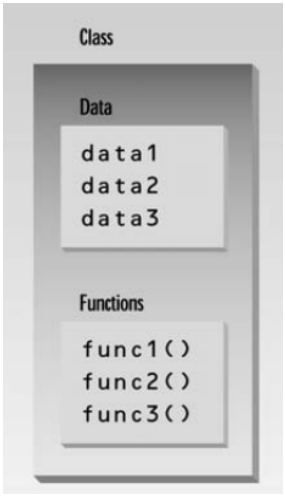
OOP ဘယ်လဲ ဘာလဲ (၂)

ယခင် အပတ်တုန်းက OOP ကို ကျွန်တော်တို့ ဘာကြောင့် သုံးရသလဲ ဆိုတာရယ်၊ ယခင်က အသုံးပြုခဲ့ကြတဲ့ Structural Programming မှာ ဘာကြောင့် အခက်အခဲတွေ ရှိခဲ့လဲဆိုတာတွေကို ရှင်းပြခဲ့ပြီးပါပြီ။ ဒီအပတ်မှာတော့ OOP ရဲ့ သဘောသဘာဝအချို့ကို အကြမ်းဖျင်း ဆက်လက် ရှင်းပြလိုပါတယ်။

Methods and Classes

အထက်ပါ ကားဥပမာဟာ OOP အတွက် အရေးကြီးတဲ့ concept အချို့ကို မိတ်ဆက် ပေးခဲ့ပါတယ်။ ပရိုဂရမ်ထဲမှာ လုပ်ငန်းတွေ လုပ်ဆောင်နိုင်ဖို့ method ဒါမှ မဟုတ် function တွေ ရေးသားထားဖို့ လိုပါတယ်။ အဲဒီ method တွေ attribute တွေကို စုစည်းပြီး class တစ်ခုထဲမှာ ထည့်သွင်း ရေးသားထားရမှာ ဖြစ်ပါတယ်။ class တွေထဲက method တွေဟာ user ကို အသေးစိတ် အချက်အလက်တွေကနေ ဖုံးကွယ်ထားပေးမှာ ဖြစ်ပါတယ်။



Attributes and Instance Variables

ကားတစ်စီးမှာ အရှိန်မြှင့်တာ၊ ဘရိတ်အုပ်တာ၊ ကွေ့တာ စတဲ့ function တွေအပြင်၊ အမြန်နှုန်း၊ ကားအရောင်၊ အင်ဂျင်ပါဝါ အစရှိတဲ့ attributes တွေလဲ ရှိပါတယ်။ကား ဒီဇိုင်းထဲမှာ အဲဒီ အချက်အလက်တွေကို စောင့်ကြည့်ဖို့ odometer နဲ့ fuel gauge တွေ ထည့်သွင်းထား သလိုပါပဲ။ class တစ်ခုထဲမှာ data variable တွေ ထည့်သွင်း ထားနိုင်ပါတယ်။

ဥပမာ-

```
private String Model;

private int Color;

private float Power;
```

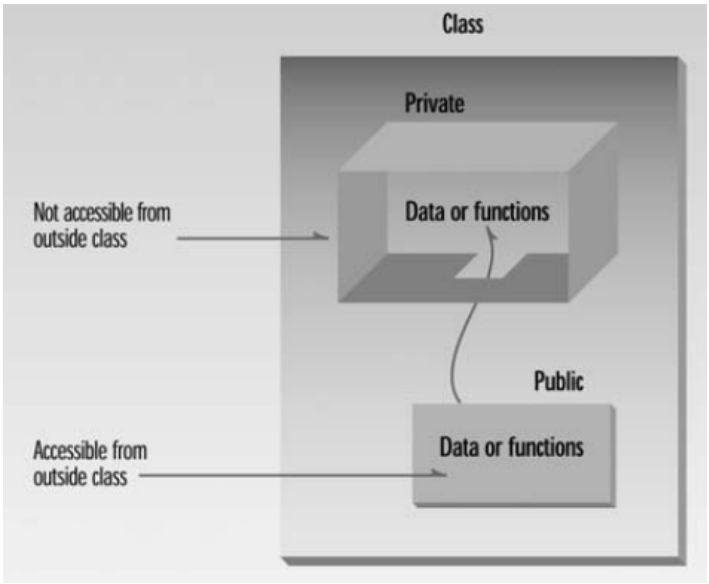
Data hiding or Encapsulation

Object တွေကို ဖန်တီးလိုက်တဲ့ အခါမှာ မှာ object တစ်ခုချင်းစီမှာ သီးသန့် ဒေတာတွေ ဖြစ်လာပါတယ်။ ဥပမာ ကားတစ်စီးက fuel gauge ဟာ သူ့ထဲက ဆီလက်ကျန် ကို ပြပေးနိုင်ပေမယ့် အခြား ကားရဲ့ ဆီလက်ကျန်ကို မသိနိုင် မပြနိုင်သလိုပါပဲ။ Object တစ်ခုထဲမှာပါတဲ့ function ကို member function ဒါမှမဟုတ် method လို့ ခေါ်ပါတယ်။ Object ထဲက attributes(datas) တွေကို အဲဒီ member function တွေကပဲ ရယူနိုင်ပါတယ်။ တကယ်လို့ Object တစ်ခုထဲက data တစ်ခုခုကို ရယူချင်တယ်ဆိုရင် သက်ဆိုင်ရာ member function ကို ခေါ်ယူ အသုံးချရမှာ ဖြစ်ပါတယ်။ အဲဒီ member function က လိုအပ်တဲ့ data ကို object ထဲက ရယူပြီး return ပြန်ပေးမှာ ဖြစ်ပါတယ်။

```
public void SetModel(String mdl)

{

    Model = mdl;

}

public String GetModel()

{

    return Model;

}
```

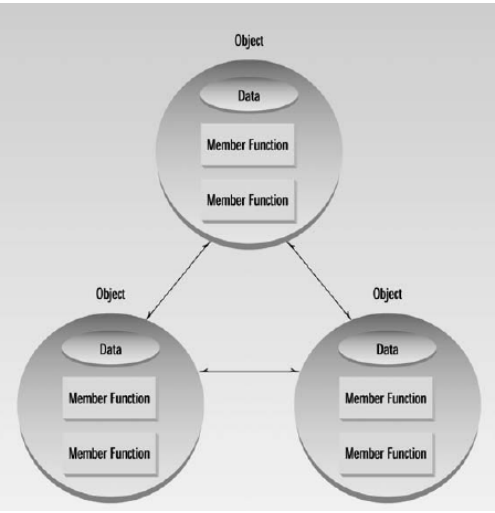
Object တစ်ခုထဲက ဒေတာ တွေကို တိုက်ရိုက်ရယူဖို့ မဖြစ်နိုင်ပါဘူး။ ဒေတာတွေကို တိုက်ရိုက် ကိုင်တွယ်ခြင်းကြောင့် ပြဿနာတွေ ဖြစ်ခဲ့တဲ့ အကြောင်း၊ global variable တွေရဲ့ ပြဿနာတွေကို ရှင်းပြခဲ့ပြီးပါပြီ။ ဒါကြောင့် OOP မှာတော့ ဒေတာတွေကို သက်ဆိုင်ရာ Object ထဲမှာပဲ ဖုံးကွယ် သိုဝှက်ထားခြင်း (hidden) နဲ့ မတော်တဆ ပြုပြင်ပြောင်းလဲ ခံရခြင်းကနေ

ကာကွယ်ပေးထားပါတယ်။ အပြင်လောကမှာ မိမိ အရပ်အမောင်းကို သူများက သတ်မှတ် ဖန်တီးပေးလို့ မရသလိုပါပဲ။ အဲဒီလို hidden လုပ်ဖို့ ပရိုဂရမ် အတော်များများမှာ private ဆိုတဲ့ keyword နဲ့ ဖန်တီးယူရပါတယ်။ တကယ်လို့ variable တစ်ခု ကြေငြာရာမှာ ဘာမှ ပြောမထားဘူးဆိုရင် default အနေနဲ့ private လို့ ယူဆလေ့ ရှိပါတယ်။ အဲဒီလို ဒေတာတွေကို ဖွက်ထားခြင်းကို data encapsulation လို့လဲ ခေါ်ဆိုကြပါတယ်။



Message

OOP နဲ့ ရေးသားထားတဲ့ ပရိုဂရမ် တစ်ခုမှာ object အများအပြား ပါဝင်နိုင်ပါတယ်။ အဲဒီ object အချင်းချင်း member function တွေခေါ်သုံးခြင်းအားဖြင့် ဆက်သွယ်လုပ်ကိုင်နေကြပါတယ်။

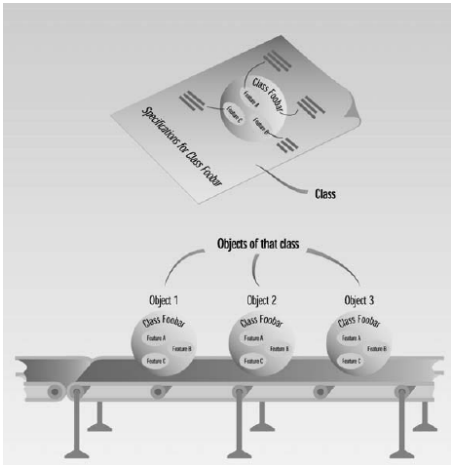


Object ရဲ့ member function တစ်ခုကို ခေါ်ယူခြင်းကို အဲဒီ object ကို message တစ်ခု ပေးပို့ခြင်းလို့လည်း ခေါ်ဆိုနိုင်ပါတယ်။ ဌာနတစ်ခုမှာဆိုရင် manager တစ်ယောက်က ဝင်ငွေ၊ ထွက်ငွေကို သိရဖို့အတွက် စာရင်းကိုင်ဌာနကို ကိုယ်တိုင်သွားပြီး ဖိုင်တွေ ရှာဖွေဖို့ မလိုအပ်ပါဘူး။ သက်ဆိုင်ရာ ဌာနက ဆိုင်ရာ တာဝန်ခံတွေကိုသာ အချက်အလက်တွေ တောင်းခံလိုက်ရုံပဲ ဖြစ်ပါတယ်။ ပရိုဂရမ်းမင်းမှာလဲ ဒီလိုပါပဲ။ ဒါကြောင့် ဒေတာတွေကို ပြင်ပက ပြင်ဆင် နှောက်ယှက်ခြင်း မခံရဘဲ တိတိကျကျ ရယူနိုင်ပါတယ်။

Instantiation

Class တစ်ခုကို define လုပ်ရုံနဲ့ object တွေ ဖန်တီးမပေးနိုင်ပါဘူး။ တကယ်တော့ class define လုပ်ခြင်းဟာ blue print တစ်ခု (သို့) ပုံစံခွက် တစ်ခု ဖန်တီးခြင်းမျှသာ ဖြစ်ပြီး instantiate လုပ်ပေးမှသာ object များကို RAM အတွင်းမှာ ဖန်တီးပေးနိုင်မှာ ဖြစ်ပါတယ်။ ဥပမာ-

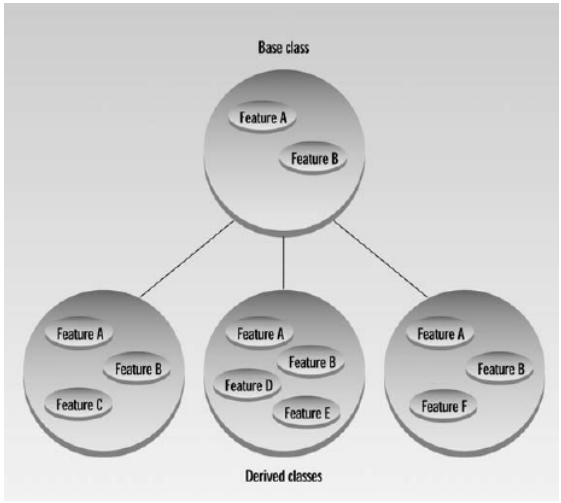
```
Automobile Auto1 = new Automobile();

Automobile Auto2 = new Automobile();
```

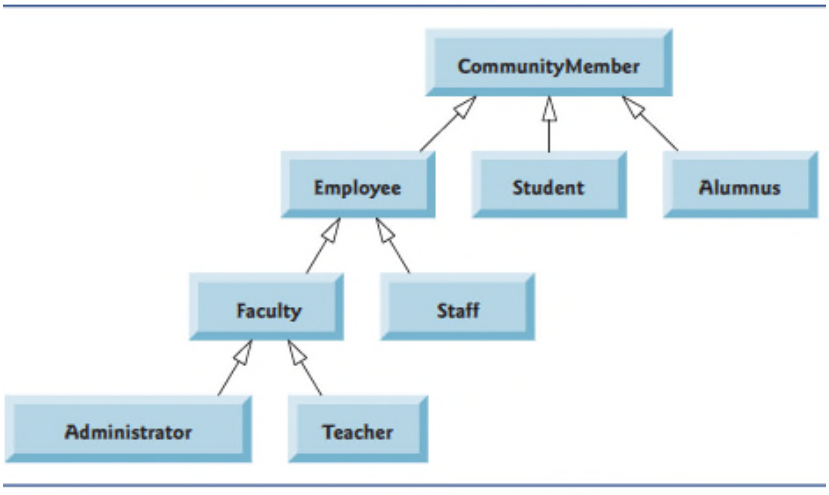


Inheritance

နေ့စဉ်ဘဝမှာ class တစ်ခုကနေ ခွဲထွက်သွားတဲ့ subclass များနဲ့ ထိတွေ့နေရပါတယ်။ ဥပမာ- ရော့ခ်၊ ပေါ့ပ်၊ ဟစ်ဟော့ စတာတွေဟာ Music class ထဲက subclass တွေ ဖြစ်ပါတယ်။ အဲဒီ subclass တွေဟာ derived class (ဒီဥပမာမှာတော့ Music class) မှာ ရှိတဲ့ member data နဲ့

member function တွေကို အမွေ ဆက်ခံထားသလို ကိုယ်ပိုင် data နဲ့ function တွေ ထပ်မံပေါင်းထည့် ပါဝင်နိုင်ပါတယ်။ ဥပမာ - convertible ဆိုတဲ့ ကားအမျိုးအစားဟာ သာမန် automobile ကားအမျိုးအစားကိုပဲ အမိုးတင်နိုင်ချနိုင်တဲ့ အရည်အသွေး ထပ်မံပေါင်းထည့်ထားတာပဲ ဖြစ်ပါတယ်။





Inheritance hierarchy for university CommunityMembers.

Reusability

class တစ်ခုကို ရေးသား ဖန်တီးပြီးလို့ debugged ပြုလုပ်ပြီးပြီ ဆိုရင် အခြား ပရိုဂရမ်မာတွေကို သူတို့ရေးမယ့် ပရိုဂရမ်တွေမှာ အသုံးပြုနိုင်အောင် ဖြန့်ဖြူးပေးနိုင်စွမ်း ရှိလာပါပြီ။ အဲဒါကို reusability လို့ ခေါ်ပါတယ်။ inheritance ကို သုံးခွင့် ရလို့လည်း reusability

ပိုဖြစ်လာပါတယ်။ ပရိုဂရမ်မာ တစ်ယောက်အနေနဲ့ ရှိပြီးသား class တစ်ခုကို တိုက်ရိုက် ပြင်ဆင်ရေးသားဖြင်း မပြုဘဲ inheritance ကိုသုံးပြီး features အသစ်တွေ ထပ်ထည့် ပေးနိုင်လာလို့ပဲ ဖြစ်ပါတယ်။

Creating New Data Types

Object ရဲ့ အားသာချက်တစ်ခုက data types အသစ်တွေကို အလွယ်တကူ ဖန်တီးခွင့်ပေးခြင်းပဲ ဖြစ်ပါတယ်။

Polymorphism and Over loading

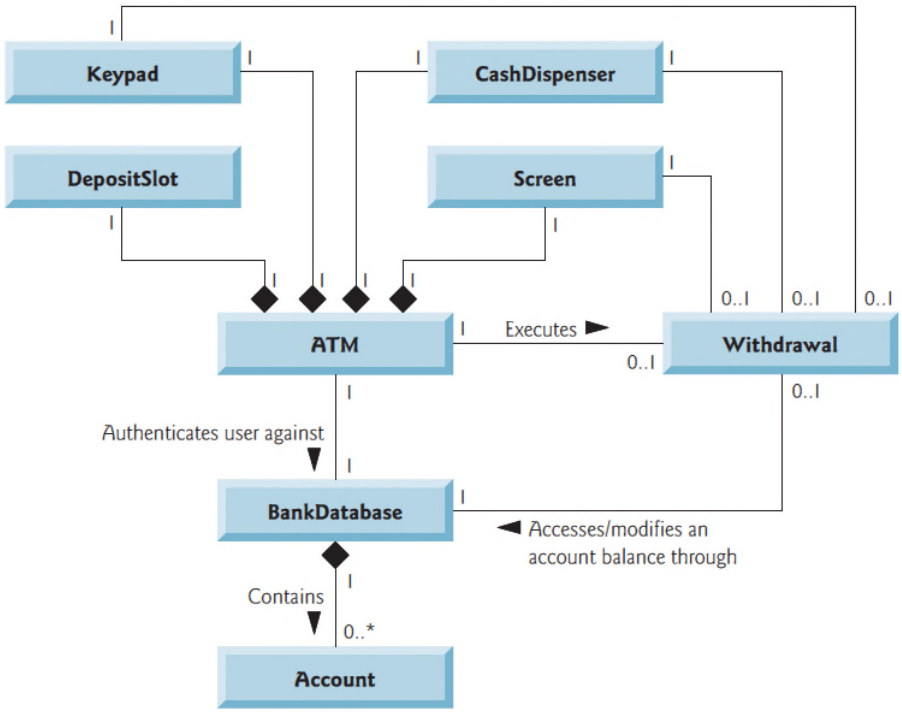
(+, -, =) စတဲ့ operator တွေကို bulid in data types တွေနဲ့ အသုံးပြုဖို့ ဖန်တီးပေးထားပါတယ်။ ကျွန်တော်တို့ ဖန်တီးပေးမယ့် class တွေမှာ အသုံးပြုဖို့ အတွက်ကတော့ class ရဲ့ member function အနေနဲ့ operator overloading သုံးပြီး ရေးပေးရမှာ ဖြစ်ပါတယ်။

Object-Oriented Analysis and Design (OOAD)

ပရိုဂရမ်တစ်ခု စရေးမယ်ဆိုရင် အတော်များများက ကွန်ပျူတာရှေ့မှာ ထိုင်ပြီး စာစရိုက်လေ့ ရှိပါတယ်။ တကယ်တော့ ကြီးမားတဲ့ ပရိုဂရမ်တွေ ဥပမာ- ပရိုဂရမ်မာ တစ်ထောင်လောက် စုပြီးရေးရတဲ့ လေကြောင်းထိန်းသိမ်းရေး software လိုမျိုးမှာဆိုရင် ဒီလို တန်းရေးဖို့ မဖြစ်နိုင်ပါဘူး။ အကောင်းဆုံး solution ကို ရဖို့အတွက် ကျွန်တော်တို့ ရေးသားမယ့် ပရောဂျက်မှာ လိုအပ်ချက်တွေကို အသေးစိတ် စဉ်းစားသတ်မှတ် ရမှာ ဖြစ်ပါတယ်။ ပြီးရင် အဲဒီ လိုအပ်ချက် (requirements) တွေကို ပြေလည်စေမယ့် software ဒီဇိုင်းကို ရေးသားရမှာ ဖြစ်ပါတယ်။ OOP (point of view) အမြင်နဲ့ အဲလို လေ့လာဆန်းစစ်တာကို (OOAD) process လို့ ခေါ်ကြပါတယ်။

UML

OOAD process တွေကနေ ထွက်လာတဲ့ ရလာဒ်တွေကို ဆက်သွယ်ပေးဖို့အတွက် graphical language တစ်ခုကတော့ (Unified Modeling Language) UML ပဲ ဖြစ်ပါတယ်။ class တွေနဲ့ ဘယ်လို ဖွဲ့စည်း တည်ဆောက်ထားပြီး ဆက်စပ်ချက်တွေ ဘယ်လိုရှိတယ်ဆိုတာကို အပေါ်စီးကနေ အလွယ်တကူ ကြည့်ရှုလို့ရအောင် အသုံးပြုတာ ဖြစ်ပါတယ်။ ဘယ်လောက်အထိ progress ပြီးစီးပြီဆိုတာကိုလဲ သိနိုင်ပါတယ်။



အကိုးအကားများ

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

ဒေါက်တာ အောင်ဝင်းထွဋ်

(bluephoenix)

http://tech4mm.com