાંથી, આપણે ઘણી સમાન કારનું ઉત્પાદન કરી શકીએ છીએ. તમે કમ્પ્યૂટર અથવા મોબાઇલ પર જે રમતો રમી હશે, તે ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને બનાવવામાં આવેલી હોય છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ કોડને ખૂબ જ સ્પષ્ટ, સમજવામાં સરળ અને સુધારવા તથા વ્યવસ્થાપન કરવા માટે સરળ રાખવાની મંજૂરી આપે છે. તે ઑબ્જેક્ટ, ક્લાસ, એન્કેપ્સ્યુલેશન (encapsulation), એબ્સ્ટ્રેક્શન (abstraction), ઈન્હેરિટન્સ (inheritance) અને પોલિમોર્ફિઝમ (polymorphism) જેવા ખ્યાલોની આસપાસ કાર્ય કરે છે. આ પ્રકરણમાં આ તમામ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના ખ્યાલોને વાસ્તવિક-દુનિયાના ઉદાહરણો સાથે શીખીશું. આ પ્રકરણના અંતમાં ખૂબ જ લોકપ્રિય અને વ્યાપકપણે ઉપયોગમાં લેવાતી ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓનો સંક્ષિપ્ત પરિચય પણ મેળવીશું.

પ્રોસિજરલ પ્રોગ્રામિંગની મર્યાદાઓ (Limitations of Procedural Programming)

1990 પહેલાં લોકો પ્રોગ્રામ લખવા માટે પ્રોસિજરલ ભાષાઓ (procedural languages) જેમ કે BASIC, FORTRAN, COBOL, Pascal વગેરેનો ઉપયોગ કરતા હતા. પ્રોસિજરલ પ્રોગ્રામમાં ક્રમબદ્ધ સૂચનાઓ લખવામાં આવે છે. અગાઉના પ્રકરણોમાં આપણે C ભાષામાં પ્રોગ્રામ લખ્યા છે, જ્યાં આપણે આખો પ્રોગ્રામ *main()* ફંક્શનમાં સ્ટેટમેન્ટના સ્વરૂપમાં લખીએ છીએ. પ્રોસિજરલ પ્રોગ્રામિંગ નાના અથવા મધ્યમ કદના પ્રોગ્રામ માટે યોગ્ય છે. પરંતુ જ્યારે મોટા પ્રોગ્રામની વાત આવે છે, ત્યારે તેમાં ઘણા પ્રશ્નો ઉભા થાય છે. ચાલો, આપણે તેમને સંક્ષિપ્તમાં સમજીએ:

- મનુષ્યો ઑબ્જેક્ટના સ્વરૂપમાં વિશ્વને સરળતાથી સમજી અને તેની સાથે ક્રિયા-પ્રતિક્રિયા કરી શકે છે, જે સહજ રીતે જટિલ સમસ્યાઓનું નિરાકરણ લાવે છે. પ્રોસિજરલ પ્રોગ્રામિંગ પ્રોગ્રામને ક્રમબદ્ધ પગલામાં આપેલ સૂચનાઓ તરીકે લખવાની અનુમતિ આપે છે, જે મુખ્યત્વે પ્રક્રિયા (process) એટલે કે ફંક્શન/પ્રોસિજર (Functions/Procedures) પર ધ્યાન કેન્દ્રિત કરે છે, ઑબ્જેક્ટ્સ પર નહીં. અહીં માનવ વિશ્વને જે રીતે સમજે છે અને તેની સાથે ક્રિયા-પ્રતિક્રિયા કરે છે તેને પ્રતિબિંબિત કરતું નથી. તેથી, ક્યારેક

Get More Learning Materials Here : CLICK HERE www.studentbro.in

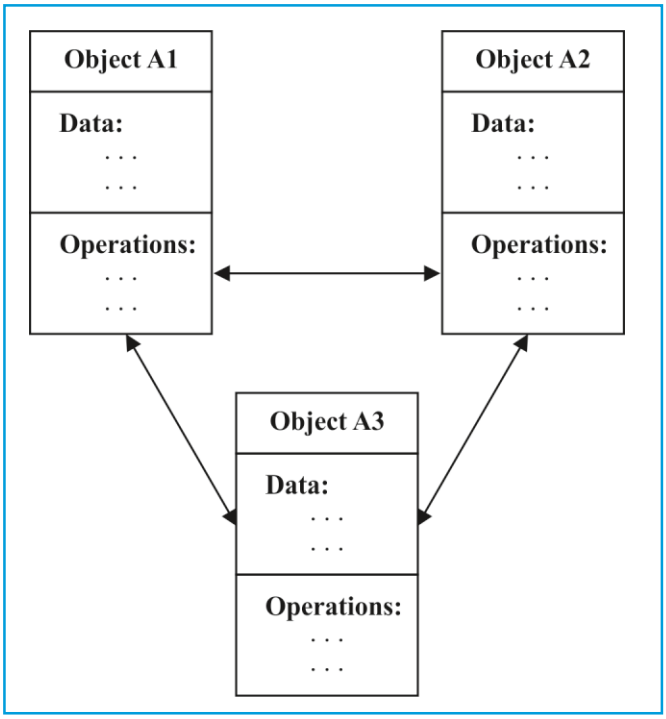
પ્રોગ્રામ મોટો અને જટિલ બની જાય છે, ત્યારે પ્રોસિજરલ પ્રોગ્રામિંગમાં સમસ્યાઓનું વ્યવસ્થાપન, સમજણ અને નિરાકરણ કરવું મુશ્કેલ બની જાય છે.

- પ્રોસિજરલ પ્રોગ્રામિંગમાં ડેટા અને ફંક્શન (ફંક્શન એ ચોક્કસ કાર્ય રજૂ કરતી એક નાની પ્રક્રિયા છે) અલગ હોય છે અને તેઓ કોઈપણ પ્રતિબંધો વિના એકબીજા સાથે ભળી જાય છે. આનાથી ગૂંચવણ થાય છે અને પ્રોગ્રામમાં ફેરફાર કરવો અથવા સમસ્યા હલ કરવી મુશ્કેલ બની જાય છે. એવું પણ શક્ય છે કે કોડ ઘણી વખત પુનરાવર્તિત થાય, જેનાથી પ્રોગ્રામ બિનજરૂરી રીતે લાંબો બને છે. ડેટાનો અનિયંત્રિત ઍક્સેસ, ઉદાહરણ તરીકે C માં ગ્લોબલ વેરીએબલ (global variable) સુરક્ષાની ચિંતાઓ પણ ઊભી કરે છે. તેનાથી પ્રોગ્રામનો કોઈપણ ભાગ અકસ્માતે મહત્ત્વપૂર્ણ ડેટા બદલી શકે છે.
- વર્તમાન એપ્લિકેશન ખૂબ જ મોટી અને જટિલ હોય છે, કારણકે તે મોટા પ્રમાણમાં ડેટા સાથે કામ કરે છે અને વિવિધ પ્રકારની સેવાઓ પ્રદાન કરે છે. આવા સંજોગોમાં, વ્યવસાયને કોડના સંગઠનની એવી જરૂર છે જે ભૂલોને સુધારવા, સેવાઓ ઉમેરવા અથવા બદલવા વગેરે સહિત ફેરફારોને ત્વરિત સમર્થન આપે. પ્રોસિજરલ પ્રોગ્રામિંગ સારી રીતે સંરચિત છે, પરંતુ તે પ્રોગ્રામ અથવા સૉફ્ટવેરનું આયોજન જે રીતે કરે છે તેના કારણે મોટી અને જટિલ એપ્લિકેશન્સ માટે ઝડપથી અપડેટ અને ફેરફારોની સરળતા પ્રદાન કરતું નથી. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ પ્રોગ્રામને ઑબ્જેક્ટના સ્વરૂપમાં ગોઠવીને આ ફાયદો આપે છે, જેની સાથે વ્યવહાર કરવો સરળ બને છે કારણ કે મનુષ્યો સહજ રીતે રોજિંદા જીવનમાં ઑબ્જેક્ટ સાથે કામ કરવા સક્ષમ છે.

ઉપરોક્ત તથ્યોને જોતાં, આધુનિક સોફ્ટવેર ડેવલપમેન્ટ માટે ઉદ્યોગો ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓનો ઉપયોગ કરે છે. જે ખ્યાલ પર ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ કાર્ય કરે છે તેને સમજવું ખૂબ જ જરૂરી છે. તેને ઑબ્જેક્ટ ઓરિએન્ટેડ ખ્યાલ અથવા સિદ્ધાંતો કહેવામાં આવે છે. આ પ્રકરણમાં આપણે આધુનિક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ દ્વારા સમર્થિત ઑબ્જેક્ટ ઓરિએન્ટેડ ખ્યાલની વિગતવાર ચર્ચા કરીશું.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ (Object-Oriented Approach)

આપણે જાણીએ છીએ કે વાસ્તવિક દુનિયા ઑબ્જેક્ટની બનેલી છે. વાસ્તવિક દુનિયાની કોઈપણ સમસ્યા કેટલાક ઑબ્જેક્ટ, તેમના સંબંધો અને તેમની વચ્ચેના સંચાર (communication)ની બનેલી હોય છે. ચાલો, એક ઘરનું ઉદાહરણ લઈએ જેમાં એક નાનો પરિવાર રહે છે. ઘરના બેડરૂમમાં પલંગ, ટેબલ, કબાટ વગેરે હોય છે. રસોડું રસોઈ માટે વપરાય છે, જેમાં ગેસ સ્ટવ, ડીશ, કપ, મિક્સર, ફ્રિજ અને અન્ય ઘણી વસ્તુઓનો સમાવેશ થાય છે. હૉલમાં સોફા, ટેલિવિઝન સેટ અને અન્ય ઉપયોગી ફર્નિચર ગોઠવાયેલા હોય છે. ઘરમાં પંખા, એસી અને અન્ય ઇલેક્ટ્રોનિક ઉપકરણો પણ છે. ઘરમાં ઘણી પ્રવૃત્તિઓ થાય છે જેમ કે રસોઈ કરવી, સાથે બેસીને વાતો કરવી, એસી ચાલુ-બંધ કરવું વગેરે. પરિવારમાં માતા-પિતા-બાળક, પતિ-પત્ની, પિતા-પુત્ર જેવા સંબંધો છે. પરિવારના સભ્યો



આકૃતિ 11.1 : ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામનું માળખું

Get More Learning Materials Here : CLICK HERE www.studentbro.in

વિવિધ સુવિધાઓનો ઉપયોગ તેમની સાથે ક્રિયા-પ્રતિક્રિયા (interacting) કરીને કરે છે, ઉદાહરણ તરીકે, શાકભાજી કાઢવા માટે ફ્રિજ ખોલવું. ઘરનું આ વર્ણન વાસ્તવિક દુનિયામાં એક સિસ્ટમ અથવા સમસ્યાનું ઉદાહરણ છે. તેની રચના શેના પર આધારિત છે? વિવિધ ઑબ્જેક્ટ. પરિવારનો દરેક સભ્ય, બેડરૂમ, ગેસ સ્ટવ, ડીશ, કપ, પંખા, સોફા, એસી વગેરે એ ઑબ્જેક્ટના ઉદાહરણો છે. ઘરમાં રહેલી વસ્તુઓનો ઉપયોગ પરિવારના સભ્યો દ્વારા થાય છે અને પરિવારના દરેક સભ્ય એકબીજા સાથે સંબંધિત છે. આથી, ઘરને મનુષ્યો અને વિવિધ નિર્જીવ વસ્તુઓ સહિત, ચોક્કસ રીતે ગોઠવાયેલા વિવિધ ઑબ્જેક્ટનું સંગઠન કહી શકાય.

ઑબ્જેક્ટ શેનાથી બને છે? ચાલો આપણે એક પંખાને ધ્યાનમાં લઈએ. પંખામાં રંગ, કિંમત, મોડેલ, તેની બનાવટનું વર્ષ વગેરે જેવા કેટલાક એટ્રિબ્યુટ્સ હોય છે. આપણે પંખા પર ચાલુ (ON) કે બંધ (OFF) કરવો, સ્પીડ વધારવા કે ઘટાડવા જેવા ઓપરેશન કરી શકીએ છીએ. તેથી, ઑબ્જેક્ટને ગુણધર્મો (properties) અને વર્તન (behaviour) (તેના પર કરવામાં આવતી ક્રિયાઓ જેને ફંક્શન અથવા મેથડ પણ કહેવાય છે) તરીકે વર્ણવવામાં આવે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ ઑબ્જેક્ટ પર એક બિલ્ડિંગ બ્લોક (building block) તરીકે ધ્યાન કેન્દ્રિત કરે છે અને પ્રોગ્રામ બનાવે છે. આપણે કહી શકીએ કે ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનું કેન્દ્ર ઑબ્જેક્ટ છે, જેમાં ડેટા (ગુણધર્મો) અને ઓપરેશન્સ (ફંક્શન્સ અથવા મેથડ્સ)નો સમાવેશ થાય છે. એક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામમાં ઘણા વિવિધ પ્રકારના ઑબ્જેક્ટ હોઈ શકે છે જે અમુક રીતે એકબીજા સાથે સંબંધિત હોય છે અને તેઓ વિવિધ પ્રવૃત્તિઓ અથવા કાર્યો કરવા માટે એકબીજા સાથે સંચાર (communicate) કરે છે. આકૃતિ 11.1 ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામનું માળખું દર્શાવે છે.

આકૃતિમાં ત્રણ ઑબ્જેક્ટ દર્શાવ્યા છે, જે દરેક પોતાનો ડેટા (data) અને મેથડ (method) ધરાવે છે. ઑબ્જેક્ટને જોડતા તીર (arrow) દર્શાવે છે કે ઑબ્જેક્ટ્સ એકબીજાના ફંક્શનને કૉલ કરીને એકબીજા સાથે સંચાર (communicate) કરે છે. મોટા પ્રોગ્રામમાં, ઑબ્જેક્ટનો એક સમૂહ કોઈક રીતે એકબીજા સાથે સંબંધિત હોઈ શકે છે.

પિઝાના નાના રેસ્ટોરન્ટનું નીચેનું ઉદાહરણ પ્રક્રિયાગત (procedural) અને ઑબ્જેક્ટ ઓરિએન્ટેડ (object-oriented) એમ બંને પ્રકારના અભિગમને સ્પષ્ટ કરશે.

પ્રક્રિયાગત અભિગમ (Procedural Way) : આપણે કણક (dough) કેવી રીતે બનાવવો, કેવી રીતે બેક કરવો, ઑર્ડર કેવી રીતે લેવો અને ટેબલ કેવી રીતે સાફ કરવા તેની પ્રક્રિયાના પગલાંની એક યાદી તૈયાર કરીએ છીએ. જ્યારે ગ્રાહક પિઝાનો ઑર્ડર આપે છે, ત્યારે આપણે દર વખતે મેન્યુઅલમાં આપેલી યાદીને ઉપરથી નીચે સુધી અનુસરીએ છીએ. હવે ધારો કે આપણે સેન્ડવીચ (sandwich) ઉમેરી રહ્યા છીએ, તો તમે છેલ્લે એક પગલું ઉમેરી રહ્યા છો, ભલે કેટલાક પગલાં સામાન્ય હોય. કોઈપણ ફેરફાર માટે સમગ્ર મેન્યુઅલનો સંદર્ભ લેવો પડે છે અને જેમ જેમ વધુ વસ્તુઓ ઉમેરાતી જાય છે તેમ તેમ તે ગૂંચવણભર્યું બનતું જાય છે.

ઑબ્જેક્ટ અભિગમ (Object Way) : આપણી પાસે મેનૂ, પિઝા મેકર, વેઈટર, કૅશિયર વગેરે જેવા ઑબ્જેક્ટની સ્પષ્ટ યાદી છે. આ બધા પોતપોતાનું કાર્ય ખૂબ સારી રીતે જાણે છે અને જ્યારે સોંપવામાં આવે ત્યારે તે કાર્ય ખૂબ સારી રીતે કરે છે. જો આપણે નવી વસ્તુ ઉમેરીએ, જેમ કે વિવિધ પ્રકારના પિઝા, તો આપણે માત્ર પિઝામેકરની સૂચનાઓ જ અપડેટ કરવાની જરૂર છે. કૅશિયર માત્ર રોકડનું સંચાલન કરે છે અને અન્ય વિગતોની દેખરેખ રાખતો નથી. વેઈટર માત્ર ડિલિવરી પર ધ્યાન કેન્દ્રિત કરે છે. આથી, તેમાંથી કોઈ એકની સૂચનાઓમાં ફેરફાર કરવાથી અન્યને અસર થતી નથી.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમ (Object Oriented Concepts)

ઑબ્જેક્ટ એ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો પાયાનો ભાગ છે. વાસ્તવિક દુનિયાની સિસ્ટમ વિવિધ ઑબ્જેક્ટ અને તેમના સંબંધોની બનેલી હોય છે. અગાઉ ચર્ચા કરેલ નાના પરિવારવાળા ઘરના ઉદાહરણને યાદ

Get More Learning Materials Here : CLICK HERE www.studentbro.in

રાખો. જોકે ઑબ્જેક્ટ એ ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના કેન્દ્રમાં છે, પરંતુ સિસ્ટમમાં ઑબ્જેક્ટ અને અન્ય ઑબ્જેક્ટ સાથેના તેમના સંબંધોને લાક્ષણિક બનાવવા માટે આપણે વિવિધ ખ્યાલોને સમજવા પડશે. આનાથી આપણે વાસ્તવિક દુનિયાની સિસ્ટમને યોગ્ય રીતે અને ચોક્કસાઈથી સમજી શકીશું, જેને પછી ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને અમલમાં મૂકી શકાય છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ અભિગમમાં નીચેના ખ્યાલોનો સમાવેશ થાય છે:

- ઑબ્જેક્ટ (Object)
- ક્લાસ (Class)
- એબ્સ્ટ્રેક્શન (Abstraction)
- એન્કેપ્સ્યુલેશન (Encapsulation) અને ડેટા હાઈડિંગ (Data hiding)
- મેસેજ પાસિંગ (Message passing)
- કમ્પોઝિશન (Composition) અને એગ્રિગેશન (Aggregation)
- ઈન્હેરીટન્સ (Inheritance)
- પોલિમોર્ફિઝમ (Polymorphism)

ચાલો, આપણે એક પછી એક ઉદાહરણો સાથે આ ખ્યાલોની ચર્ચા કરીએ.

ઑબ્જેક્ટ (Objects)

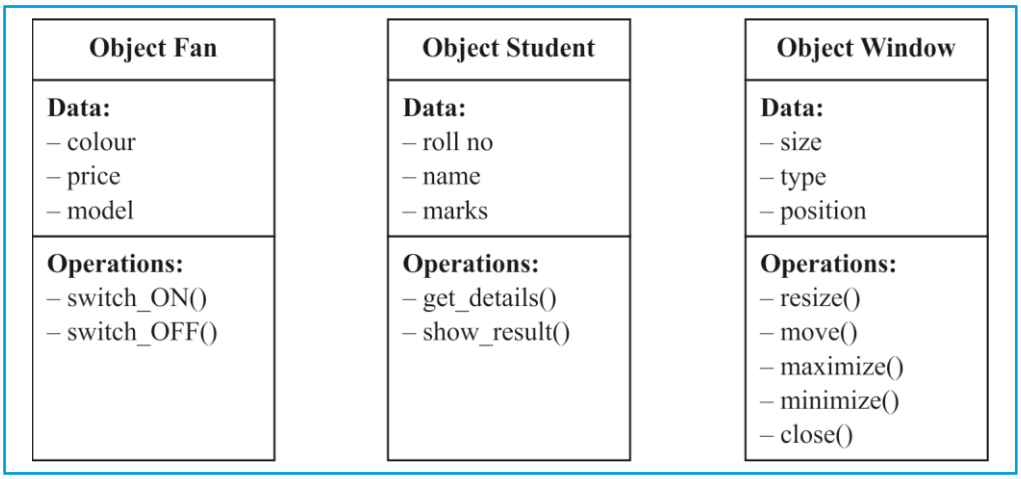
ઑબ્જેક્ટને વાસ્તવિક દુનિયાની એન્ટીટી તરીકે વ્યાખ્યાયિત કરવામાં આવે છે. દૈનિક જીવનમાં શિક્ષક, સ્કૂલ બેગ, સાયકલ, બ્લેકબોર્ડ, મોબાઈલ ફોન અને અન્ય અનેક એન્ટીટી સાથે કાર્ય કરીએ છીએ. આ બધા ઑબ્જેક્ટનાં ઉદાહરણો છે. વાસ્તવિક દુનિયાના ઑબ્જેક્ટ પ્રોપર્ટી (properties) અને બિહેવિયર (behaviours)ના બનેલા હોય છે.

પ્રોપર્ટી (Properties) : આને ઑબ્જેક્ટ્સની લાક્ષણિકતાઓ (characteristics) અથવા એટ્રિબ્યુટ્સ (attributes) પણ કહેવામાં આવે છે. ઉદાહરણ તરીકે, પંખાના ગુણધર્મો રંગ (colour), કિંમત (price) અને મૉડેલ (model) છે. પ્રોગ્રામિંગમાં ઑબ્જેક્ટની પ્રોપર્ટીને ડેટા (data) તરીકે રજૂ કરવામાં આવે છે. ચાલો વિદ્યાર્થીને એક ઑબ્જેક્ટ તરીકે લઈએ. પ્રોગ્રામમાં તેની પ્રોપર્ટી રોલ નંબરને પૂર્ણાંક (integer) ડેટા પ્રકાર તરીકે, નામને સ્ટ્રિંગ (string) તરીકે અને તેના માર્ક્સ ને ફ્લોટિંગ-પોઈન્ટ (floating-point) નંબર તરીકે રજૂ કરી શકીએ.

બિહેવિયર (Behaviour) : બિહેવિયર એટલે કે ઑબ્જેક્ટને જ્યારે કહેવામાં આવે ત્યારે ઑબ્જેક્ટ દ્વારા કરવામાં આવતી ક્રિયા (action). બિહેવિયરને ઑબ્જેક્ટ પર કરવામાં આવતા ઑપરેશન તરીકે પણ વ્યાખ્યાયિત કરવામાં આવે છે. ઉદાહરણ તરીકે, જ્યારે પંખો ચાલુ થાય છે, ત્યારે તે કાર્ય કરવાનું શરૂ કરશે. તેવી જ રીતે, વિદ્યાર્થી પરીક્ષા આપી શકે છે. પ્રોગ્રામમાં ઑપરેશનને ફંક્શનનો ઉપયોગ કરીને વ્યાખ્યાયિત કરવામાં આવે છે (જેને મેથડ (methods) પણ કહેવામાં આવે છે). જ્યારે આપણને આપેલ ઑબ્જેક્ટ પર કોઈ ચોક્કસ ઑપરેશન કરવાની જરૂર હોય, ત્યારે ફંક્શન અથવા મેથડને કૉલ કરવામાં આવે છે.

ઑબ્જેક્ટને લંબચોરસના રૂપમાં ઑબ્જેક્ટ ડાયાગ્રામનો ઉપયોગ કરીને રજૂ કરી શકાય છે, જેના ત્રણ ભાગ હોય છે: ઑબ્જેક્ટનું નામ, પ્રોપર્ટી અને ઑપરેશન. આકૃતિ 11.2 ઑબ્જેક્ટનાં ત્રણ અલગ-અલગ ઉદાહરણો દર્શાવે છે.

Get More Learning Materials Here : CLICK HERE www.studentbro.in



આકૃતિ 11.2 : ઑબ્જેક્ટ ડાયાગ્રામ (ઉદાહરણ : પંખો, વિદ્યાર્થી, વિન્ડો)

આકૃતિમાં દર્શાવ્યા પ્રમાણે પંખાને રંગ, કિંમત અને મૉડેલ જેવી તેની પ્રોપર્ટી સાથે વ્યાખ્યાયિત કરવામાં આવ્યો છે, અને તેના ઑપરેશનમાં પંખાને ચાલુ (switch_ON()) અને બંધ (switch_OFF()) કરવાનો સમાવેશ થાય છે. વિદ્યાર્થીને તેની પ્રોપર્ટી રોલ નંબર, નામ અને માર્ક્સ સાથે વ્યાખ્યાયિત કરવામાં આવેલ છે, અને વિદ્યાર્થીની વિગતો મેળવવા (get_details()) અને પરિણામ બતાવવા (show_result()) તે તેના ઑપરેશન છે. ત્રીજું ઉદાહરણ એ કમ્પ્યૂટર સૉફ્ટવેરમાં આવેલી વિન્ડો (Window) ઑબ્જેક્ટનું છે. તેને તેના ગુણધર્મો જેમ કે તેનું કદ, પ્રકાર અને સ્ક્રીન પર તેનું સ્થાન સાથે વ્યાખ્યાયિત કરવામાં આવેલ છે. તેના પર ઑપરેશન જેમ કે, તેનું કદ બદલવું (resize()), અલગ સ્થાન પર ખસેડવું (move()), વિન્ડો મોટી કરવી (maximize()), વિન્ડો નાની કરવી (minimize()), વિન્ડો બંધ કરવી (close()) વગેરે કરી શકાય છે.

જુદી જુદી શ્રેણીઓના ઑબ્જેક્ટના ઉદાહરણો નીચે આપેલા છે:

મનુષ્ય : વિદ્યાર્થી, શિક્ષક, કર્મચારી

ફર્નિચર : પલંગ, સોફા, ટેબલ, ખુરશી, બેન્ચ

ઇલેક્ટ્રોનિક વસ્તુઓ : મોબાઇલ, કમ્પ્યૂટર, ટેલિવિઝન, હાર્ડ ડિસ્ક

ભૌમિતિક આકાર : બિંદુ, ચોરસ, લંબચોરસ, ત્રિકોણ, સમઘન, ગોળો

GUI ઘટક : વિન્ડો, મેનૂ, લિસ્ટ, ચેકબોક્સ, રેડિયો બટન

યુઝર દ્વારા નિર્ધારિત ડેટા : સમય, બિંદુ, રેલવે ટિકિટ, વિદ્યાર્થીનું પરિણામ

ઇમેજ ફાઇલ : JPG, PNG, GIF

ડોક્યુમેન્ટ ફાઇલ : વર્ડ ફાઇલ, સ્પ્રેડશીટ, PDF, ટેક્સ્ટ ફાઇલ, C ફાઇલ

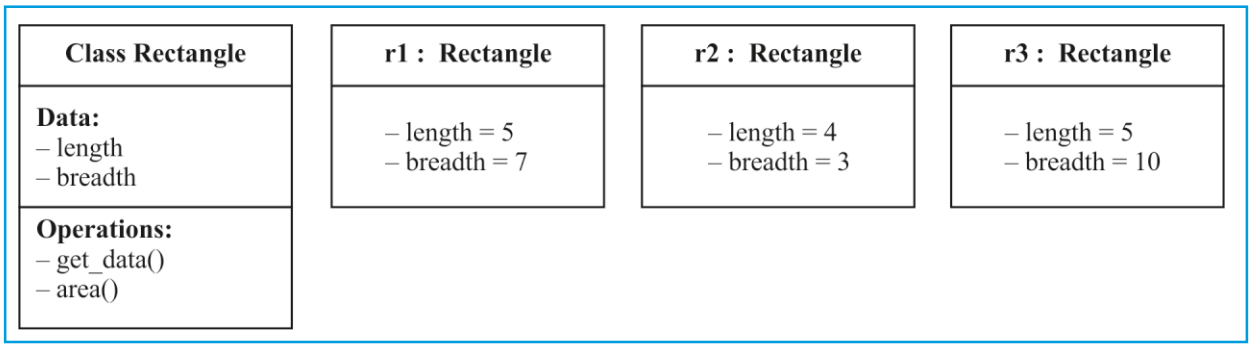
ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગ હંમેશાં એ કેન્દ્રમાં રાખે છે કે આપેલ સમસ્યાને ઑબ્જેક્ટમાં કેવી રીતે વહેંચવામાં આવે (જ્યારે પ્રોસીજરલ પ્રોગ્રામિંગમાં એ કેન્દ્રમાં હોય છે કે સમસ્યાને પ્રક્રિયાઓ એટલે કે ફંક્શન્સમાં કેવી રીતે વહેંચવામાં આવે). ઑબ્જેક્ટની દ્રષ્ટિએ વિચારવું મનુષ્યો માટે સરળ બને છે, કારણ કે આપણે હંમેશાં વાસ્તવિક દુનિયાની સિસ્ટમને એવા ઑબ્જેક્ટ તરીકે સમજીએ છીએ જેની સાથે આપણે સરળતાથી સંવાદ કરી શકીએ છીએ.

ક્લાસ (Class)

ચાલો, એક ઑબ્જેક્ટ - Student પર વિચાર કરીએ. એક વર્ગમાં ઘણા વિદ્યાર્થીઓ હોય છે જે પ્રોપર્ટી અને બિહેવીયર જેવી સમાન વિગતો ધરાવે છે. સમાન પ્રોપર્ટી અને બિહેવીયર ધરાવતા ઑબ્જેક્ટ એક જૂથ અથવા

Get More Learning Materials Here : CLICK HERE www.studentbro.in

ક્લાસ બનાવે છે જેને ઑબ્જેક્ટ ક્લાસ (Object Class) તરીકે ઓળખવામાં આવે છે. તેને સામાન્ય રીતે ફક્ત ક્લાસ કહેવાય છે. ક્લાસ તેનામાંથી તારવવામાં આવેલા તમામ ઑબ્જેક્ટ દ્વારા વહેંચાયેલી પ્રોપર્ટી અને ઓપરેશનનું વર્ણન કરે છે. તેથી જ ક્લાસ એક ડિઝાઈન ડોક્યુમેન્ટ અથવા ટેમ્પલેટ તરીકે કાર્ય કરે છે. જેમકે, કારની ડિઝાઈન સમાન હોય તો તેમાંથી ઘણી સમાન કારનું ઉત્પાદન કરી શકાય છે. એકવાર ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી તેમાંથી ગમે તેટલા ઑબ્જેક્ટ બનાવી શકાય છે. સમાન ક્લાસ ટેમ્પ્લેટમાંથી ઉતરી આવેલા દરેક ઑબ્જેક્ટને તે ક્લાસનો ઈન્સ્ટન્સ (instance) કહેવામાં આવે છે. તેથી, દરેક વ્યક્તિગત ઑબ્જેક્ટ તે ક્લાસનો ઈન્સ્ટન્સ છે જેમાંથી તે ઉતરી આવ્યો છે. ઘણીવાર ક્લાસ અને ઑબ્જેક્ટ શબ્દોનો ઉપયોગ એકબીજાના બદલે થાય છે, પરંતુ સંદર્ભ અર્થ સ્પષ્ટ કરી શકે છે. ક્લાસને ક્લાસ ડાયાગ્રામનો ઉપયોગ કરીને દર્શાવવામાં આવે છે. આકૃતિ 11.3 ક્લાસ અને ઈન્સ્ટન્સનું ઉદાહરણ દર્શાવે છે.



આકૃતિ 11.3 : લંબચોરસ (Rectangle) ક્લાસ અને તેના ઈન્સ્ટન્સ

આકૃતિમાં Rectangle નામનો એક ક્લાસ વ્યાખ્યાયિત કરેલો છે જેમાં length (લંબાઈ) અને breadth (પહોળાઈ) પ્રોપર્ટી અને get_data તથા area ઓપરેશન આપેલાં છે. આકૃતિમાં આ ક્લાસના ત્રણ ઈન્સ્ટન્સ r1, r2, અને r3 દર્શાવેલા છે. જોઈ શકાય છે કે આ ત્રણેય ઈન્સ્ટન્સ સમાન માળખું ધરાવે છે, પરંતુ તેમની લંબાઈ અને પહોળાઈના મૂલ્યોના કારણે તેઓ એકબીજાથી અલગ પડે છે. એક ઈન્સ્ટન્સના બધા મૂલ્યોના સમૂહને તે ઑબ્જેક્ટની સ્થિતિ (state) કહેવામાં આવે છે. તે પણ શક્ય છે કે બે ઈન્સ્ટન્સ માટે બંને ગુણધર્મોના મૂલ્યો સમાન હોય, તેમ છતાં પણ તેમને અલગ ગણી શકાય, કારણ કે કમ્પ્યૂટરમાં બંનેને અલગ મેમરી ફાળવવામાં આવે છે. ચાલો, હવે આપણે C++ માં Rectangle વર્ગને વ્યાખ્યાયિત કરીએ.

```
class Rectangle {
    private:
        /* properties */
        int length;
        int breadth;
    public:
        /* Operations */
        void getData() {
                ...
        }
        int area() {
              ...
        }
};
```


Get More Learning Materials Here : CLICK HERE www.studentbro.in

ગૂંચવણ ટાળવા માટે અહીં અમલની વિગતો આપવામાં આવી નથી. અહીં આપેલો ક્લાસ માત્ર એક ટેમ્પ્લેટની વ્યાખ્યા કરે છે અને તેને કોઈ મેમરી ફાળવવામાં આવી નથી. નીચે આપેલું main() ફંક્શન દર્શાવે છે કે કેવી રીતે ક્લાસમાંથી ઑબ્જેક્ટ બનાવવામાં (instantiate કરવામાં) આવે છે:

```
void main( )
{
    Rectangle r1, r2, r3;
    …
}
```

આ ડેટાના પ્રકાર મુજબ ચલ બનાવ્યા જેવું છે, એટલે કે Rectangle પ્રકારના ઑબ્જેક્ટ બનાવવા. એકવાર ઈન્સ્ટન્સ બનાવીએ, પછી તેને મેમરી ફાળવવામાં આવે છે. તેનો અર્થ એ છે કે ઉપરોક્ત કિસ્સામાં, ત્રણ ઈન્સ્ટન્સ r1, r2 અને r3 માટે અલગ-અલગ મેમરી ફાળવવામાં આવે છે, અને તે બધા સમાન કાર્યો (getData() અને area())ને વહેંચે છે. getData() ઓપરેશનનો ઉપયોગ ઑબ્જેક્ટ માટેની લંબાઈ અને પહોળાઈ મેળવવા માટે થાય છે અને area() નો ઉપયોગ તેની મેમરીમાં સંગ્રહિત લંબાઈ અને પહોળાઈના મૂલ્યોનો ઉપયોગ કરીને આપેલ ઈન્સ્ટન્સનું ક્ષેત્રફળ ગણવા માટે થાય છે.

રોજિંદા જીવનમાં આપણે ઘણી એવી સિસ્ટમનો ઉપયોગ કરીએ છીએ જ્યાં આપણે સમાન ઑબ્જેક્ટના સમૂહનો ઉપયોગ કરીએ છીએ. તમે જે શાળામાં અભ્યાસ કરો છો તે તેનું એક ઉદાહરણ છે. શાળામાં ઘણા શિક્ષકો અને વિદ્યાર્થીઓ હોય છે. શિક્ષકો વિવિધ ધોરણોમાં અલગ-અલગ વિષયો ભણાવે છે. આપણે Teacher, Student, Subjects વગેરે માટે ક્લાસ બનાવી શકીએ છીએ, જેથી તેમના ટેમ્પ્લેટ (એટલે કે તેમની પ્રોપર્ટી અને ઓપરેશન) વ્યાખ્યાયિત કરી શકાય. એકવાર આ ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી જરૂરિયાત મુજબ આ દરેક ક્લાસના જેટલા જોઈએ તેટલા ઈન્સ્ટન્સ (objects) બનાવી શકીએ છીએ. વિદ્યાર્થીઓને તેમના વર્ગો અને ધોરણો પ્રમાણે જૂથબદ્ધ પણ કરી શકાય છે. શિક્ષક દ્વારા ચોક્કસ વિભાગને ભણાવવામાં આવતા વિષયને સંગ્રહિત કરવા માટે સિસ્ટમમાં teacher-subject-division મેપિંગ પણ બનાવી શકાય છે. સંપૂર્ણ સિસ્ટમને, સિસ્ટમની જરૂરિયાત પ્રમાણે ઉપર જણાવેલ ઑબ્જેક્ટ ઉપરાંત વધારાના ઑબ્જેક્ટની પણ જરૂર પડી શકે છે.

એબ્સ્ટ્રેક્શન (Abstraction)

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં એબ્સ્ટ્રેક્શન એક ખૂબ જ મહત્વપૂર્ણ ખ્યાલ છે. એબ્સ્ટ્રેક્શનનો અર્થ એ છે કે બિનજરૂરી વિગતો છુપાવવી અને માત્ર જરૂરી વિગતોને જ પ્રકાશિત કરવી. આપણે જાણીએ છીએ કે ઑબ્જેક્ટ તેની પ્રોપર્ટી અને ઓપરેશન દ્વારા વ્યાખ્યાયિત કરવામાં આવે છે. વાસ્તવિક દુનિયામાં એક ઑબ્જેક્ટમાં મોટી સંખ્યામાં પ્રોપર્ટી હોય છે. શું આપણે તે બધાને પ્રોગ્રામમાં ધ્યાનમાં લેવા જોઈએ કે તેમાંથી ફક્ત અમુકને જ? અહીં એબ્સ્ટ્રેક્શન મદદરૂપ થાય છે. આપણે ફક્ત તે જ પ્રોપર્ટીને ધ્યાનમાં લેવાની હોય છે જે આપેલ વાસ્તવિક-વિશ્વની સમસ્યા માટે સોફ્ટવેર સિસ્ટમને અમલમાં મૂકવા માટે જરૂરી હોય. આને ડેટા એબ્સ્ટ્રેક્શન (data abstraction) તરીકે પણ ઓળખી શકાય છે. આ જ બાબત ઓપરેશનને પણ લાગુ પડે છે.

ચાલો, આને એક ઉદાહરણ દ્વારા સમજીએ. આપણે અગાઉના વિભાગમાં Rectangle (લંબચોરસ) વર્ગને ફક્ત length (લંબાઈ) અને breadth (પહોળાઈ) સાથે વ્યાખ્યાયિત કર્યો હતો. જોકે, જો આપણે ગ્રાફિકલ સિસ્ટમ વિકસાવી રહ્યા હોઈએ, તો લંબચોરસમાં અન્ય પ્રોપર્ટી પણ હોઈ શકે છે, જેમ કે તેને રંગથી ભરવા માટે વપરાતો fill-color, position (સ્થાન), thickness of broder (બોર્ડરની જાડાઈ), fill-style (રંગ ભરવાની સ્ટાઈલ) અને અન્ય ઘણા ગુણધર્મો પણ તેમાં હોઈ શકે છે. આ બધા ગુણધર્મો ગ્રાફિકલ સિસ્ટમ માટે આવશ્યક છે, પરંતુ આપણા કિસ્સામાં આપણે ફક્ત ક્ષેત્રફળની ગણતરી કરવા માટે લંબાઈ અને પહોળાઈ મેળવવા માંગતા હતા. તેથી fill-color,

Get More Learning Materials Here : CLICK HERE www.studentbro.in

position, thickness, fill-style જેવી પ્રોપર્ટીને છુપાવવામાં આવ્યા હતા. ગ્રાફિકલ સિસ્ટમ માટે, આપણને changeColor (રંગ બદલવો), move (સ્થાન બદલવું) જેવા ઓપરેશનની પણ જરૂર પડે, જે અગાઉના વિભાગમાં છુપાવેલા રાખવામાં આવ્યા હતા.

એબ્સ્ટ્રેક્શન જટિલ સિસ્ટમ્સને સરળ બનાવે છે. તે કોડિંગ સ્તર પર ખૂબ જ સ્પષ્ટતા પૂરી પાડે છે. તે માત્ર જરૂરી વિગતો પર ધ્યાન કેન્દ્રિત કરીને વધુ સારી ડિઝાઈન બનાવવામાં સક્ષમ બનાવે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં ક્લાસની વ્યાખ્યા પોતે જ એબ્સ્ટ્રેક્શનનું એક ઉદાહરણ છે, કારણ કે તે વાસ્તવિક-વિશ્વની એન્ટિટીને માત્ર આવશ્યક વિગતો સાથે સંક્ષિપ્ત કરે છે.

એન્કેપ્સ્યુલેશન અને ડેટા હાઈડિંગ (Encapsulation and Data Hiding)

આપણે શીખ્યા તે મુજબ ઑબ્જેક્ટ ડેટા (માહિતી) અને ઑપરેશન (મેથડ)નો બનેલો હોય છે. ક્લાસ એક જ યુનિટમાં પ્રોપર્ટી (ડેટા) અને ઑપરેશન (મેથડ) બંનેને વ્યાખ્યાયિત કરે છે, જેનું નામ વાસ્તવિક દુનિયાની એન્ટીટી જેવું જ હોય છે. ડેટા અને ઑપરેશનને એક જ યુનિટમાં જોડવાની પ્રક્રિયાને એન્કેપ્સ્યુલેશન કહેવામાં આવે છે. ક્લાસ ડેટા અને મેથડ બંનેને એક જ નામ હેઠળ એન્કેપ્સ્યુલેટ કરે છે, જેમ કે અગાઉ ચર્ચા કરેલ Rectangle ક્લાસ. એન્કેપ્સ્યુલેશન બે સૌથી મહત્વપૂર્ણ પાસાંઓ રજૂ કરે છે:

- વાસ્તવિક દુનિયાની એન્ટીટીને વ્યાખ્યાયિત કરવા માટે ડેટા અને મેથડને એકસાથે મૂકે છે. આ સિસ્ટમ ડિઝાઈનને ખૂબ જ સ્પષ્ટ બનાવે છે, જે અન્ય કોઈ પણ ઑબ્જેક્ટને અસર કર્યા વિના સિસ્ટમમાં ઑબ્જેક્ટની આંતરિક વિગતો બદલવાની મંજૂરી આપે છે.
- ઑબ્જેક્ટના અમલ જેવા આંતરિક પાસાંને બાહ્ય પાસાં એટલે કે, સિસ્ટમના બાકીના ભાગથી અલગ પાડે છે. તે બાહ્ય દુનિયાથી ડેટા છુપાવે છે અને ઑબ્જેક્ટની મેથડનો ઉપયોગ કરીને જ તેને સુરક્ષિત રીતે ઍક્સેસ કરવાની મંજૂરી આપે છે. ડેટાને બાહ્ય દુનિયાથી છુપાવવાની આ પ્રક્રિયાને ડેટા હાઈડિંગ (Data Hiding) કહેવામાં આવે છે. ઑબ્જેક્ટના ડેટાનો ઉપયોગ કરવા માટે, બાહ્ય દુનિયા, એટલે કે સિસ્ટમના અન્ય ઑબ્જેક્ટ, ટાર્ગેટ ઑબ્જેક્ટની મેથડને કૉલ કરે છે અને ઑબ્જેક્ટના છુપાયેલા ડેટા પર કોઈપણ ઑપરેશન કરવા માટે તેમની સાથે સંવાદ કરે છે.

મેસેજ પાસિંગ (Message Passing)

વાસ્તવિક દુનિયાના ઑબ્જેક્ટ કાર્યો કરવા માટે એકબીજાને સંદેશા મોકલીને અને પ્રાપ્ત કરીને સંવાદ કરે છે. આપણે કહી શકીએ કે મેસેજ પાસિંગ વાસ્તવિક દુનિયામાં સંદેશાવ્યવહારની પદ્ધતિનું મોડેલિંગ કરે છે, જે વિવિધ પ્રવૃત્તિઓનું સંચાલન કરે છે. ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગનો ઉપયોગ કરીને વિકસાવવામાં આવેલ સોફ્ટવેર અથવા પ્રોગ્રામ વાસ્તવિક દુનિયાની સમસ્યાનો ઉકેલ છે. તેથી વપરાશકર્તાઓને કાર્યો કરીને સેવાઓ પ્રદાન કરવા માટે પ્રોગ્રામમાં વ્યાખ્યાયિત ઑબ્જેક્ટ્સ વચ્ચે સંદેશાવ્યવહાર માટે તે સમાન પદ્ધતિનો ઉપયોગ કરે છે. એકવાર ક્લાસ વ્યાખ્યાયિત થઈ જાય, પછી આપણે તે ક્લાસમાંથી ઑબ્જેક્ટ્સના ઇન્સ્ટન્સ બનાવી શકીએ છીએ અને પછી ઑબ્જેક્ટ્સ સંદેશા મોકલી અથવા પ્રાપ્ત કરી શકે છે. અગાઉ વ્યાખ્યાયિત કરેલ Rectangle ક્લાસને ધ્યાનમાં લઈએ, જ્યાં *main()* ફંક્શનમાં, આપણે ત્રણ ઇન્સ્ટન્સ r1, r2 અને r3 બનાવ્યા હતા. નીચેનું વિધાન એક મેસેજ પાસિંગનું ઉદાહરણ છે:

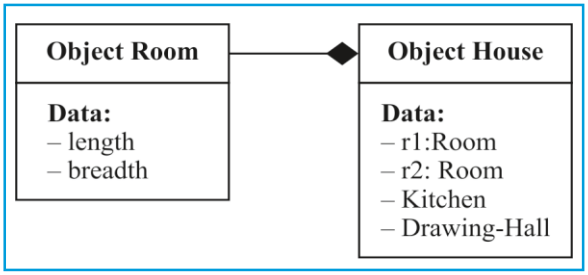
```
a = r1.area( );
```

અહીં, એપ્લિકેશન દ્વારા મોકલાયેલો સંદેશ ઑબ્જેક્ટ r1 ક્ષેત્રફળની ગણતરી કરવા માટે પ્રાપ્ત કરે છે. જવાબમાં, એપ્લિકેશન તેના વેરિયેબલ a માં ક્ષેત્રફળનું મૂલ્ય પ્રાપ્ત કરે છે. એક ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામ જરૂરી ક્લાસને વ્યાખ્યાયિત કરે છે, ક્લાસને ઇન્સ્ટન્સિએટ કરીને ઑબ્જેક્ટ્સ બનાવે છે, અને અંતે આ ઑબ્જેક્ટ્સ તેમનું કાર્ય પૂર્ણ કરવા માટેની પ્રવૃત્તિઓ સિદ્ધ કરવા માટે સંદેશા મોકલી અને પ્રાપ્ત કરીને વાતચીત કરે છે.

Get More Learning Materials Here : CLICK HERE www.studentbro.in

કમ્પોઝીશન અને એગ્રીગેશન (Composition and Aggression)

ઘણી વખત, વાસ્તવિક દુનિયાનો એક ઑબ્જેક્ટ બીજા ઘણા ઑબ્જેક્ટનો, એટલે કે ભાગનો, બનેલો હોય છે. આવા સંજોગોમાં, ભાગ (part) અને મુખ્ય ઑબ્જેક્ટ (main object) વચ્ચેનો સંબંધ પાર્ટ-ઓફ (part-of) અથવા હેઝ-એ (has-a) પ્રકારનો હોય છે. જો ભાગ અને મુખ્ય ઑબ્જેક્ટ વચ્ચેનો સંબંધ પાર્ટ-ઓફ હોય, તો તેને કમ્પોઝિશન કહેવામાં આવે છે. અને જો ભાગ અને મુખ્ય ઑબ્જેક્ટ વચ્ચેનો સંબંધ હેઝ-એ હોય, તો તેને ઍગ્રિગેશન કહેવામાં આવે છે. ચાલો આપણે વાસ્તવિક દુનિયાના ઉદાહરણો દ્વારા કમ્પોઝિશન અને ઍગ્રિગેશનને સમજીએ.

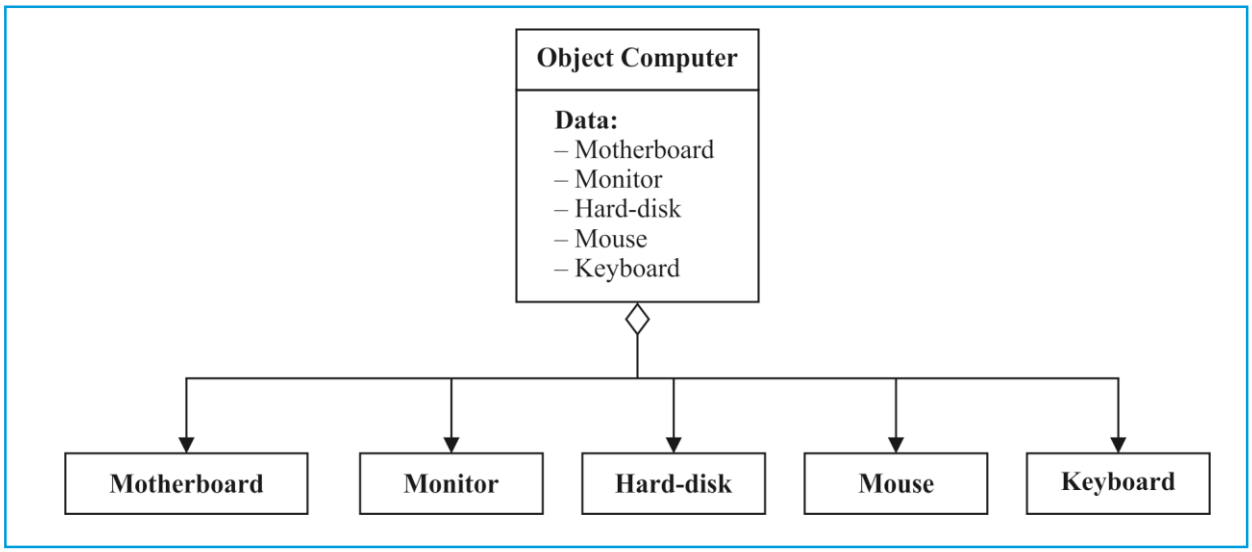


આકૃતિ 11.4 : કમ્પોઝીશનનું ઉદાહરણ

કમ્પોઝિશન (Composition) : આપણે જાણીએ છીએ કે કોઈપણ ઘર એક અથવા વધુ રૂમ, રસોડું, અને ડ્રોઈંગ હૉલનું બનેલું હોય છે. આપણે કહી શકીએ કે ઘર એ રૂમ, રસોડું અને ડ્રોઈંગ હૉલ દ્વારા બનેલું છે. એનો અર્થ એ થયો કે ઘર એક ઑબ્જેક્ટ છે જે અન્ય ઑબ્જેક્ટ (ભાગ)નું બનેલું છે. ઘર એક સંપૂર્ણ ઑબ્જેક્ટ છે, જ્યારે રૂમ, રસોડું અને ડ્રોઈંગ હૉલ તેના ભાગ છે, જે પોતે પણ ઑબ્જેક્ટ છે. આ અભિગમને આકૃતિ 11.4 માં બતાવ્યા પ્રમાણે કમ્પોઝિશનનો ઉપયોગ કરીને મોડેલ કરી શકીએ છીએ. સરળતા માટે અહીં ફક્ત ઘર અને રૂમ બતાવ્યા છે.

આ એક કન્ટેનરશિપ (containership) જેવું છે, જેમ કે ઘર રૂમને સમાવે છે. જો ઘરમાં બે રૂમ હોય, તો આકૃતિમાં બતાવ્યા પ્રમાણે આપણી પાસે ઘર ઑબ્જેક્ટના સભ્ય તરીકે બે રૂમ ઑબ્જેક્ટ હશે. કમ્પોઝિશન દર્શાવવા માટે રંગ ભરેલા ડાયમંડ (solid diamond)નો ઉપયોગ થાય છે. કમ્પોઝિશન એક મજબૂત અથવા વિશિષ્ટ સંબંધ (exclusive relationship) છે, જ્યાં સુધી સંપૂર્ણ ઑબ્જેક્ટ અથવા પેરન્ટ ઑબ્જેક્ટ અસ્તિત્વમાં હોય ત્યાં સુધી જ તેનો ભાગ અસ્તિત્વમાં રહે છે. એનો અર્થ એ છે કે, રૂમ, રસોડું અને ડ્રોઈંગ હૉલ સ્વતંત્ર રીતે અસ્તિત્વ ધરાવતા નથી અને જો ઘર દૂર કરવામાં આવે (નાશ કરવામાં આવે) તો તેઓ પણ દૂર થઈ જાય છે.

એગ્રિગેશન (Aggregation) : કમ્પ્યૂટર સિસ્ટમ (computer system) મધરબોર્ડ, મોનિટર, હાર્ડ ડિસ્ક, માઉસ અને કીબોર્ડ જેવા વિવિધ ભાગોનું બનેલું હોય છે. આપણે કહી શકીએ કે કમ્પ્યૂટરની પાસે મધરબોર્ડ છે, મોનિટર છે, હાર્ડ ડિસ્ક છે, માઉસ છે અને કીબોર્ડ છે. આ સંબંધ આકૃતિ 11.5 માં બતાવ્યા પ્રમાણે પોલા ડાયમન્ડ સાથે દર્શાવવામાં આવે છે. સરળતા માટે, ઘટક ઑબ્જેક્ટની પ્રોપર્ટી અહીં દર્શાવવામાં આવી નથી.



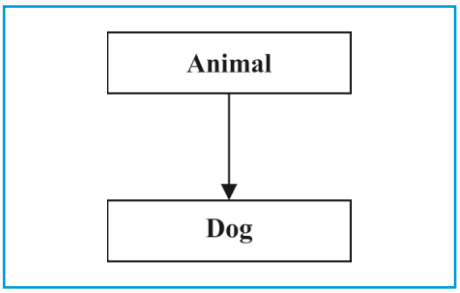
આકૃતિ 11.5 : એગ્રીગેશનનું ઉદાહરણ

Get More Learning Materials Here : CLICK HERE www.studentbro.in

એગ્રિગેશન (Aggregation) એ નિર્બળ અથવા નોન-એક્સક્લુઝીવ સંબંધ (non-exclusive relationship) છે, કારણકે તેના ભાગ સંપૂર્ણ ઑબ્જેક્ટ અથવા પેરન્ટ ઑબ્જેક્ટ વિના પણ સ્વતંત્ર રીતે અસ્તિત્વ ધરાવી શકે છે. ઉદાહરણ તરીકે, માઉસ કમ્પ્યૂટર વિના પણ અસ્તિત્વ ધરાવી શકે છે.

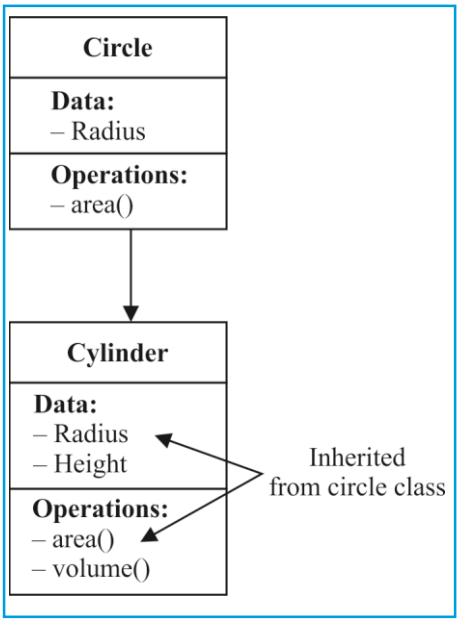
ઈન્હેરીટન્સ (Inheritance)

ઈન્હેરીટન્સ એક એવી પ્રક્રિયા છે જેના દ્વારા એક ક્લાસ બીજા ક્લાસની લાક્ષણિકતાઓ વારસામાં મેળવે છે. જે ક્લાસમાંથી લાક્ષણિકતાઓ વારસામાં મેળવવામાં આવે છે તેને બેઝ ક્લાસ (Base Class) અથવા પેરન્ટ ક્લાસ (Parent Class) કહેવામાં આવે છે. અને જે ક્લાસ લાક્ષણિકતાઓ વારસામાં મેળવે છે તેને ડિરાઈવ્ડ ક્લાસ (Derived Class) અથવા ચાઈલ્ડ ક્લાસ (Child Class) કહેવામાં આવે છે. આકૃતિ 11.6 ઈન્હેરીટન્સ દર્શાવે છે જ્યાં ડોગ (Dog) એ પેરન્ટ ક્લાસ એનિમલ (Animal)નો ચાઈલ્ડ ક્લાસ છે. સ્વાભાવિક છે કે શ્વાન એ પ્રાણીની ઘણી લાક્ષણિકતાઓ વારસામાં મેળવે છે, કારણ કે શ્વાન પણ એક પ્રાણી જ છે. ઈન્હેરીટન્સ "is-a" સંબંધનું પ્રતિનિધિત્વ કરે છે.



આકૃતિ 11.6 : ઈન્હેરીટન્સનું ઉદાહરણ

ઈન્હેરીટન્સ સંબંધિત ઑબ્જેક્ટને એકસાથે જૂથબદ્ધ કરવાની મંજૂરી આપે છે, જે કેટલીક સામાન્ય લાક્ષણિકતાઓ વહેંચે છે. તે આપણને ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગમાં હાલના કોડને વિસ્તૃત કરવા (extend), પુનઃઉપયોગ કરવા (reuse) અને સુધારવા (improve) ની મંજૂરી આપે છે. આકૃતિ 11.7માં બતાવ્યા પ્રમાણે વર્તુળ/સર્કલ (Circle) અને નળાકાર/સિલિન્ડર (Cylinder) ક્લાસનું નીચેનું ઉદાહરણ ધ્યાનમાં લો.



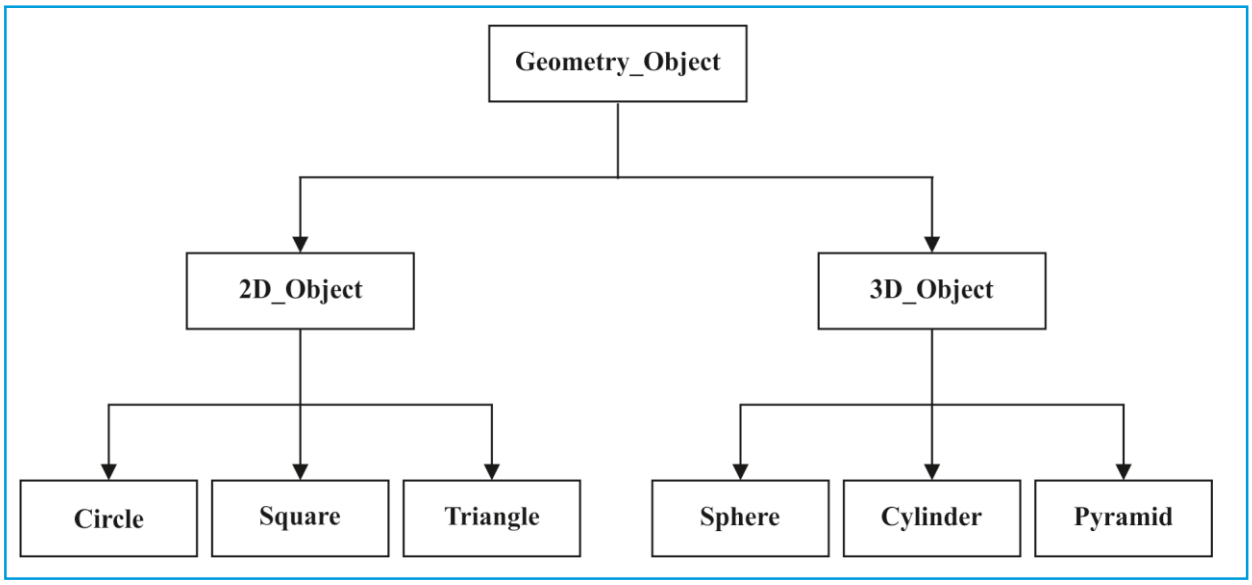
આકૃતિ 11.7 : ઈન્હેરીટન્સ દ્વારા વિસ્તૃતીકરણ અને પુનઃઉપયોગ

Circle ક્લાસ Radius પ્રોપર્ટી અને area() ઑપરેશન સાથે વ્યાખ્યાયિત થયેલ છે. તેને સિલિન્ડર ક્લાસ (Cylinder class) સુધી વિસ્તૃત કરવામાં આવે છે, જેમાં પ્રોપર્ટી તરીકે Radius (જે સર્કલ ક્લાસમાંથી વારસામાં મળે છે) અને Height ઉમેરવામાં આવે છે. આ એક વિસ્તરણ (extension)નું ઉદાહરણ છે, કારણ કે સિલિન્ડર ક્લાસ સર્કલમાં એક વધુ પ્રોપર્ટી ઉમેરીને તેને વિસ્તૃત કરે છે. area() ઑપરેશન સર્કલ ક્લાસમાંથી વારસામાં મળે છે, જેનો ઉપયોગ સિલિન્ડર ક્લાસમાં ઉમેરવામાં આવેલા volume() ઑપરેશનને વ્યાખ્યાયિત કરવા માટે થઈ શકે છે, કારણ કે સિલિન્ડરનું ઘનફળ $\pi r^2 h$ છે, જ્યાં πr^2 ક્ષેત્રફળ દર્શાવે છે. સર્કલ પહેલેથી જ ક્ષેત્રફળ ઑપરેશનને વ્યાખ્યાયિત કરે છે, જેનો ઘનફળની ગણતરી કરવા માટે પુનઃઉપયોગ કરવામાં આવ્યો છે.

ઈન્હેરીટન્સ એ વાસ્તવિક જીવનમાં વ્યાપકપણે ઉપયોગમાં લેવાતો ખ્યાલ છે, જ્યાં આપણે એક મેઈન ક્લાસને સબક્લાસમાં વિભાજિત કરીએ છીએ. બધા સબક્લાસ કેટલીક સામાન્ય લાક્ષણિકતાઓ (પ્રોપર્ટી અને ઑપરેશન્સ) વહેંચે છે, જ્યારે તેમની પોતાની અનન્ય લાક્ષણિકતાઓ પણ ધરાવે છે. સામાન્ય લાક્ષણિકતાઓ ટોચના ક્લાસ (top class)માં વ્યાખ્યાયિત કરવામાં આવે છે, જે બધા સબક્લાસ દ્વારા વહેંચાય છે. સબક્લાસને સમાન રીતે તેમના પણ સબક્લાસમાં આગળ વિભાજિત કરી શકાય છે અને તે આગલા સ્તર માટે બેઝ ક્લાસ

Get More Learning Materials Here : CLICK HERE www.studentbro.in

તરીકે કાર્ય કરે છે. આકૃતિ 11.8 ભૌમિતિક ઑબ્જેક્ટ (geomatry objects)નું ઉદાહરણ દર્શાવે છે, જેને 2D ઑબ્જેક્ટ અને 3D ઑબ્જેક્ટમાં વિભાજિત કરવામાં આવે છે. 2D ઑબ્જેક્ટને આગળ વર્તુળ (Circle), ચોરસ (Square) અને ત્રિકોણ (Triangle) જેવા આકારોમાં વિભાજિત કરવામાં આવે છે, અને 3D ઑબ્જેક્ટને ગોળો (Sphere), નળાકાર (Cylinder) અને પિરામિડ (Pyramid)માં વિભાજિત કરવામાં આવે છે.



આકૃતિ 11.8 : ભૌમિતિક ઑબ્જેક્ટનો વંશવેલો

આકૃતિમાં દર્શાવ્યા મુજબ, Geometry_Object ક્લાસમાં એવી વિશેષતાઓ સામેલ છે જે 2D_Object અને 3D_Object બંને ક્લાસ દ્વારા વહેંચાયેલી છે. 2D_Object ક્લાસ દ્વારા વ્યાખ્યાયિત વિશેષતાઓ Circle, Square અને Triangle માટે સામાન્ય છે, અને તેમની પોતાની વિશેષતાઓ પણ છે.

ઑબ્જેક્ટ વચ્ચેના વિવિધ સંબંધોને ધ્યાનમાં લેતા, આપણે પાંચ જુદા જુદા પ્રકારના ઇન્હેરીટન્સને વ્યાખ્યાયિત કરી શકીએ છીએ:

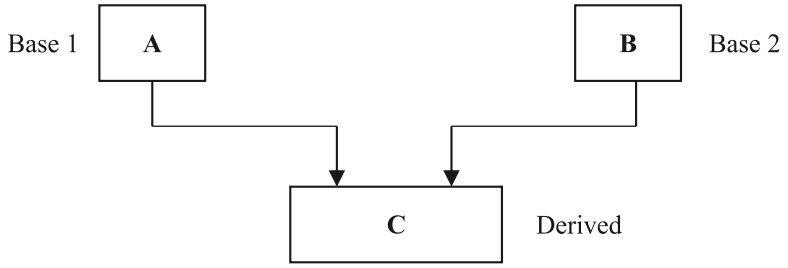
- સિંગલ ઇન્હેરીટન્સ (Single Inheritance)
- મલ્ટીલેવલ ઇન્હેરીટન્સ (Multilevel Inheritance)
- મલ્ટીપલ ઇન્હેરીટન્સ (Multiple Inheritance)
- હાઇરાર્કીકલ ઇન્હેરીટન્સ (Hierarchical Inheritance)
- હાઇબ્રિડ ઇન્હેરીટન્સ (Hybrid Inheritance)

આકૃતિ 11.9 આ પાંચેય પ્રકારના વારસાગત સંબંધો દર્શાવે છે. જ્યારે એક ક્લાસ બીજા ક્લાસમાંથી તારવવામાં (derived) આવે છે, ત્યારે તેને સિંગલ ઇન્હેરીટન્સ કહેવામાં આવે છે. તે પિતા અને બાળક (parent-child) વચ્ચેના સંબંધને રજૂ કરે છે. મલ્ટીલેવલ ઇન્હેરીટન્સમાં એક ક્લાસ એવા અન્ય ક્લાસમાંથી ઉત્પન્ન થાય છે જે પોતે પણ મૂળભૂત ક્લાસ (base class) માંથી ઉત્પન્ન થયો હોય. તે દાદા, પિતા અને બાળક (grandfather-parent-child) વચ્ચેના સંબંધને રજૂ કરે છે. મલ્ટીપલ ઇન્હેરીટન્સ બે કે તેથી વધુ મૂળભૂત ક્લાસનો ઉપયોગ કરીને એક નવો ક્લાસ ઉત્પન્ન કરે છે, જેમાં બંનેની વિશેષતાઓનું સંયોજન હોય છે. ઉદાહરણ તરીકે, આપણે કાર (car) અને વિમાન (plane) ની વિશેષતાઓને જોડીને એર-ટેક્સી (air_taxi) બનાવી શકીએ છીએ. આપણે ઘણા કુદરતી પદાનુક્રમો તેમજ સંસ્થાઓ અને વ્યવસાયોમાં પદાનુક્રમોનો ઉપયોગ કરીએ છીએ. તેને હાઇરાર્કીકલ સંબંધનો ઉપયોગ કરીને રજૂ કરવામાં આવે છે. જ્યારે કોઈ સંબંધ રજૂ કરવા માટે એક કરતા વધુ પ્રકારના વારસાગત સંબંધનો ઉપયોગ કરવામાં આવે છે, ત્યારે તેને હાઇબ્રિડ વારસાગત સંબંધ કહેવામાં આવે છે.

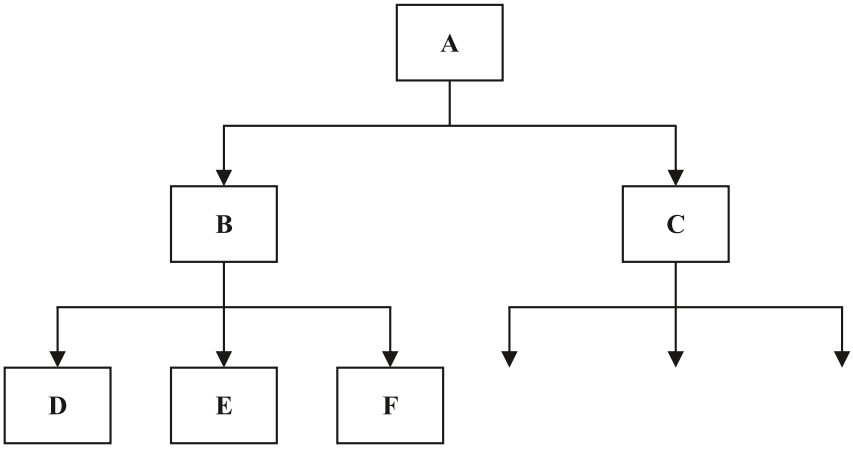
Get More Learning Materials Here : CLICK HERE www.studentbro.in

(a) Single Inheritance

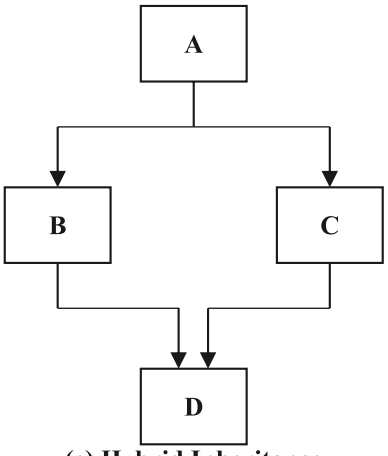
(b) Multilevel Inheritance



(c) Multiple Inheritance



(d) Hierarchical Inheritance



(e) Hybrid Inheritance

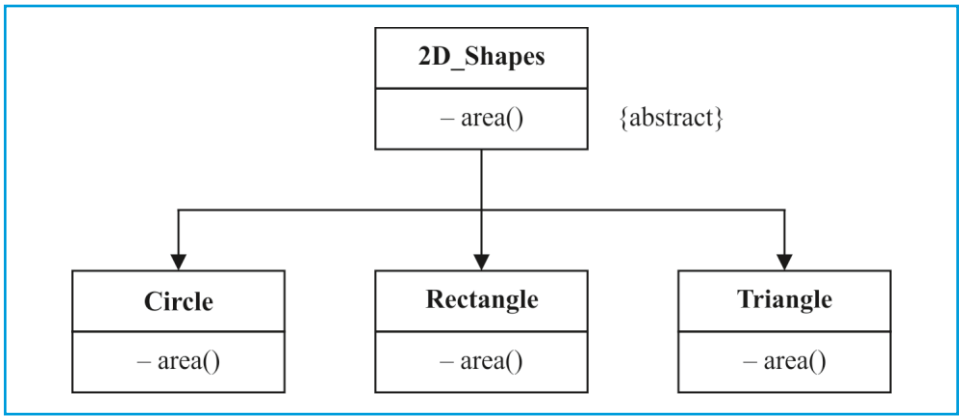
આકૃતિ 11.9 : ઇન્હેરીટન્સના પ્રકાર

Get More Learning Materials Here : CLICK HERE www.studentbro.in

પૉલિમૉર્ફિઝમ (Polymorphism)

પૉલિમૉર્ફિઝમનો અર્થ છે અનેક સ્વરૂપો (many forms). તેને એક જ વસ્તુ દ્વારા બહુવિધ વર્તન (multiple behaviour) તરીકે પણ વર્ણવી શકાય છે. આપણે C પ્રોગ્રામિંગમાં + ગણિતીય ઑપરેટરનો ઉપયોગ integer, float, double વગેરે જેવા વિવિધ ડેટા પ્રકાર સાથે કર્યો છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ આ + ઑપરેટરને યુઝર દ્વારા વ્યાખ્યાયિત ઑબ્જેક્ટ સાથે પણ કાર્ય કરવા માટે વિસ્તારે છે. આને ઑપરેટર ઓવરલોડિંગ (operator overloading) કહેવામાં આવે છે, કારણ કે ઑપરેટરને યુઝર દ્વારા વ્યાખ્યાયિત ઑબ્જેક્ટ સાથે કામ કરવા સક્ષમ બનાવવામાં આવે છે. તેનો અર્થ એ છે કે, આ જ ઑપરેટર જુદા જુદા ઑબ્જેક્ટ માટે અલગ રીતે વર્તે છે, કારણકે સરવાળાની પ્રક્રિયા દરેક ઑબ્જેક્ટ પ્રમાણે બદલાય છે. આ રીતે, ઑપરેટર + ના ઘણા સ્વરૂપો છે. આથી, ઑપરેટર ઓવરલોડિંગ એ પૉલિમૉર્ફિઝમનું એક ઉદાહરણ છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં બીજું ઉદાહરણ ફંક્શન ઓવરલોડિંગ (function overloading) છે, જે સમાન નામવાળા પરંતુ જુદા જુદા કાર્યો કરતા બે અથવા વધુ ફંક્શનને મંજૂરી આપે છે. ઑપરેટર ઓવરલોડિંગ અને ફંક્શન ઓવરલોડિંગ બંને કમ્પાઇલ ટાઇમ પૉલિમૉર્ફિઝમ (compile time polymorphism)ના ઉદાહરણો છે, કારણ કે ઘણા બિહેવિયરમાંથી કોઈ એકને પસંદ કરવાનો અંતિમ નિર્ણય પ્રોગ્રામનું કમ્પાઇલિંગ કરતી વખતે એટલે કે કમ્પાઇલ ટાઇમ પર લેવામાં આવે છે.

ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં ઇન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ બંને સાથે મળીને એક મહત્વપૂર્ણ ભૂમિકા ભજવે છે. આકૃતિ 11.10માં આપેલ વારસાગત સંબંધના પદાનુક્રમને ધ્યાનમાં લો, જે 2D_Shapes નામના બેઝ ક્લાસને તેના સબક્લાસ Circle, Rectangle અને Triangle સાથે રજૂ કરવામાં આવે છે.



આકૃતિ 11.10 : રન ટાઇમ પૉલિમૉર્ફિઝમ

આપણે જાણીએ છીએ કે દરેક આકારનું ક્ષેત્રફળ હોય છે, જે વિવિધ આકારો માટે અલગ અલગ રીતે ગણવામાં આવે છે. બધા આકારો આ સામાન્ય ઑપરેશનને વહેંચે છે, તેથી તેનો ઉલ્લેખ 2D_Shapes નામના બેઝ ક્લાસમાં કરવામાં આવ્યો છે. જોકે, તેની વ્યાખ્યા દરેક ઑબ્જેક્ટ માટે અલગ હોય છે. આ કારણોસર, area() ઑપરેશન બેઝ ક્લાસમાં એબ્સ્ટ્રેક્ટ (abstract) છે; એટલે કે, માત્ર તેનું ઇન્ટરફેસ (interface) પૂરું પાડવામાં આવે છે, પરંતુ તે દરેક સબક્લાસ દ્વારા તેની પોતાની વ્યાખ્યા સાથે અમલમાં મૂકવામાં આવે છે. ઉદાહરણ તરીકે, Circle ક્લાસ πr^2 નો ઉપયોગ કરીને *area()*નો અમલ કરે છે, Rectangle ક્લાસ length*breadth નો ઉપયોગ કરીને અમલ કરે છે અને Triangle ક્લાસ (height*base)/2 નો ઉપયોગ કરીને અમલ કરે છે. આ એક પ્રશ્ન ઊભો કરે છે કે જ્યારે આપણે 2D_shape ઑબ્જેક્ટ પર ક્ષેત્રફળની ગણતરી કરવા માંગીએ છીએ, ત્યારે ત્રણ અમલીકરણોમાંથી શેનો ઉપયોગ કરવો? જ્યારે પ્રોગ્રામ રન કરવામાં આવે છે ત્યારે કે આ આકાર વર્તુળ (circle), લંબચોરસ (rectangle) કે ત્રિકોણ (triangle)માંથી કયો છે એ માહિતી ઉપલબ્ધ થશે. આથી,

Get More Learning Materials Here : CLICK HERE www.studentbro.in

આને રન-ટાઇમ પૉલિમૉર્ફિઝમ (run-time polymorphism) તરીકે ઓળખવામાં આવે છે. આ એક ખૂબ જ શક્તિશાળી પદ્ધતિ છે જે ડાયનેમિઝમ (dynamism) અને ફ્લેક્સીબિલિટી (flexibility) પ્રદાન કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ પ્રોગ્રામિંગના લાભ (Benefits of Object Oriented Programming)

ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ (OOP) જાળવણીમાં સરળતા અને ખર્ચ-અસરકારકતા (cost-effectiveness) સાથે ગુણવત્તાયુક્ત સોફ્ટવેર બનાવવા માટે ઘણા ફાયદાઓ પ્રદાન કરે છે. તેના મુખ્ય લાભ નીચે મુજબ છે:

- તે વાસ્તવિક દુનિયાની વસ્તુઓનું મૉડલ બનાવે છે, જે સિસ્ટમમાં સ્પષ્ટ ડિઝાઇન અને સીમાઓને કારણે જટિલ સિસ્ટમને કુદરતી રીતે અને સરળતાથી સમજવાની મંજૂરી આપે છે.
- તે સંબંધિત ડેટા અને ક્રિયાઓને ક્લાસ નામના એક યુનિટમાં એકસાથે જોડે છે, જેનાથી કોડ વાંચવામાં અને તેનું સંચાલન કરવામાં સરળતા રહે છે.
- એક જ ક્લાસનો ઉપયોગ અન્ય પ્રોગ્રામમાં અથવા તે જ પ્રોગ્રામના જુદા જુદા ભાગમાં થઈ શકતો હોવાથી પુનઃઉપયોગિતા વધે છે.
- એક ક્લાસને પ્રોગ્રામના અન્ય ભાગોને અસર કર્યા વિના અપડેટ કરી અથવા બદલી શકાય છે, જેનાથી ફેરફારો સરળ બને છે.
- એન્કેપ્સ્યુલેશન ડેટાને સુરક્ષિત બનાવે છે અને ડેટા સંરક્ષણમાં વધારો કરીને માત્ર સુરક્ષિત રીતે ઉપયોગ કરવાની મંજૂરી આપે છે.
- ઈન્હેરીટન્સ વાસ્તવિક દુનિયાના સંબંધોને સરળતાથી રજૂ કરે છે અને કોડનું પુનરાવર્તન ટાળે છે, કારણ કે તે એક ક્લાસને બીજા ક્લાસની વિશેષતાઓનો ઉપયોગ કરવાની મંજૂરી આપે છે.
- પૉલિમૉર્ફિઝમ જુદા જુદા ઑબ્જેક્ટ્સ માટે સમાન ફંક્શનને અલગ રીતે અમલમાં મૂકવાની મંજૂરી આપે છે, જે કોડને વધુ ડાયનેમિક અને શક્તિશાળી બનાવે છે.
- તે વધુ ક્લાસ ઉમેરીને અથવા વારસાગત સંબંધોનો ઉપયોગ કરીને હાલના ક્લાસનું વિસ્તરણ કરીને સ્કેલેબિલિટીને સમર્થન આપે છે.
- ટીમ સરળતાથી સહયોગ કરી શકે છે, કારણ કે વિવિધ સભ્યો કોઈપણ મૂંઝવણ વિના જુદા જુદા ક્લાસ વિકસાવી શકે છે.
- તમામ આધુનિક ભાષાઓ OOPને સમર્થન આપે છે, જે ડેવલપર માટે વધુ વિકલ્પો પ્રદાન કરે છે.

ઑબ્જેક્ટ ઓરિએન્ટેડ ભાષાઓ (Object Oriented Languages)

ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ એ એવી પ્રોગ્રામિંગ ભાષાઓ છે જે સમગ્ર પ્રકરણમાં ચર્ચા કરાયેલા ઑબ્જેક્ટ-ઑરિએન્ટેડ ખ્યાલોને સમર્થન આપે છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ યુઝરને વાસ્તવિક દુનિયાની સિસ્ટમનું મૉડલ બનાવવા માટે ઈન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ સાથે ઑબ્જેક્ટ અને ક્લાસના સંદર્ભમાં મોટી અને જટિલ એપ્લિકેશનનું આયોજન કરવાની મંજૂરી આપે છે. તેના કારણે વાસ્તવિક દુનિયાની ઘટનાઓ સાથેની તેની સમાનતાને કારણે તેની સમજણ અને જાળવણી સરળ બને છે. કોઈપણ ભાષાને ઑબ્જેક્ટ-ઑરિએન્ટેડ તરીકે લાયક ઠરવા માટે, તેણે એન્કેપ્સ્યુલેશન, ઈન્હેરીટન્સ અને પૉલિમૉર્ફિઝમને સમર્થન આપવું આવશ્યક છે. ચાલો, લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ વિષે જાણકારી મેળવીએ.

Get More Learning Materials Here : CLICK HERE www.studentbro.in

C++

C++ એ ખૂબ જ જૂની અને લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા છે, જેની રચના 1980ના દાયકાની શરૂઆતમાં યુએસએ (USA) માં આવેલી AT&T બેલ લેબોરેટરીઝમાં બજાર્ને સ્ટ્રોસ્ટ્રૂપ (Bjarne Stroustrup) દ્વારા કરવામાં આવી હતી. Simula67 એ પ્રથમ ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા હતી, જે ઓલે-જોહાન ડાહલ (Ole-Johan Dahl) અને ક્રિસ્ટન નાયગાર્ડ (Kristen Nygaard) દ્વારા નોર્વેમાં ટ્રાફિક અથવા ફેક્ટરી પ્રક્રિયાઓ જેવી વાસ્તવિક દુનિયાની સિસ્ટમનું અનુકરણ કરવા માટે વિકસાવવામાં આવી હતી. તે જ સમયે, C તેની સરળતા, ફ્લેક્સીબિલિટી અને કાર્યક્ષમતાને કારણે સિસ્ટમ પ્રોગ્રામિંગમાં વ્યાપકપણે ઉપયોગમાં લેવાતી સૌથી લોકપ્રિય ભાષા હતી. સ્ટ્રોસ્ટ્રૂપ Simula67 અને C ભાષા બંનેની વિશેષતાઓને જોડવા માંગતા હતા. તેમણે C ભાષાનું વિસ્તરણ Simula67 માંથી લેવાયેલા ક્લાસ સાથે કર્યું અને તેનું નામ 'C with classes' રાખ્યું. પાછળથી તેનું નામ બદલીને C++ રાખવામાં આવ્યું, જેમાં ઈક્રિમેન્ટ ઑપરેટર ++ નો વિચાર કરીને સૂચવવામાં આવ્યું કે C++ એ C + 1 છે, એટલે કે તે પ્રોસિજરલ અને ઑબ્જેક્ટ-ઑરિએન્ટેડ બંને અભિગમોને સમર્થન પૂરાં પાડે છે. C++ એ ગેમ્સ, ઑપરેટિંગ સિસ્ટમ અને ટૂલ્સ વિકસાવવા માટે વિદ્યાર્થીઓ, એન્જિનિયરો અને ઉદ્યોગમાં ખૂબ જ લોકપ્રિય રહી છે.

C++ ને C ભાષાની લોકપ્રિયતા અને વ્યાપક ઉપયોગનો ફાયદો મળ્યો છે, કારણ કે C++ OOP વિશેષતાઓ સાથે C સિન્ટેક્સને અનુસરે છે, જે શીખવાની પ્રક્રિયાને ઝડપી બનાવે છે. આજે, સિન્ટેક્સની દ્રષ્ટિએ મોટા ભાગની આધુનિક ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ C ભાષામાંથી ઉતરી આવેલી છે, અને તે C++ જેવી જ ઑબ્જેક્ટ-ઑરિએન્ટેડ સુવિધાઓને સમર્થન આપે છે. આ જ કારણ છે કે આજે પણ C++ ઑબ્જેક્ટ-ઑરિએન્ટેડ ખ્યાલ અને પ્રોગ્રામિંગ શીખવા માટે પ્રથમ પસંદગી રહી છે.

જાવા (Java)

જાવા એ અન્ય એક લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષા છે, જે 1990ના દાયકાના મધ્યમાં સન માઈક્રોસિસ્ટમ્સ (Sun Microsystems)માં જેમ્સ ગોસલિંગ (James Gosling) અને તેમની ટીમ દ્વારા વિકસાવવામાં આવી હતી. જાવા સંપૂર્ણપણે ઑબ્જેક્ટ-ઑરિએન્ટેડ છે અને દરેક વસ્તુ ઑબ્જેક્ટની આસપાસ રહે છે. જાવાની સૌથી આકર્ષક વિશેષતા છે ''કોડ એકવાર લખો અને ગમે ત્યાં ચલાવો'' (write code once and run it anywhere). એટલે કે, એક સિસ્ટમ પર લખેલો કોડ ફરીથી કમ્પાઈલ કર્યા વિના કે તેમાં ફેરફાર કર્યા વિના કોઈપણ અન્ય સિસ્ટમ પર વાપરી શકાય છે. તેને પ્લેટફોર્મ-સ્વતંત્ર (platform independent) પણ કહેવામાં આવે છે, કારણ કે તે કોઈપણ હાર્ડવેર અને ઑપરેટિંગ સિસ્ટમ સંયોજન પર ફેરફાર વિના કામ કરે છે. જાવાની મુખ્ય વિશેષતાઓમાં ઑબ્જેક્ટ-ઑરિએન્ટેડ, પ્લેટફોર્મ-સ્વતંત્ર, સુરક્ષિત (secure), મજબૂત (robust), ડાયનેમિક, વિતરિત (distributed) અને મલ્ટિથ્રેડેડ (multithreaded)નો સમાવેશ થાય છે. જાવાનો સૌથી મોટો ફાયદો એ છે કે તે નાના અને સરળથી લઈને મોટા અને જટિલ વિવિધ પ્રકારના એપ્લિકેશન્સ માટે વિવિધ લાઈબ્રેરીઓના સંદર્ભમાં મોટો સપોર્ટ પૂરો પાડે છે. આનાથી વિકાસ ઝડપી બને છે, કારણ કે ડેવલપરે ફક્ત એપ્લિકેશનના મુખ્ય તર્ક પર જ ધ્યાન કેન્દ્રિત કરવું પડે છે. તે સ્ટાન્ડર્ડ, એન્ટરપ્રાઈઝ અને મોબાઈલ એપ્લિકેશન ડેવલપમેન્ટને સપોર્ટ કરે છે, જે તેને ઉદ્યોગ માટે આકર્ષક બનાવે છે.

પાયથોન (Python)

પાયથોન (Python) ખૂબ જ સરળતાથી શીખી શકાય તેવી આધુનિક પ્રોગ્રામિંગ ભાષા છે, જે ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગને સંપૂર્ણપણે સમર્થન આપે છે. તે 1990ના દાયકાની શરૂઆતમાં એમ્સ્ટરડેમમાં ગાઈડો વેન રોસમ (Guido van Rossum) દ્વારા એક સરળ ધ્યેય સાથે વિકસાવવામાં આવી હતી: પ્રોગ્રામિંગને વધુ સરળ અને વધુ વાંચનક્ષમ બનાવવું. તે અંગ્રેજી જેવી લાગે છે, જે તેને શાળાના વિદ્યાર્થીઓ માટે પણ યોગ્ય બનાવે છે. પાયથોન ડેટા સાયન્સ (Data Science) અને આર્ટિફિશિયલ ઈન્ટેલિજન્સ/મશીન લર્નિંગ (AI/

Get More Learning Materials Here : CLICK HERE www.studentbro.in

Machine Learning) માટે સર્વસંમત પ્રોગ્રામિંગ ભાષા તરીકે સ્વીકારવામાં આવી છે, કારણ કે તેને તેના માટે સમૃદ્ધ લાઇબ્રેરીઓનું સમર્થન મળે છે. તેમાંની કેટલીક નીચે આપવામાં આવી છે:

- NumPy : સંખ્યાઓ, એરે (arrays) સાથે કામ કરવા અને ગણિતની પ્રક્રિયા ઝડપથી કરવા માટે.
- Pandas : ટેબલનું સંચાલન કરવા અને અવ્યવસ્થિત (messy) ડેટાને દૂર કરવા માટે.
- Matplotlib : ચાર્ટ્સ અને ગ્રાફ્સ એટલે કે વિઝ્યુલાઇઝેશન માટે.
- Scikit-learn : આગાહી અને વર્ગીકરણ જેવા મશીન લર્નિંગ મૉડલ્સ બનાવવા માટે.
- TensorFlow : ડીપ લર્નિંગ અને એડવાન્સ્ડ AI માટે.

આવા વ્યાપક લાઇબ્રેરી સપોર્ટને કારણે પાયથોનનો ઉપયોગ ફેસ રેકગ્નેશન, રેકોમેન્ડેશન સિસ્ટમ્સ, ટ્યુમર ડિટેક્શન જેવી હેલ્થકેર એપ્લિકેશન્સ સહિત તમામ પ્રકારના AI/ML એપ્લિકેશન ડેવલપમેન્ટ માટે થાય છે. જનરેટિવ AI ખૂબ જ શક્તિશાળી LLMs (Large Language Models), જેમ કે ChatGPT, Claude, Gemini, LLaMA વગેરેની ઉપલબ્ધતાને કારણે વધુને વધુ લોકપ્રિય બની રહ્યું છે. LLMs બનાવવા માટે પણ પાયથોન સૌથી યોગ્ય પ્રોગ્રામિંગ ભાષા છે.

ઉપર જણાવેલ લોકપ્રિય ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ ભાષાઓ ઉપરાંત, સૉફ્ટવેર ઉદ્યોગ નીચેની OOPને સમર્થન આપતી ભાષાઓનો પણ વ્યાપકપણે ઉપયોગ કરે છે:

જાવાસ્ક્રિપ્ટ (JavaScript) : વેબ ડેવલપમેન્ટ માટે વપરાય છે. તે ઑબ્જેક્ટ આધારિત છે, બ્રાઉઝરમાં ચાલે છે અને ડાયનેમિક કન્ટેન્ટ પૂરૂ પાડે છે.

C # (સી શાર્પ) : માઇક્રોસોફ્ટ દ્વારા વિન્ડોઝ એપ્લિકેશન ડેવલપમેન્ટ અને ટૂલ્સ માટે વિકસાવવામાં આવી છે.

પીએચપી (PHP) : વેબ ડેવલપમેન્ટ માટે વપરાય છે. સર્વર બાજુની સ્ક્રિપ્ટિંગ માટે સારી છે.

સારાંશ

પ્રોસિજરલ પ્રોગ્રામિંગ કોડને ક્રમબદ્ધ પગલાની સૂચનાઓ તરીકે ગોઠવે છે, ડેટા અને ફંક્શનને એકબીજા સાથે ભેળવી દે છે અને પ્રતિબંધો વિના ડેટાને ઍક્સેસ કરવાની મંજૂરી આપે છે. તે પ્રોગ્રામને મોટો અને જટિલ બન્યા પછી કોડનું સંચાલન, અપડેટ અને તેમાં રહેલી સમસ્યાઓના નિવારણ જેવા કાર્યો મુશ્કેલ બનાવે છે. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ કોડને વાસ્તવિક દુનિયાની વસ્તુઓની જેમ ઑબ્જેક્ટ તરીકે ગોઠવીને આ તમામ સમસ્યાઓનું નિરાકરણ લાવે છે. કોડને ઑબ્જેક્ટ્સના સંદર્ભમાં ગોઠવવાથી તે વાસ્તવિક-દુનિયાની સિસ્ટમ સાથે ગાઢ સામ્યતાને કારણે અને ખાસ કરીને સિસ્ટમ મોટી અને જટિલ હોય ત્યારે, તેનું સંચાલન અને સુધારા સરળ બને છે. ઑબ્જેક્ટમાં પ્રોપર્ટી (ડેટા) અને બિહેવીયર (ફંક્શન અથવા મેથડ) હોય છે, અને સમાન ડેટા અને મેથડ વહેંચતા ઑબ્જેક્ટનો સમૂહ ઑબ્જેક્ટ ક્લાસ બનાવે છે. ક્લાસ એ ઑબ્જેક્ટ માટેની એક બ્લુપ્રિન્ટ છે, અને દરેક ઑબ્જેક્ટ તે ક્લાસનો ઈન્સ્ટન્સ છે. ઑબ્જેક્ટ અને ક્લાસ ઉપરાંત, ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ એન્કેપ્સ્યુલેશન, ડેટા હાઇડિંગ અને એબ્સ્ટ્રેક્શન જેવા ખ્યાલોને સમર્થન આપે છે. એન્કેપ્સ્યુલેશન ડેટા અને મેથડને એકસાથે મૂકે છે અને આકસ્મિક ફેરફારોથી ડેટાનું રક્ષણ કરે છે. કમ્પોઝિશન અને એગ્રિગેશનનો ઉપયોગ અનુક્રમે પાર્ટ-ઓફ (part-of) અને હેઝ-એ (has-a) સંબંધોને દર્શાવવા માટે થાય છે. ઈન્હેરીટન્સ અને પૉલિમૉર્ફિઝમ ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં મહત્વની ભૂમિકા ભજવે છે. વારસાગત સંબંધો એક ક્લાસને બીજા ક્લાસની વિશેષતાઓનો ઉપયોગ કરવાની તેમજ ક્લાસને સબક્લાસમાં વિભાજીત કરવાની મંજૂરી આપે છે. આપણે પાંચ જુદા જુદા પ્રકારના વારસાગત સંબંધો જોયા : સિંગલ, મલ્ટીપલ, મલ્ટીલેવલ, હાઇરાર્કીકલ અને હાઇબ્રિડ. પૉલિમૉર્ફિઝમ એક જ મેથડના બહુવિધ સ્વરૂપોને મંજૂરી આપે છે, જે ડાઈનેમિઝમ અને ફ્લેક્સીબિલિટીને સમર્થન આપે છે. આ પ્રકરણની સમાપ્તિ લોકપ્રિય અને વ્યાપકપણે ઉપયોગમાં લેવાતી ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષાઓ C++, Java અને Python ની ચર્ચા સાથે કરવામાં આવી.

Get More Learning Materials Here : CLICK HERE www.studentbro.in

સ્વાધ્યાય

1. પ્રોસિજરલ પ્રોગ્રામિંગની મર્યાદાઓ શું છે?
2. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગનો બિલ્ડિંગ બ્લોક શું છે? ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામનું માળખું જણાવો.
3. ઑબ્જેક્ટ શું છે? તમે જાણતા હો તેવા ઓછામાં ઓછા 5 ઑબ્જેક્ટની સૂચિ બનાવો.
4. ઑબ્જેક્ટ શેનાથી બને છે? એક ઉદાહરણ આપો.
5. તમે ક્લાસ વિશે શું સમજો છો? તે ઑબ્જેક્ટથી કેવી રીતે અલગ છે?
6. નીચેના ઑબ્જેક્ટની પ્રોપર્ટી અને ઓપરેશન જણાવો : વિદ્યાર્થી, બસની ટિકિટ, વર્તુળ, સાયકલ.
7. ઈન્હેરીટન્સ શું છે? વારસાગત સંબંધોના વિવિધ પ્રકારોની સૂચિ બનાવો.
8. પૉલિમૉર્ફિઝમ શું છે? તેનો ઉપયોગ શું છે?
9. ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગમાં એન્કેપ્સ્યુલેશન શા માટે મહત્વનું છે?
10. એન્કેપ્સ્યુલેશન અને એબ્સ્ટ્રેક્શનનો અર્થ સમજાવો.
11. **સાચું કે ખોટું જણાવો.**
    (1) ઑબ્જેક્ટ એ ક્લાસનો એક ઈન્સ્ટન્સ છે.
    (2) એબ્સ્ટ્રેક્શન ડેટાને છુપાવીને સુરક્ષિત બનાવે છે.
    (3) એન્કેપ્સ્યુલેશન પ્રોપર્ટીને છુપાવે છે અને ઓપરેશનને દર્શાવે છે.
    (4) ઈન્હેરીટન્સ એક ક્લાસની પ્રોપર્ટી અને મેથડનો બીજા ક્લાસમાં ઉપયોગ કરવાની મંજૂરી આપે છે.
    (5) પક્ષી એક સંકલ્પનાત્મક ક્લાસ (conceptual class) છે.
12. **ખાલી જગ્યા પૂરો.**
    (1) _____ એક ક્લાસની લાક્ષણિકતાઓનો અન્ય ક્લાસમાં ઉપયોગ કરવાની મંજૂરી આપે છે.
    (2) ડેટા અને મેથડને એકજૂથ કરવાની પ્રક્રિયાને _____ કહે છે.
    (3) ઑબ્જેક્ટ બનાવવા માટેની બ્લ્યુ પ્રિન્ટને _____ કહે છે.
    (4) બિનજરૂરી વિગતોને છુપાવવાની પ્રક્રિયાને _____ કહે છે.
    (5) _____ દ્વારા એકથી વધુ ફંક્શનને એક સમાન નામ આપી શકાય છે.
13. **બહુ વિકલ્પી પ્રશ્નો. સૌથી યોગ્ય જવાબ પસંદ કરો.**
    (1) OOP (ઑબ્જેક્ટ-ઑરિએન્ટેડ પ્રોગ્રામિંગ) માં ઑબ્જેક્ટ શું છે?
        (a) ક્લાસ બનાવવા માટેની બ્લુપ્રિન્ટ
        (b) ડેટા ટાઈપ
        (c) વાસ્તવિક દુનિયાની એન્ટિટી
        (d) ટાસ્ક
    (2) OOP માં ક્લાસ શું છે?
        (a) ઑબ્જેક્ટ્સનો સમૂહ
        (b) ઑબ્જેક્ટ્સ બનાવવા માટેનું એક ટેમ્પલેટ
        (c) ખાસ પ્રકારનો ઑબ્જેક્ટ
        (d) મેથડનો સમૂહ
    (3) નીચેનામાંથી શેના દ્વારા બાહ્ય હસ્તક્ષેપ સામે ડેટા અને મેથડને સુરક્ષિત બનાવવામાં આવે છે?
        (a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

Get More Learning Materials Here : CLICK HERE www.studentbro.in

(4) નીચેનામાંથી કયું બિનજરૂરી વિગતો છુપાવે છે અને માત્ર જરૂરી વિગતો જ પ્રસ્તુત કરે છે?

(a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

(5) નીચેનામાંથી શેના દ્વારા એક ક્લાસની વિશેષતાઓનો ઉપયોગ બીજા ક્લાસમાં કરી શકાય છે?

(a) Inheritance (b) Polymorphism (c) Encapsulation (d) Abstraction

(6) ઑબ્જેક્ટ વિશે નીચેનામાંથી કયું વિધાન સાચું નથી?

(a) ઑબ્જેક્ટમાં એટ્રિબ્યુટ્સ અને મેથડનો સમાવેશ થાય છે

(b) ઑબ્જેક્ટ ક્લાસનું એક ઈન્સ્ટન્સ છે

(c) ઑબ્જેક્ટ એક વાસ્તવિક દુનિયાની એન્ટિટી છે

(d) ઑબ્જેક્ટ ક્લાસની બ્લુપ્રિન્ટ છે

(7) પૉલિમૉર્ફિઝમનો અર્થ શું છે?

(a) એક જ ફંક્શનના વિવિધ સ્વરૂપો (b) એક જ ક્લાસના વિવિધ ઈન્સ્ટન્સ

(c) બેઝ ક્લાસ માટેનો સબક્લાસ (d) ઑબ્જેક્ટનું એગ્રિગેશન

(8) નીચેનામાંથી કયો ઑબ્જેક્ટ નથી?

(a) લંબચોરસ (Rectangle) (b) ત્રિજ્યા (Radius)

(c) વર્તુળ (Circle) (d) ત્રિકોણ (Triangle)

(9) ઈનહેરીટન્સના કયા પ્રકારમાં બે કે તેથી વધુ ક્લાસનો બેઝ ક્લાસ તરીકે ઉપયોગ થાય છે?

(a) સિંગલ (b) મલ્ટીપલ (c) મલ્ટીલેવલ (d) હાઈરાર્કીકલ

(10) નીચેનામાંથી કઈ ઑબ્જેક્ટ-ઑરિએન્ટેડ ભાષા નથી?

(a) C (b) C++ (c) Java (d) Python

પ્રાયોગિક સ્વાધ્યાય

1. તમારી શાળાની લાઈબ્રેરીની મુલાકાત લો અને ત્યાં જોવા મળતા ઓછામાં ઓછા 5 ઑબ્જેક્ટની યાદી બનાવો.
2. લાઈબ્રેરીમાં પુસ્તક (Book) સૌથી મહત્વપૂર્ણ ઑબ્જેક્ટમાંથી એક છે. તેના એટ્રીબ્યુટ અને બિહેવીયરની યાદી બનાવો.
3. નીચેના ઑબ્જેક્ટને તેમની સામાન્ય વિશેષતાઓના આધારે વિવિધ ક્લાસમાં વર્ગીકૃત કરો:
   કાર (Car), કેળું (Banana), મોર (Peacock), સફરજન (Apple), સ્કૂટર (Scooter), ચકલી (Sparrow), દ્રાક્ષ (Grape), ટ્રક (Truck), કબૂતર (Dove), બસ (Bus)
4. શોપિંગ મૉલના વિવિધ વિભાગોમાં વેચાતી વસ્તુઓ માટે એક ઈનહેરીટન્સ હાઈરાર્કી તૈયાર કરો.
   ઉદાહરણ તરીકે, કાપડ (cloth) એક વસ્તુ છે જે આગળ પુરુષો, સ્ત્રીઓ અને બાળકોના કપડાંમાં વિભાજિત થાય છે. પુરુષોના કપડાં આગળ શર્ટ અને પેન્ટ વગેરેમાં વિભાજિત થઈ શકે છે.
5. વિવિધ પ્રકારના વાહનો (vehicles) માટે હાઈરાર્કી તૈયાર કરો.
   ટોચના સ્તરે એવા સામાન્ય કાર્યોને ઓળખો, જેને નીચલા સ્તરે અલગ અલગ અમલની જરૂર હોય.

Get More Learning Materials Here : CLICK HERE www.studentbro.in