ကျောင်းတွင် သင်သော စုတ်ပြတ်သတ် လက်ရေးများဖြင့် Notes လိုက်မှတ်ထားသည်များကို ဆီလျော်အောင် ပြန်လည် ရေးသားပါသည်.. မသင့်တော်သည်များကို Admin ကြီး saturngod မှ ပြင်ဆင်ပေလိမ့်မည် :D

အရင်ဆုံး Java အကြောင်း မပြောခင်မယ် Object-Oriented-Programming အကြောင်း အရင် ပြောဖို့လိုမယ်ထင်ပါတယ်..

OOP ဆိုတာ Object ဆိုတဲ့အရာများကို အသုံးပြုပြီးတော့ Apps တွေ Program တွေ Design လုပ်ထားတဲ့ Programming Standard တစ်ခု ဖြစ်ပါသည်.. OOP မှာ နည်းလမ်းတွေ အများအကြီးရှိပါတယ်.. အဲထဲက အဓိကဟာတွေကတော့

- Inheritance
- Polymorphism
- Encapsulation

တို့ဖြစ်ပါတယ်.. အောက်ဘက်မှာ အသေးစိတ် ပြန်ရှင်းပါ့မယ်

အရင်ဆုံး Object ဆိုတာကြီးကို Define လုပ်ကြည့်တာပေါ့

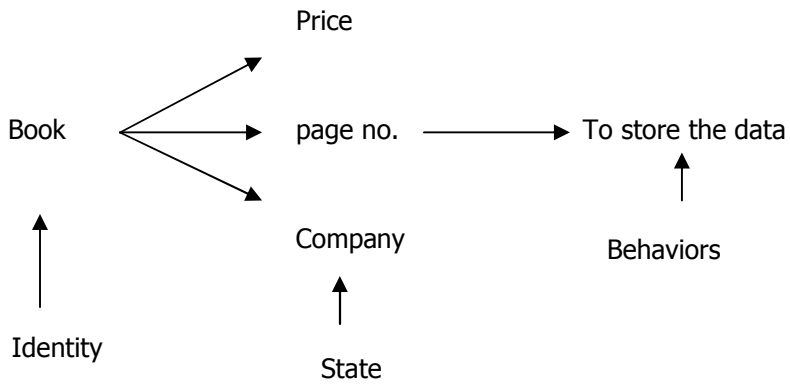
Object ဆိုတာ Subject မဟုတ်တာလို့ ပြောမှာလားဆိုတော့ မဟုတ်ပါဘူး :P

An object is a thing that you can see, hold, or touch လို့ အရှင်းဆုံး ပြောလို့ရပါတယ်.. ကျနော်တို့ ပါတ်ဝတ်ကျင်းမှာ ကိုင်တွယ်နိုင်တဲ့အရာတွေ၊ မြင်နိုင်တဲ့အရာတွေ ၊ ထိတွေ့နိုင်တဲ့အရာတွေ အားလုံးဟာ Object ပါ..

Object ဆိုတာမှာ Characteristics သုံးမျိုး ရှိပါမယ်..

1) Identity — Common Name of a group of Objects
2) State — Information of Identity
3) Behaviors — Functions

ပိုပြီး မြင်သာသွားအောင် ဥပမာ တစ်ခုနဲ့ ပြောပြပါ့မယ်.. စာအုပ်တစ်အုပ် ဆိုကြပါစို့



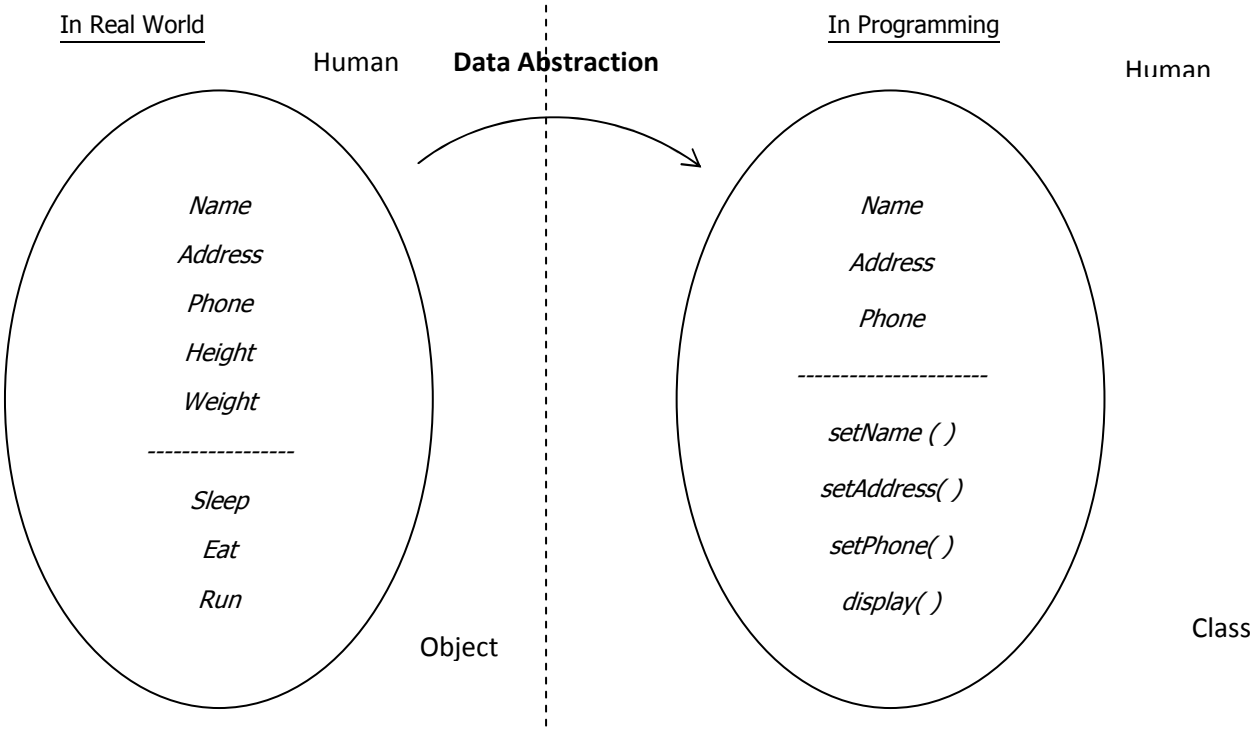
အပေါ်က ဥပမာ မှာ ကြည့်မယ်ဆိုလျှင် Identity, State, Behavior ဆိုတာကို ကွဲကွဲပြားပြား မြင်ရမယ် ထင်ပါတယ်.. Book ဆိုတဲ့ Identity (တစ်နည်း Object) ဆိုတာမှာ Price, Page No. company ဆိုတဲ့ State တွေ ရှိနေပါတယ်.. အဲဒီအောက် Page No. ဆိုတဲ့ Identity အောက်မှာ စာတွေပဲ ဖြစ်ဖြစ် ပုံတွေပဲ ဖြစ်ဖြစ် data တွေ store လုပ်ထားမယ့် State ရဲ့ Function ရှိနေပါတယ်..

Object ဆိုတာနဲ့ ပါတ်သတ်လို့ ပြောစရာ တစ်ခုရှိတာက Natural Materials တွေဖြစ်တဲ့ လေ (Air), မီး (Fire), ရေ(Water) စတဲ့ဟာတွေဟာ Object မဟုတ်ပါဘူး :)

အဲလိုဆို Object အကြောင်းလေး ရှေ့နည်းနည်း ဆက်ကြတာပေါ့

တစ်ခုလောက် ပြောချင်တာက **Data Abstraction** ဆိုတဲ့အကြောင်းလေးပါ..

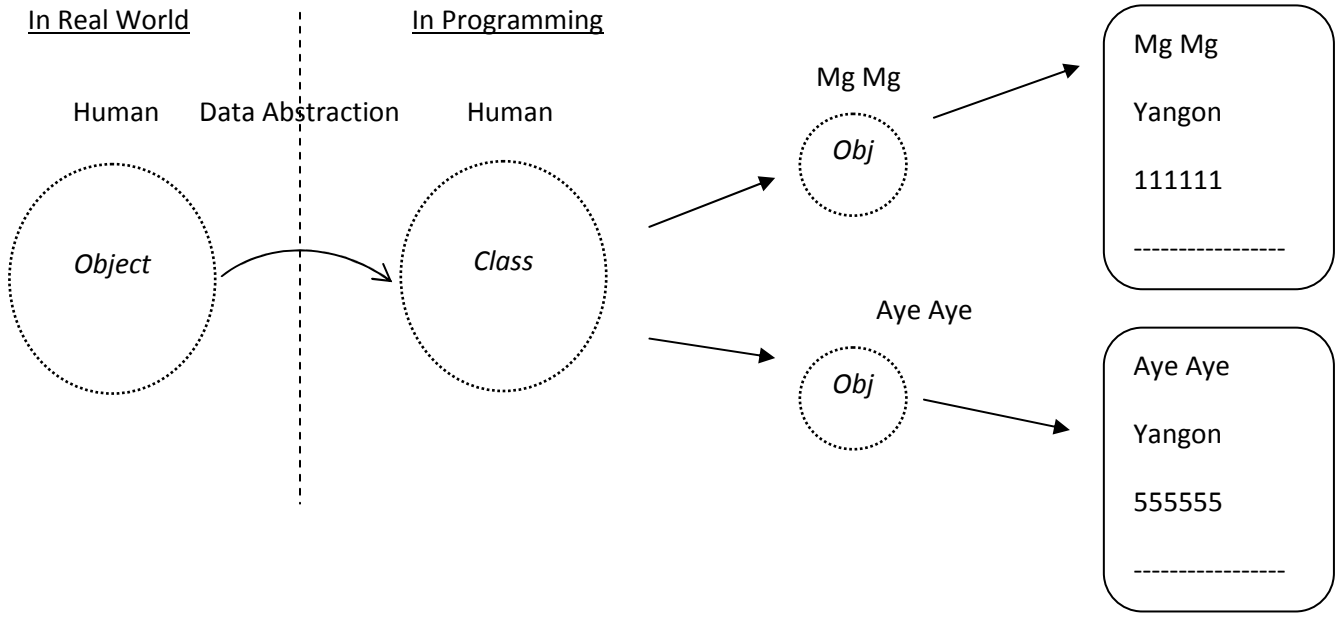
Data Abstract ဆိုတာဟာ ပါဝင်တဲ့ Entity (Object) တွေကို Identifying လုပ်မယ်, နောက် Attribute တွေကို စနစ်တကျဖြစ်အောင်လုပ်မယ် အဲဒီနောက် Entity တစ်ခုချင်းစီနဲ့ သက်ဆိုင်တဲ့ action တွေကို ဆက်လုပ်သွားမယ်... အဲဒီလို Process ကို Data Abstraction လုပ်တယ်လို့ ခေါ်ပါတယ်



ဒီပုံမှာ ကြည့်မယ်ဆိုရင် Human ဆိုတဲ့ Object ထဲမှာ Name, Address, Phone, Height, Weight ဆိုတဲ့ Entity တွေကို တွေ့ရမှာဖြစ်ပါတယ် နောက် Sleep, Eat, Run အစရှိတဲ့ Attribute တွေလည်း ပါလာပါလိမ့်မယ် နောက် Data Abstraction လုပ်လိုက်တဲ့အခါမှာ လိုချင်တဲ့ Entity တွေကိုပဲ Abstract လုပ်လိုက်ပါတယ်.. ပုံမှာ အတိုင်းဆိုရင် Name, Address, Phone ဆိုပြီးတော့ပေါ့

နောက် Entity တစ်ခုချင်းစီနဲ့ ပါတ်သတ်တဲ့ Action တွေဖြစ်တဲ့ setName(), setAddress(), setPhone(), display() စတာတွေကို ဆက်လုပ်သွားပါတယ်.. ဒါပါပဲ

အဲ.. အပေါ်ဘက်မှာ Class ဆိုတဲ့ စာလုံး အသစ်တစ်လုံး တွေ့မိမယ် ထင်ပါတယ်.. Class ဆိုတာဘာလဲ.. သာမန်ဆိုရင်တော့ စာသင်ခန်းပေါ့.. စာသင်ခန်း ဆိုတာမျိုးဟာ ပေါင်းစုထားတဲ့ နေရာဖြစ်ပါတယ်.. ကျောင်းသားတွေ ပေါင်းစုပြီးတော့ တည်ရှိနေတယ်ပေါ့.. Programming မှာလည်း ဒီလိုပါပဲ.. Class ဆိုတာဟာ Object တစ်ခုရဲ့ Abstract Characteristics တွေကို Define လုပ်ထားတဲ့ အစုအစည်းတစ်ခုပါ.. Object ရဲ့ Characters တွေပါမယ်.. Object ထဲမှာပါတဲ့ Attribute တွေ ဒါမှမဟုတ် Properties တွေ နောက် သူ့ရဲ့ Behavior သို့မဟုတ် Method , Feature ဆိုတာတွေ ပါပါ့မယ် ..Object နဲ့ Class ဆက်စပ်ပုံကို ဆက်သွားပါ့မယ်



Real World က Human ဆိုတဲ့ Object ကို ကျွန်တော်တို့ Programming ဘက်မှာ Data Abstraction လုပ်တဲ့အခါမှာ (ကျွန်တော်တို့ Programming ဘက်မှာဆိုလို့ Programming က ကျွန်တော် ပိုင်တာမဟုတ်ပါဘူး :P) Human ဆိုတဲ့ Class ဖြစ်လာပါတယ်.. အဲဒီ Class ထဲမှာမှ Mg Mg ရယ် Aye Aye ရယ်ဆိုတဲ့ Object နှစ်ခု ရှိနေပါတယ်.. နောက် Mg Mg , Yangon , 111111 , ---, Aye Aye , Yangon , 55555, ----- စတာတွေကတော့ သူ့ရဲ့ Properties တွေဖြစ်ပါတယ်.. အလွယ်ဆုံး မှတ်ထားမယ်ဆိုရင်တော့ Object တွေ ပေါင်းစည်းထားတဲ့ အစုအစည်းတစ်ခုကို Class လို့ ခေါ်ပါတယ်..

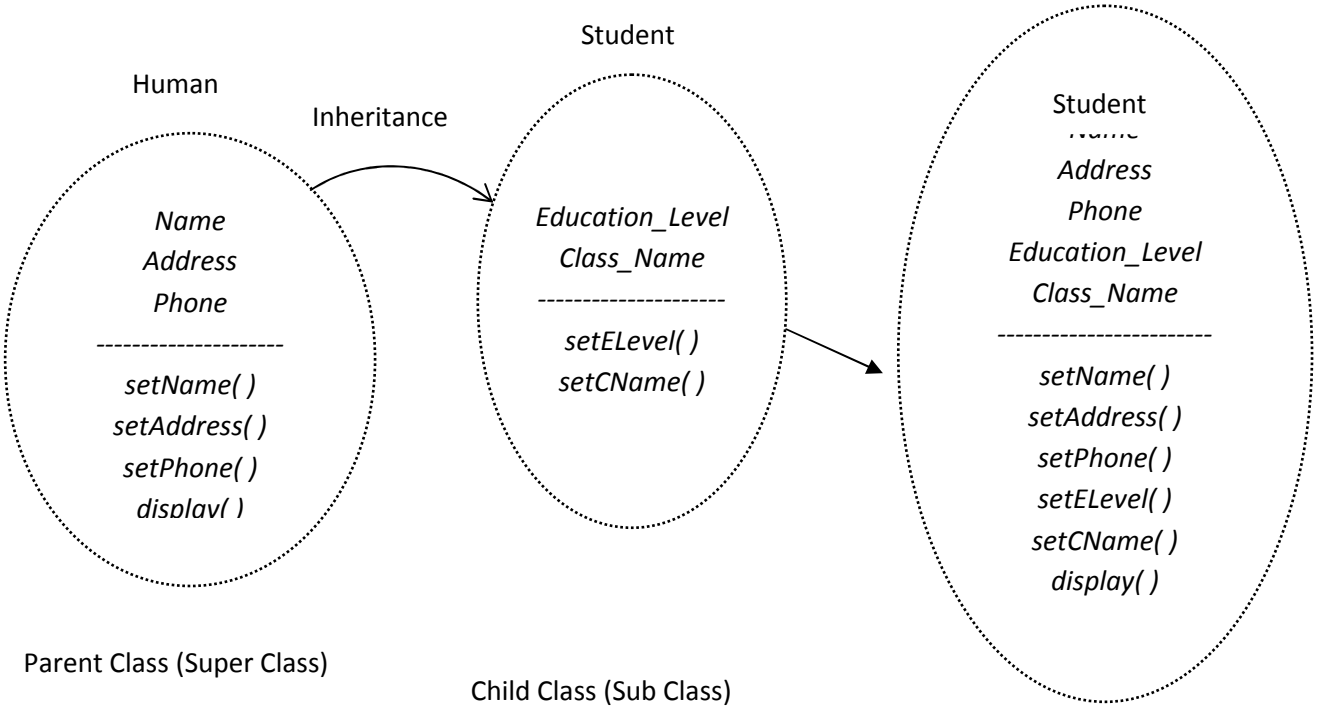
နောက် OOP ရဲ့ အဓိက Principles သုံးခုဖြစ်တဲ့

- Inheritance
- Polymorphism
- Encapsulation

အကြောင်း ဆက်လက်ပြောပါ့မယ် ကို Sevenlamp ရဲ့ basic basic basic of C# မှာ OOP အကြောင်း ပြောထားတာ အင်မတန်ကောင်းလှပါတယ်.. အဲဒါကိုပဲ ကျနော် နည်းနည်း ထပ်ဖြည့်လိုက်ပါတယ် (ဖြတ်-ညှပ်-ကပ် လုပ်တယ်ခေါ်မလားပဲ :P)

1. **Inheritance**

ဒီလိုလေးတွေးကြည့်ရအောင်ဗျာ အခုကျွန်တော်တို့ မိဘတွေပိုင်ဆိုင်တဲ့ ပိုင်ဆိုင်မှုတွေဟာ ကျွန်တော်တို့ရဲ့ ပိုင်ဆိုင်မှုတွေပါပဲ။ ဒီလိုပါပဲ ကျွန်တော်တို့ရဲ့ ပိုင်ဆိုင်မှုတွေကိုလည်း ကျွန်တော်တို့ရဲ့ မျိုးဆက်သစ်တွေက ပိုင်ဆိုင်ကြဦးမှပါ။ ဒီသဘောတရားလေးကို ယူပြီး OOမှာ inheritance ဆိုတာပေါ်ပေါက်လာတာပါ။ class တစ်ခုရဲ့ ပိုင်ဆိုင်မှု property တွေဟာ အဲ့ဒီ class ရဲ့ child တွေကလည်း ပိုင်ဆိုင်နိုင်ပါတယ်။ ဒီလို ပိုင်ဆိုင်နိုင်ခြင်းအားဖြင့် ကျွန်တော်တို့က class တွေအားလုံး အတွက် တူညီတဲ့ property တွေကို အကြိမ်ကြိမ်ကြေငြာပေးနေစရာ မလိုတော့ပါဘူး။ ဥပမာဗျာ ကျွန်တော်က လူတစ်ယောက်ကို ကိုယ်စားပြုမယ့် class တစ်ခုဆောက်မယ်ဆိုရင် အဲ့ဒီ class ထဲမှာ ဘယ်လို member တွေပါမလဲ စဉ်းစားကြည့်၊ နံမည်ပါမယ်၊ နေရပ်လိပ်စာပါမယ် ဖုန်းနံပါတ်ပါမယ် ဗျာ၊ စသည်ဖြင့်ပေါ့။ အောက်က ပုံလေး နဲ့ ယှဉ်ကြည့်နိုင်ပါတယ် နောက် အခြား Class တစ်ခုကနေ နောက်တစ်ခုဆီသို့ သူ့ရဲ့ Behavior တွေ Action တွေ Inherit လုပ်ထားတဲ့ Class ကို **Subclass** လို့ ခေါ်ပါတယ် (ဒါမှမဟုတ် **Child Class**) Inherit အလုပ်ခံရတဲ့ Class ကိုတော့ **Super Class** လို့ခေါ်တယ်.. တစ်နည်း **Parent Class** ပေါ့ ..



အပေါ်မှာဆိုရင် Student ဆိုတဲ့ Sub Class ဟာ Human ဆိုတဲ့ Super Class ဆီကနေ Inherit လုပ်ထားတာကို မြင်ရမှာပါ.. အလားတူ နောက် step တစ်ခုမှာလည်း ထိုနည်းအတူပါပဲ

2. **Polymorphism**

ကျွန်တော်တို့ ရပ်ကွက်တွေမှာလုပ်လေ့ရှိတယ်ဗျာ တခါတလေ ရပ်ကွက်ရုံးကနေပြီးတော့ တစ်အိမ်ကို လူတစ်ယောက်နှုန်း အစည်းအဝေးတက်ရမယ်တို့ ဘာတို့ ကြုံဖူးမှာပေါ့။ ဟုတ်ကဲ့ ရပ်ကွက်ရုံးက ခေါ်တာက လူတစ်ယောက်လို့ ခေါ်တာပါ တနည်းအားဖြင့် human class ကို ခေါ်တာပါ။ အဲ့ဒါကို အိမ်မှာ ရှိနေတဲ့ ကျောင်းသားတစ်ယောက်က သွားတက်လို့ မရဘူးလား၊ ရပါတယ်။ Teacher တစ်ယောက်သွားတက်မယ်ဆိုလို့လဲ ရပါတယ်။ Teacher ကလည်း human တစ်ယောက်ပဲ ဖြစ်လို့ပါ။ ဒါပေမယ့် Student ပဲ သွားသည်ဖြစ်စေ၊ Teacher ပဲသွားသည်ဖြစ်စေ၊ ဒါမှမဟုတ် Human ကိုယ်တိုင်ပဲ သွားသည်ဖြစ်စေ၊ အားလုံးကို Human လို့ပဲ မြင်ပါတယ်။ အားလုံး အခွင့်အရေးတန်းတူပါပဲ။ Student မို့ အခွင့်အရေးပိုရမလား မရပါဘူး။ အိုကေ အဲ့ဒီလိုမျိုး Human ကိုခေါ်တာကို Stuent သွားလိုက်တာကို အသွင်ပြောင်းလဲခြင်း (polymorphism) လို့ ခေါ်ပါတယ်။

Polymorphism မှာ သတိထားရမည့် အချက်ကတော့ child ကို parent အသွင်ပြောင်းနိုင်ပေမယ့် parent ကိုတော့ child အဖြစ်ပြောင်းလို့ မရပါဘူး။ ဆိုလိုတာကတော ကျောင်းသားအားလုံး စာမေးပွဲလာဖြေပါဆိုပြီး ခေါ်တာကို အိမ်မှာ ရှိတဲ့ လူကြီးတွေသွားဖြေလို့ ရမလား မရပါဘူး၊ ဒီသဘောပါ။ Student ကိုတောင်းရင်တော့ Human ပေးလို့ မရပါဘူး၊ Student ပဲ ပေးမှ ရမှာပါ။

အဲ ဒီနေရာမှာ Data Hiding ဆိုတာလေး ကို ထည့်ပြောချင်တယ်.. နံပါတ်ပဲ ထိုးလိုက်ပါ့မယ်...

3. **Data Hiding**

Data hiding ဆိုတာကတော့ ကျွန်တော်တို့ အိမ်မှာ မိသားစုပိုင်ဆိုင်မှုတွေကို တခြားလူတွေသိအောင် ထုတ်ကြေငြာလေ့ရှိလား? မရှိပါဘူး တိတ်တိတ်လေးပဲ သိမ်းထားလေ့ရှိပါတယ်။ ဒီသဘောပါပဲ class တွေထဲမှာရှိတဲ့ member data တွေကိုလည်း အပြင်ကနေ ခေါ်လို့မရအောင် private လုပ်ထားခြင်းကို data hiding လို့ ခေါ်ပါတယ်။

4. **Encapsulation**

Encapsulation ဆိုတာကတော့ ကျွန်တော်တို့ hide လုပ်ထားတဲ့ data ထဲက လိုအပ်တဲ့ dataတွေကို function တွေကနေသော်လည်းကောင်း၊ property တွေကနေတဆင့်သော်လည်းကောင်း၊ အသုံးပြုနိုင်ဖို့အတွက် စီမံပေးခြင်းပဲ ဖြစ်ပါတယ်။

အခုလောက်ဆိုရင်တော့ Object Oriented Programming ဆိုတာ ဘာလဲ, ဘယ်လို Character တွေ ရှိသလဲဆိုတာ အကြမ်းဖျဉ်း သိလောက်မှာပါ..

Brief History of JAVA

၁၉၉၁ မှာ (၉၁ ဆိုတော့ ကျနော့်ထက် တစ်နှစ်ကြီးတဲ့ အစ်ကိုကြီးပဲ Big Brother Java :P :P) James Gosling ဆိုတဲ့သူရဲ့ Oak အမည်ရတဲ့ Project ကနေ ထွက်ပေါ်လာပါတယ်.. C / C++ ရဲ့ Notation တွေကို အခြေခံပေမယ့် C/C++ ထက် ပိုပြီး ရှင်းလင်းလွယ်ကူအောင် လုပ်ဖို့ပါပဲ.. ပထမဆုံး Public ကို Implementation လုပ်တဲ့ Java 1.0 ဟာ 1995 မှာ ထွက်ပေါ်လာပါတယ်.. လူအကြား ရေပန်းစားလာတဲ့ အကြောင်းက **"Write Once, Run Anywhere"** ဆိုတဲ့ အချက်နဲ့ပါ ဆိုလိုချင်တာက Java ဟာ Platform Dependency မရှိပါဘူး.. Window ပေါ်မှာ ရေးထားတဲ့ Java codes တွေကို Unix ပေါ်မှာ run လို့ ရပါတယ်.. Unix ကို အခြေခံထားတဲ့ Linux, Mac မှာလည်း run လို့ ရပါတယ်.. ဒါ သူရဲ့ အားသာချက်လို့ ပြောနိုင်ပါတယ်..

နောက်ပိုင်းမှာ large enterprises တွေအတွက် J2EE (Java 2 Platform, Enterprise Edition), Mobile အတွက်J2ME (Java 2 Platform, Micro Edition) ဆိုပြီး ထွက်ပေါ်လာပါတယ်..

Java Language ရဲ့ အဓိက ရည်ရွယ်ချက်တွေက

၁. Java language ဟာ Object Oriented Programming methodology တွေကို Support လုပ်ပါမယ်

၂. ဒီ Program ကိုပဲ Multiple Operating Systems မှာ run နိုင်ရပါမယ် Platform Dependency မရှိရဘူးပေါ့..

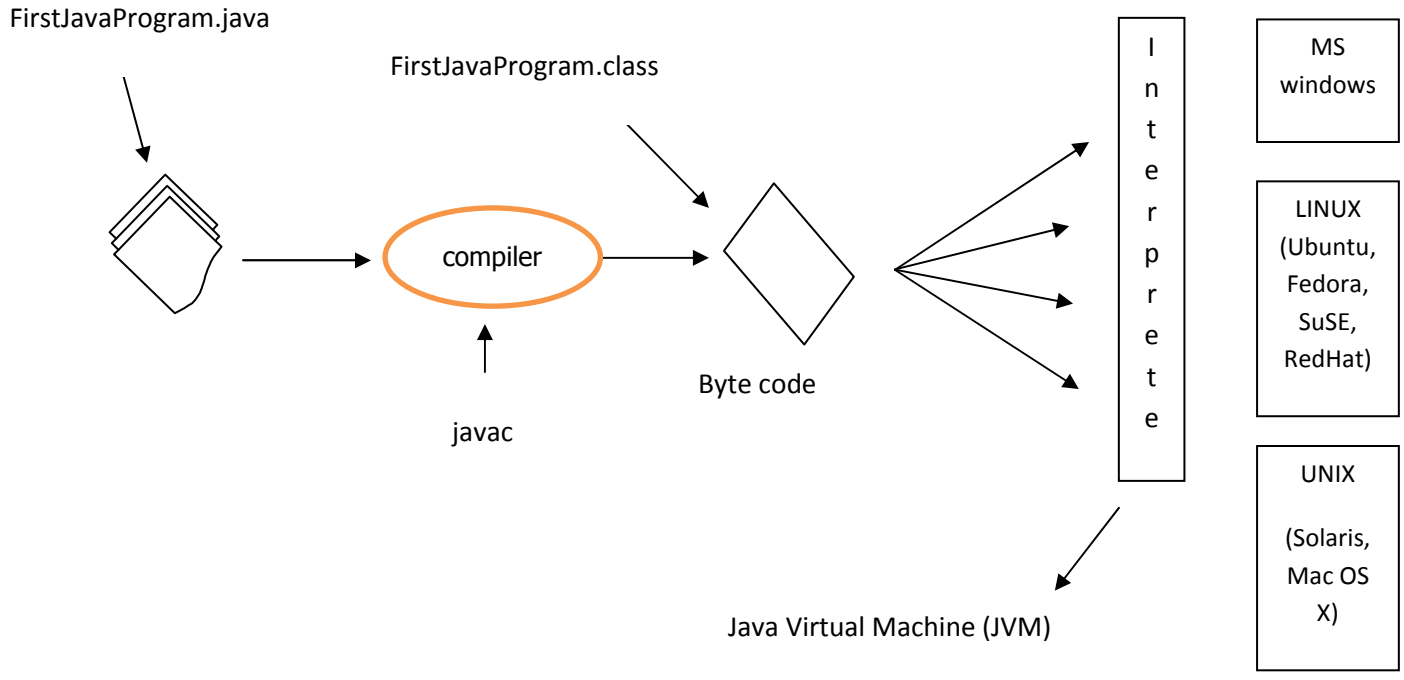
၃. Computer network ကို Built-In support လုပ်ပါမယ်..

၄. တခြား Remote Sources တွေကနေ Security ကောင်းမွန်စွာ Execute လုပ်နိုင်အောင် Design လုပ်ထားပါတယ်..

၅. OOP ကို support လုပ်ထားတဲ့ Language တစ်ခုအနေနဲ့ လေ့လာရ လွယ်ကူစေရပါမယ်..

နောက် Java ဟာ ဘာကြောင့် Platform Independent ဖြစ်ရတာလဲလို့ ရှင်းပြပါ့မယ်.. အောက်က ပုံလေးမှာ မြင်နိုင်ပါတယ်..

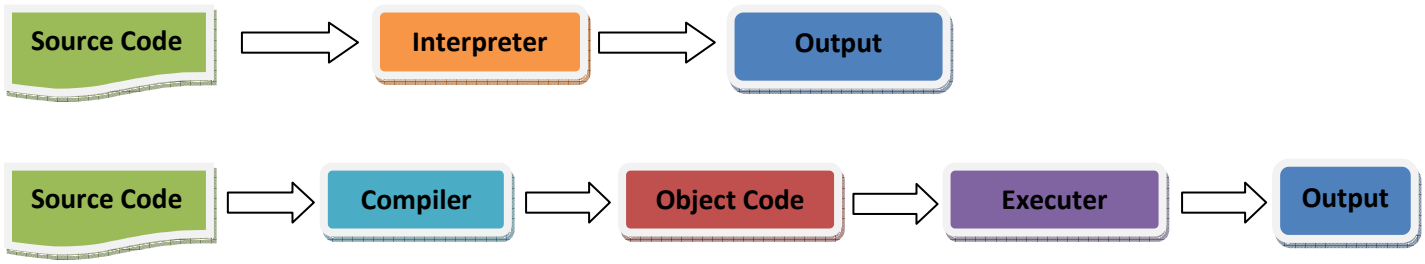
Platform Independent



အရင်ဆုံးက FirstJavaProgram.java ဆိုတဲ့ java code ကို JavaC လို့ ခေါ်တဲ့ Compiler နဲ့ Compile လုပ်လိုက်ပါတယ်.. အဲဒီအခါမှာ FirstJavaProgram.class ဆိုတဲ့ Class ဖြစ်လာပါတယ်.. အဲဒီ Class ဟာ Byte Code အနေနဲ့ ရှိနေတာပါ..

ဒီအချက်ကြောင့် Java ဟာ Platform Independent ဖြစ်နေတာပါ.. Byte Code ဆိုတာ computer တစ်လုံးမှာ အခြေခံအကျဆုံး ဖြစ်တဲ့ 10011100 တွေပါ.. အဲဒီ byte code တွေကိုမှ Java Virtual Machine (JVM) လို့ခေါ်တဲ့ Interpreter နဲ့ ပြန်ပြီး interpret လုပ်လိုက်တဲ့အခါမှ လိုချင်တဲ့ Program ထွက်လာပါတယ်.. Compiler နဲ့ Interpreter ကွာခြားချက်က.. compiler ဟာ High-level programming language တစ်ခုခု (ဥပမာ C/C++, Java) ကို Low-level programming language ဖြစ်တဲ့ Assembly ဖြစ်ဖြစ် Machine Code ဖြစ်ဖြစ် ပြောင်းပေးတဲ့ Program ငယ်လေး ဖြစ်ပါတယ်.. အဲဒီ Assembly Code / Machine Code ကိုမှ run-able program အဖြစ် ပြောင်းပေးတာက Interpreter ဖြစ်ပါတယ်.. နောက်ပိုင်း ထွက်ပေါ်လာတဲ့ Language တွေဖြစ်တဲ့ Ruby, Python, PHP အစရှိတဲ့ Language တွေဟာ Interpreter ကို သုံးပါတယ်.. Source Code ကနေ Executable Program အဖြစ်သို့ တိုက်ရိုက် Convert လုပ်ပေးသွားပါတယ်..

အောက်က ပုံလေးကို ယှဉ်ကြည့်စေချင်ပါတယ်.



အခု ကျွန်တော်တို့ Program စရေး ကြည့်ပါ့မယ်.. Java မှာ IDE ဆိုတဲ့ Integrated Development Environment Tools တွေ အများကြီး ရှိပါတယ်.. အကုန်လုံးကတော့ သူ့ဟာနဲ့သူ အားနည်းချက် အားသာချက်တွေ ရှိကြတာပါပဲ.. အရင်တုန်းက Eclipse ကို သုံးကြပါတယ်.. နာမည်လည်း ကြီးပါတယ်.. ခုထိလည်း သုံးနေကြတုန်းပါပဲ.. ခု ကျောင်းမှာသုံးတာတော့ NetBeans ပါ.. NetBeans က Eclipse ကို Develop ဆက်လက် လုပ်ထားတဲ့ Tools ပါပဲ.. အခု Version 6.7.1 အထိ ရောက်သွားပါပြီ.. လတ်တလောမှာတော့ Java ကိုသာ မက Java နဲ့ ဆက်နွယ်နေတဲ့ Java EE, Java Web, Java ME, နောက် Open Source Language တွေဖြစ်တဲ့ PHP, Python, Ruby, C/C++ တို့ကိုပါ Support လုပ်လာပါတယ်.. Plug in လည်း ပေါများပါတယ်.. Support လည်း ကောင်းပါတယ်.. Linux သမား အတော်များများကတော့ NetBeans ကို အသုံးပြုကြပါတယ်.. ကျွန်တော်ကတော့ Ubuntu မှာရော Window မှာပါ Netbeans ကို သုံးပါတယ်.. နည်းနည်းတော့ လေးပေမယ့် Free ရတာရယ် ကိုယ် အသုံးပြုတဲ့ Language တွေကို Support လုပ်တာရယ်ကြောင့် ကြိုက်ပါတယ်.. နောက် Java ဘက်မှာဆိုရင် .NET က Visual Studio လို Button တွေ Text Box တွေ Label တွေကို Drag and Drop ဆွဲလို့ ရပါတယ်.. ဒီဘက်က ဆွဲ ဟိုဘက်က Code က Generate လုပ်သွားတဲ့ လူသက်သာသွားပါတယ် :D

အစပိုင်းမှာတော့ Microsoft ရဲ့ ဈေးအကြီးဆုံး Tools ဖြစ်တဲ့ Notepad နဲ့ပဲ စကြတာပေါ့ဗျာ :P . Java Run ဖို့အတွက်က JDK လို့ခေါ်တဲ့ Java Development Kit ကို အရင်ဆုံး ဒေါင်းလုပ်ချဖို့ လိုပါတယ်.. JDK Install ပြီးရင်တော့ Notepad နဲ့ စကြတာပေါ့ သူ့ Language မှာ အဓိက အားဖြင့်က အစိတ်အပိုင်း ငါးခုပါပါတယ်

① Identifiers

② Separators

③ Keywords

④ Literals

⑤ Operators ဆိုပြီး ဖြစ်ပါတယ်..

1) Identifiers

အရင်ဆုံး Identifier ဆိုတာဘာလဲ ??

Identifier ဟာ စာလုံး တစ်လုံး (သို့) Underscore (သို့) Dollar Sign နဲ့ စတင်ပါတယ်.. နောက်စာလုံးတွေကတော့ letter, digit, dollar sign နဲ့ underscore တွေ ဖြစ်နိုင်ပါတယ်..

နောက် Identifier မှာ Special Character နှစ်ခုပဲ ပါခွင့်ရှိပါတယ်. နှစ်ခုပဲဆိုတာ နှစ်ခါပဲ ပါရမယ်လို့ ပြောတာမဟုတ်ပါဘူး.. အဲဒီနှစ်ခုက dollar sign နဲ့ underscore တို့ ဖြစ်ပါတယ်... တခြား @,#,^,&,*,! စတာတွေ ပါလို့မရပါဘူး

နောက် သတိချပ်ရမယ့် အချက်က Identifier မှာ Space ပါခွင့်မရှိပါဘူး

နောက်ဆုံး တစ်ချက်ကတော့ Identifier ဟာ Key Word ဖြစ်လို့မရပါဘူး.. Key Work အကြောင်းက နောက်မှာ လာပါလိမ့်မယ်..

2) Separators

Program တစ်ခုကို အပိုင်းပိုင်းလိုက် ပိုင်းပေးတဲ့ sign တွေဖြစ်ပါတယ်..

ဥပမာ - { } ကို Program ရဲ့ Scope ကို Define လုပ်ပေးပါတယ်.. နောက် ; (semi column) ကို code statement တစ်ခုရဲ့ အဆုံးမှာ သုံးပါတယ်.. အဲဒီလို ဟာကို separator လို့ ခေါ်ပါတယ်

3) Keywords

Java မှာ ကြိုတင်ပြီး သတ်မှတ်ထားတဲ့ Keywords တွေရှိပါတယ်.. ဥပမာ "class", "import", "public" စသည်ဖြင့် ဖြစ်ပါတယ်..

အောက်ကဟာတွေကိုတော့ pre-defined keyword များ ဖြစ်ပါတယ်..

abstract	float	Private
boolean	for	Protected
break	future	Public
byte	generic	Rest
byvalue	goto	Return
case	if	Short
cast	implements	Static
catch	import	Super

char	inner	Switch
class	instanceof	Synchronized
const	int	This
continue	interface	Throw
default	long	Throws
do	native	Transient
double	new	Try
else	null	Var
extends	operator	Void
final	outer	Volatile
finally	package	While

4) Literals

Program တွေထဲမှာပါတဲ့ Constant Value တွေကို Literal လို့ခေါ်ပါတယ်..

5) Operators

Data ဒါမှမဟုတ် Object တွေ Calculation လုပ်တဲ့အခါ ဒါမှမဟုတ် Value တွေ Define လုပ်ပေးတဲ့အခါမှာ သုံးပါတယ်.. Java ဆိုရင် များပြားလှတဲ့ Operators တွေကို မြင်ရမှာဖြစ်ပါတယ်

e.g, + , - , ++ , < , = = , < = , > = , !=