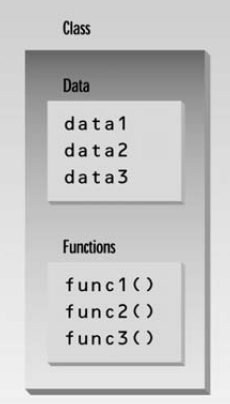
**OOP ဘယ္လဲ ဘာလဲ (၂)**

ယခင္ အပတ္တုန္းက OOP ကို ကၽြန္ေတာ္တို႔ ဘာေၾကာင္႔ သံုးရသလဲ ဆိုတာရယ္၊ ယခင္က အသံုးျပဳခဲ႔ၾကတဲ႔ Structural Programming မွာ ဘာေၾကာင္႔ အခက္အခဲေတြ ရွိခဲ႔လဲဆိုတာေတြကို ရွင္းျပခဲ႔ၿပီးပါၿပီ။ ဒီအပတ္မွာေတာ႔ OOP ရဲ႕ သေဘာသဘာဝအခ်ိဳ႕ကို အၾကမ္းဖ်င္း ဆက္လက္ ရွင္းျပလိုပါတယ္။

**Methods and Classes**

အထက္ပါ ကားဥပမာဟာ OOP အတြက္ အေရးႀကီးတဲ႔ concept အခ်ိဳ႕ကို မိတ္ဆက္ ေပးခဲ႔ပါတယ္။ ပရိုဂရမ္ထဲမွာ လုပ္ငန္းေတြ လုပ္ေဆာင္ႏိုင္ဖို႔ method ဒါမွ မဟုတ္ function ေတြ ေရးသားထားဖို႔ လိုပါတယ္။ အဲဒီ method ေတြ attribute ေတြကို စုစည္းၿပီး class တစ္ခုထဲမွာ ထည္႔သြင္း ေရးသားထားရမွာ ျဖစ္ပါတယ္။ class ေတြထဲက method ေတြဟာ user ကို အေသးစိတ္ အခ်က္အလက္ေတြကေန ဖံုးကြယ္ထားေပးမွာ ျဖစ္ပါတယ္။



**Attributes and Instance Variables**

ကားတစ္စီးမွာ အရွိန္ျမွင္႔တာ၊ ဘရိတ္အုပ္တာ၊ ေကြ႔တာ စတဲ႔ function ေတြအျပင္၊ အျမန္ႏႈန္း၊ ကားအေရာင္၊ အင္ဂ်င္ပါဝါ အစရွိတဲ႔ attributes ေတြလဲ ရွိပါတယ္။ကား ဒီဇိုင္းထဲမွာ အဲဒီ အခ်က္အလက္ေတြကို ေစာင္႔ၾကည္႔ဖို႔ odometer နဲ႔ fuel gauge ေတြ ထည္႔သြင္းထား သလိုပါပဲ။ class တစ္ခုထဲမွာ data variable ေတြ ထည္႔သြင္း ထားႏိုင္ပါတယ္။

ဥပမာ-

private String Model;

private int Color;

private float Power;

**Data hiding or Encapsulation**

Object ေတြကို ဖန္တီးလိုက္တဲ႔ အခါမွာ မွာ object တစ္ခုခ်င္းစီမွာ သီးသန္႔ ေဒတာေတြ ျဖစ္လာပါတယ္။ ဥပမာ ကားတစ္စီးက fuel gauge ဟာ သူ႔ထဲက ဆီလက္က်န္ ကို ျပေပးႏိုင္ေပမယ္႔ အျခား ကားရဲ႕ ဆီလက္က်န္ကို မသိႏိုင္ မျပႏိုင္သလိုပါပဲ။ Object တစ္ခုထဲမွာပါတဲ႔ function ကို member function ဒါမွမဟုတ္ method လို႔ ေခၚပါတယ္။ Object ထဲက attributes(datas) ေတြကို အဲဒီ member function ေတြကပဲ ရယူႏိုင္ပါတယ္။ တကယ္လို႔ Object တစ္ခုထဲက data တစ္ခုခုကို ရယူခ်င္တယ္ဆိုရင္ သက္ဆိုင္ရာ member function ကို ေခၚယူ အသံုးခ်ရမွာ ျဖစ္ပါတယ္။ အဲဒီ member function က လိုအပ္တဲ႔ data ကို object ထဲက ရယူၿပီး return ျပန္ေပးမွာ ျဖစ္ပါတယ္။

public void SetModel(String mdl)

{

Model = mdl;

}

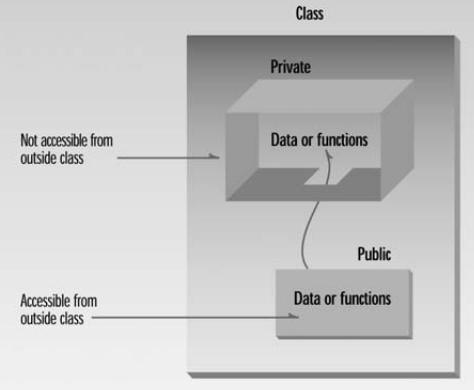
public String GetModel()

{

return Model;

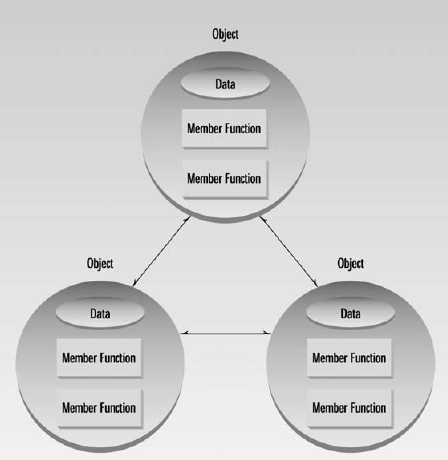
}

Object တစ္ခုထဲက ေဒတာ ေတြကို တိုက္ရိုက္ရယူဖို႔ မျဖစ္ႏိုင္ပါဘူး။ ေဒတာေတြကို တိုက္ရိုက္ ကိုင္တြယ္ျခင္းေၾကာင္႔ ျပႆနာေတြ ျဖစ္ခဲ႔တဲ႔ အေၾကာင္း၊ global variable ေတြရဲ႕ ျပႆနာေတြကို ရွင္းျပခဲ႔ၿပီးပါၿပီ။ ဒါေၾကာင္႔ OOP မွာေတာ႔ ေဒတာေတြကို သက္ဆိုင္ရာ Object ထဲမွာပဲ ဖုံးကြယ္ သိုဝွက္ထားျခင္း (hidden) နဲ႔ မေတာ္တဆ ျပဳျပင္ေျပာင္းလဲ ခံရျခင္းကေန ကာကြယ္ေပးထားပါတယ္။ အျပင္ေလာကမွာ မိမိ အရပ္အေမာင္းကို သူမ်ားက သတ္မွတ္ ဖန္တီးေပးလို႔ မရသလိုပါပဲ။ အဲဒီလို hidden လုပ္ဖို႔ ပရိုဂရမ္ အေတာ္မ်ားမ်ားမွာ private ဆိုတဲ႔ keyword နဲ႔ ဖန္တီးယူရပါတယ္။ တကယ္လို႔ variable တစ္ခု ေၾကျငာရာမွာ ဘာမွ ေျပာမထားဘူးဆိုရင္ default အေနနဲ႔ private လို႔ ယူဆေလ႔ ရွိပါတယ္။ အဲဒီလို ေဒတာေတြကို ဖြက္ထားျခင္းကို data encapsulation လို႔လဲ ေခၚဆိုၾကပါတယ္။



**Message**

OOP နဲ႔ ေရးသားထားတဲ႔ ပရိုဂရမ္ တစ္ခုမွာ object အမ်ားအျပား ပါဝင္ႏိုင္ပါတယ္။ အဲဒီ object အခ်င္းခ်င္း member function ေတြေခၚသံုးျခင္းအားျဖင္႔ ဆက္သြယ္လုပ္ကိုင္ေနၾကပါတယ္။



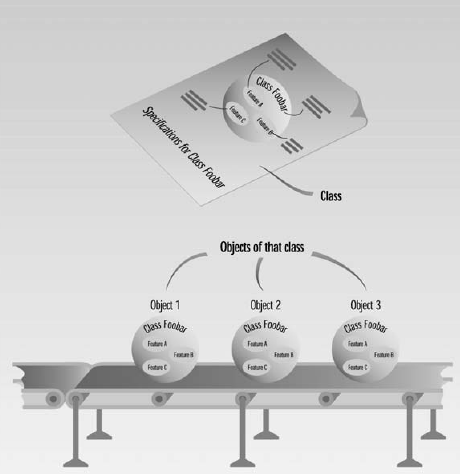
Object ရဲ႕ member function တစ္ခုကို ေခၚယူျခင္းကို အဲဒီ object ကို message တစ္ခု ေပးပို႔ျခင္းလို႔လည္း ေခၚဆိုႏိုင္ပါတယ္။ ဌာနတစ္ခုမွာဆိုရင္ manager တစ္ေယာက္က ဝင္ေငြ၊ ထြက္ေငြကို သိရဖို႔အတြက္ စာရင္းကိုင္ဌာနကို ကိုယ္တိုင္သြားၿပီး ဖိုင္ေတြ ရွာေဖြဖို႔ မလိုအပ္ပါဘူး။ သက္ဆိုင္ရာ ဌာနက ဆိုင္ရာ တာဝန္ခံေတြကိုသာ အခ်က္အလက္ေတြ ေတာင္းခံလိုက္ရံုပဲ ျဖစ္ပါတယ္။ ပရိုဂရမ္းမင္းမွာလဲ ဒီလိုပါပဲ။ ဒါေၾကာင္႔ ေဒတာေတြကို ျပင္ပက ျပင္ဆင္ ေႏွာက္ယွက္ျခင္း မခံရဘဲ တိတိက်က် ရယူႏိုင္ပါတယ္။

**Instantiation**

Class တစ္ခုကို define လုပ္ရံုနဲ႔ object ေတြ ဖန္တီးမေပးႏိုင္ပါဘူး။ တကယ္ေတာ႔ class define လုပ္ျခင္းဟာ blue print တစ္ခု (သို႔) ပံုစံခြက္ တစ္ခု ဖန္တီးျခင္းမွ်သာ ျဖစ္ၿပီး instantiate လုပ္ေပးမွသာ object မ်ားကို RAM အတြင္းမွာ ဖန္တီးေပးႏိုင္မွာ ျဖစ္ပါတယ္။ ဥပမာ-

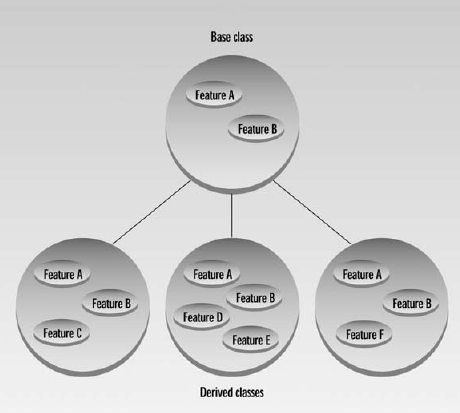
**Automobile Auto1 = new Automobile();**

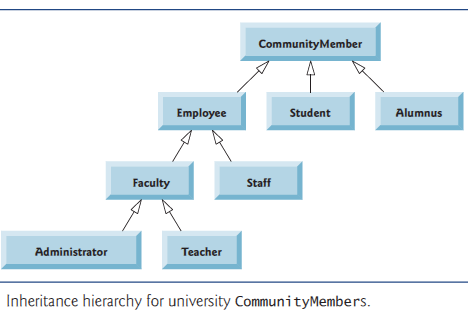
**Automobile Auto2 = new Automobile();**



**Inheritance**

ေန႔စဥ္ဘဝမွာ class တစ္ခုကေန ခြဲထြက္သြားတဲ႔ subclass မ်ားနဲ႔ ထိေတြ႔ေနရပါတယ္။ ဥပမာ- ေရာ႔ခ္၊ ေပါ႔ပ္၊ ဟစ္ေဟာ႔ စတာေတြဟာ Music class ထဲက subclass ေတြ ျဖစ္ပါတယ္။ အဲဒီ subclass ေတြဟာ derived class (ဒီဥပမာမွာေတာ႔ Music class) မွာ ရွိတဲ႔ member data နဲ႔ member function ေတြကို အေမြ ဆက္ခံထားသလို ကိုယ္ပိုင္ data နဲ႔ function ေတြ ထပ္မံေပါင္းထည္႔ ပါဝင္ႏိုင္ပါတယ္။ ဥပမာ - convertible ဆိုတဲ႔ ကားအမ်ိဳးအစားဟာ သာမန္ automobile ကားအမ်ိဳးအစားကိုပဲ အမိုးတင္ႏိုင္ခ်ႏိုင္တဲ႔ အရည္အေသြး ထပ္မံေပါင္းထည္႔ထားတာပဲ ျဖစ္ပါတယ္။



****

**Reusability**

class တစ္ခုကို ေရးသား ဖန္တီးၿပီးလို႔ debugged ျပဳလုပ္ၿပီးၿပီ ဆိုရင္ အျခား ပရိုဂရမ္မာေတြကို သူတို႔ေရးမယ္႔ ပရိုဂရမ္ေတြမွာ အသံုးျပဳႏိုင္ေအာင္ ျဖန္႔ျဖဴးေပးႏိုင္စြမ္း ရွိလာပါၿပီ။ အဲဒါကို reusability လုိ႔ ေခၚပါတယ္။ inheritance ကို သံုးခြင္႔ ရလို႔လည္း reusability ပိုျဖစ္လာပါတယ္။ ပရိုဂရမ္မာ တစ္ေယာက္အေနနဲ႔ ရွိၿပီးသား class တစ္ခုကို တိုက္ရိုက္ ျပင္ဆင္ေရးသားျဖင္း မျပဳဘဲ inheritance ကိုသံုးၿပီး features အသစ္ေတြ ထပ္ထည္႔ ေပးႏိုင္လာလို႔ပဲ ျဖစ္ပါတယ္။

**Creating New Data Types**

Object ရဲ႕ အားသာခ်က္တစ္ခုက data types အသစ္ေတြကို အလြယ္တကူ ဖန္တီးခြင္႔ေပးျခငး္ပဲ ျဖစ္ပါတယ္။

**Polymorphism and Over loading**

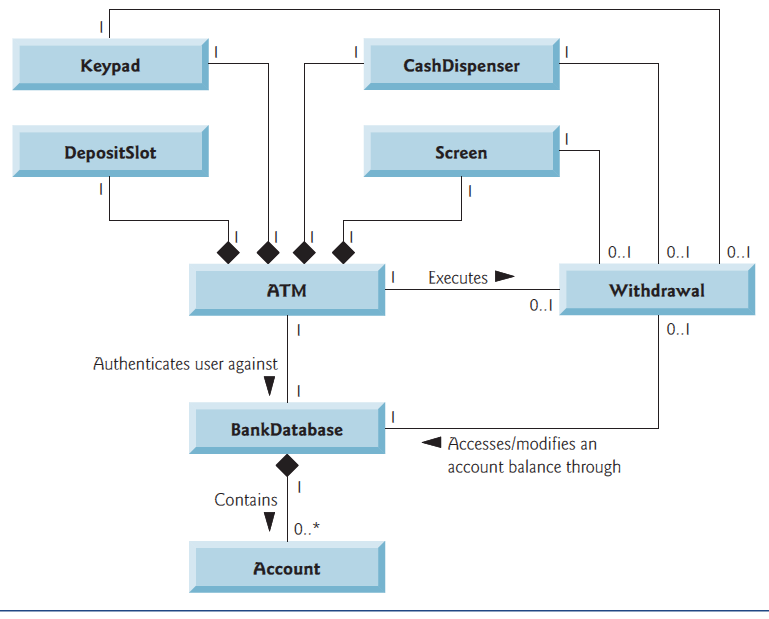
(+, -, =) စတဲ႔ operator ေတြကို bulid in data types ေတြနဲ႔ အသံုးျပဳဖုိ႔ ဖန္တီးေပးထားပါတယ္။ ကၽြန္ေတာ္တို႔ ဖန္တီးေပးမယ္႔ class ေတြမွာ အသံုးျပဳဖို႔ အတြက္ကေတာ႔ class ရဲ႕ member function အေနနဲ႔ operator overloading သံုးၿပီး ေရးေပးရမွာ ျဖစ္ပါတယ္။

**Object-Oriented Analysis and Design (OOAD)**

ပရိုဂရမ္တစ္ခု စေရးမယ္ဆိုရင္ အေတာ္မ်ားမ်ားက ကြန္ပ်ဴတာေရွ႕မွာ ထိုင္ၿပီး စာစရိုက္ေလ႔ ရွိပါတယ္။ တကယ္ေတာ႔ ႀကီးမားတဲ႔ ပရိုဂရမ္ေတြ ဥပမာ- ပရိုဂရမ္မာ တစ္ေထာင္ေလာက္ စုၿပီးေရးရတဲ႔ ေလေၾကာင္းထိန္းသိမ္းေရး software လိုမ်ိဳးမွာဆိုရင္ ဒီလို တန္းေရးဖို႔ မျဖစ္ႏိုင္ပါဘူး။ အေကာင္းဆံုး solution ကို ရဖို႔အတြက္ ကၽြန္ေတာ္တို႔ ေရးသားမယ္႔ ပေရာဂ်က္မွာ လိုအပ္ခ်က္ေတြကို အေသးစိတ္ စဥ္းစားသတ္မွတ္ ရမွာ ျဖစ္ပါတယ္။ ၿပီးရင္ အဲဒီ လိုအပ္ခ်က္ (requirements) ေတြကို ေျပလည္ေစမယ္႔ software ဒီဇိုင္းကို ေရးသားရမွာ ျဖစ္ပါတယ္။ OOP (point of view) အျမင္နဲ႔ အဲလို ေလ႔လာဆန္းစစ္တာကို (OOAD) process လို႔ ေခၚၾကပါတယ္။

**UML**

OOAD process ေတြကေန ထြက္လာတဲ႔ ရလာဒ္ေတြကို ဆက္သြယ္ေပးဖို႔အတြက္ graphical language တစ္ခုကေတာ႔ (Unified Modeling Language) UML ပဲ ျဖစ္ပါတယ္။ class ေတြနဲ႔ ဘယ္လို ဖြဲ႔စည္း တည္ေဆာက္ထားၿပီး ဆက္စပ္ခ်က္ေတြ ဘယ္လိုရွိတယ္ဆိုတာကို အေပၚစီးကေန အလြယ္တကူ ၾကည္႔ရႈလို႔ရေအာင္ အသံုးျပဳတာ ျဖစ္ပါတယ္။ ဘယ္ေလာက္အထိ progress ၿပီးစီးၿပီဆိုတာကိုလဲ သိႏိုင္ပါတယ္။



**အကိုးအကားမ်ား**

၁။ Java : how to program / P.J.Deitel, H.M. Deitel. --9th ed. 2012

၂။ Object-Oriented Programmming in C++ / Robert Lafore. --4th ed.2002

ေဒါက္တာ ေအာင္ဝင္းထြဋ္

(bluephoenix)

http://tech4mm.com