Java

Computer understands binary language(1s and 0s)

LowLevel Language - Assembly language (Close to machine understandable)

HighLevel ProgrammingLanguage Languages - C, C++, Java, Python, Ruby...

C, C++ are considered low level

Compilers /interpreters are softwares used for programs compilations/executions.

Compiler - This takes entire code as input at once and Intermediate

object code is generated by compiler.

example for compiled languages: c,c++,scala, smalltalk

byte code(intermediate code) is object code which is actually processes by virtual machine, it is different

from machine code which processor can understand

Interpreter - It takes single line or instruction as an input and executes it

, no intermediate code generated , it is faster, memory requirement is less

example for interpreted languages: Ruby, Python

int add(int a, int b){

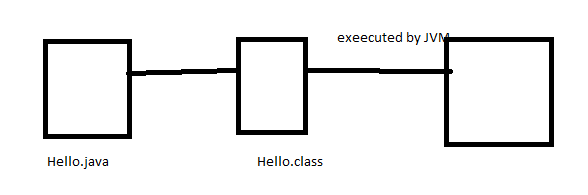
int sum = a+b;

return sum;

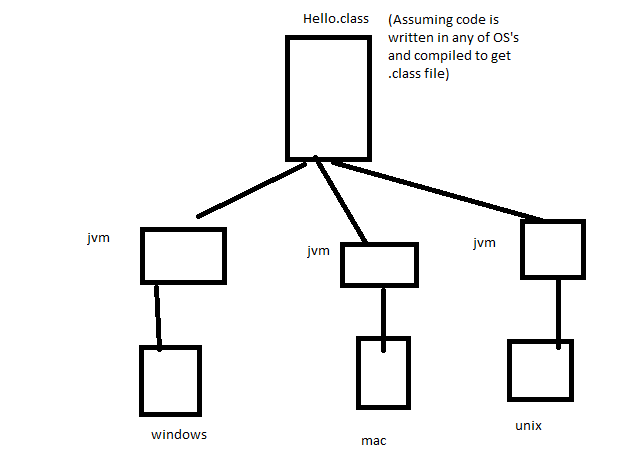
}

Hello.java --> Hello.class(byte code)-> understood by virtual machine

and it should be converted to binary code



Java follows WORA principle(write once and run anywhere) - as it is system independent



System Independent Language - can be executed on any machine irrespective of which OS

you have used to write and compile the code.

System dependent Language - need to compiled again if you want to execute in

some other machine with different OS.

JVM(java virtual machine) is software system dependent but it makes Java language system independent.

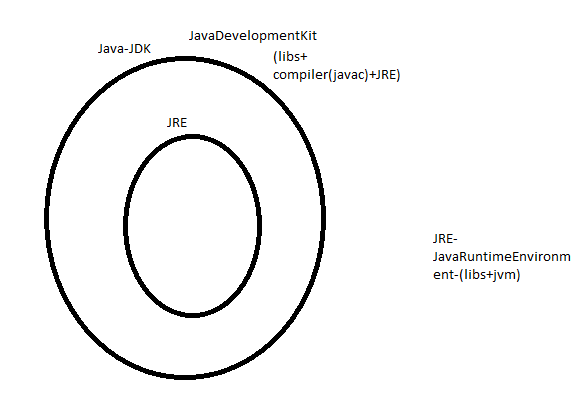
Java Features:

* Simple
* ObjectOriented - Everything in Java is inside a class and they are accessed using objects for class
* SystemIndependent
* Distributed- can communicate over network applications and with protocols tcp/ip , udp

It became popular for internet based appications which desire system independent feature

* Robust- supports good exception handling mechanisms and it handles memory and deallocation internally(by jvm)
* Secure
* MultiThreaded

**Java- jdk - jre :**

****

Download:

http://www.oracle.com/technetwork/java/javase/downloads/index-jsp-138363.html

click jdk download, accept agreement

Java SE Development Kit 8u121 - under this download your OS specfic executable

Go to downloads-> double click the executable

Set Env variables- JAVA\_HOME -C:\Program Files (x86)\Java\jdk1.8.0\_101

edit path variables and add C:\Program Files (x86)\Java\jdk1.8.0\_101\binary

After you download Java- observer the download folders-we see

Java is case sensitive:

single line comments: //

multi line comments:/\* \*/

Class: Class is a blueprint that defines variables(properties) and methods(actions).

Object: Object is instance of class which allows to access properties and methods of class.

Object is the things which you see in real world - Animal, Vehicle, Printer, Mobile, Loan, Account,Customer etc

Class Animal{

//variables- properties

String name;

String breed;

String color;

//methods

public void eatFood(){

} }

Animal a1 = new Animal();

a1.name="punto";

a1.color="white";

a1.eatFood();

Animal a2 = new Animal();

a2.name="princess";

a2.color="brown";

a2.eatFood();

public class Calculator{

public int add(int a, int b){

return a+b;

}

}

public static void main(String[] args){

A a = new A();

a.logic();

}

Explain what main method syntax is meant for:

public - accessable from anywhere

static- Can be accessed ithout creating object

- to call statci method or variable we do not need to create object

void- method is not returning anything back to whoever is calling the method

main- this is the main method that jvm looks for to start java program execution

String[] args- method input parameters

[]- means its list/array of string elements

Eclipse Tool

New Java Project - Package - Class

.project, .classpath

package, import statements, class definitions, methods

Class – model or blueprint which has properties and methods

Variable – Variable is a container which hold the values.

Method – are actions which define logic of object

Object – instance of class. this has the properties and it can call class methods

Mobile – class

color, size, buttons – variables

make a call, send a email, click pic, post, delete – operations/ methods

iphone, Samsung, google - object

Furniture – class

chair, table, bed, sofa – objects

color, legs, size, material – variables

sit, sleep – methods

Syntax for Variables :

accessModifier dataType variableName;

String cloor;

**int** size;

Syntax for Methods :

accessModifiers returnType methodName(){ code }

**public** **void** makeACall(String inputColor){

System.***out***.println("I can make a call now..");

}

public int makeACall(){

return 1;

}

**variable syntax :**

accesModifier dataType nameOfVariable;

eg:

public int id=10;

public String name=”Alan”;

**method synatax:**

access modifier returnType/output methodName(input parameters){

}

return type and input parameters are not mandatory to be present in every method.

access modifier returnType/output methodName(){}

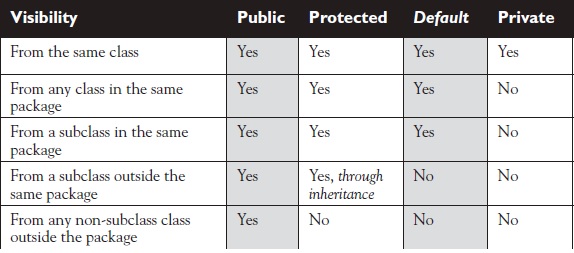
access modifier void(not returning anything) methodName(){

}

access modifier void(not returning anything) methodName(input parameters){

}

AccessModifiers- to define scope of class, method and variable



public, protected, default, private

**Rules for Names/Identifiers:**

Do not start with a number, it can only start with numbers, $, \_, it can include numbers.

Correct : mobile, mobile123, m12obil$\_e, $mobile, \_mobile, m12548

Wrong : 1mobile, 123455,

**Idetifier Rules & Naming Convetions**

Java is Case Sensitive

Class names, Interfaces – initCaps – HelloWorld, Hello, MobileTest

Package Names – in lower case – com.wbl.oops

variables, methods – camelCase - noOfButtons, size

constants – All Caps, **double** PI = 3.14;

IDE- Integrated development Environment

Java- Eclipse, IntellijIdea, NetBeans

Javascript- webstorm, phpstorm, cloud9

Python- Pydev

Eclipse download: <https://www.eclipse.org/downloads/>?

Maven – build tool – this does not replace eclipse

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Class – is a blue print

Object – instance of class

package – folder structure

Access Modifiers:

public – accessible from anywhere

private- accessible only within class

**protected**- It is accessible within the package + subclasses even if they are in other package

Protected variables can be accessed in other package only through inheritance.

package/default- It is accessible within the package

Inheritance :

A{

}

B extends A{

}

main🡪 displaySUbjects()🡪subjects()

**Constructor in Java:**

Constructor is called before creating an object.

It is similar to a method but it will have same name as class name,

It do not have return type like method, but it can take input arguments.

We can have overloaded constructors – means having more than one constructor in the same class

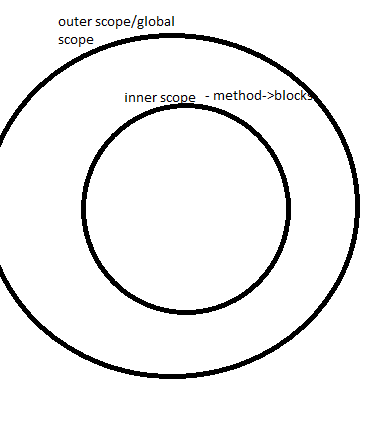
with difference in parameters(either no of parameters or datatype should be different)

Constructors are useful to initialize object data at the time of creation of object.

this - key word that refers to current object of the class

We use it mostly in constructors/ getter setter methods to refer and initialiaze class variables

**variable scopes:**



**static** -

We can use static for variables/methods and blocks.

* Static means class level not object level. Static variables data is shared between the objects.
* we do not need objects to be created for accessing static variable or methods – we can access by class name.
* In a class - we cannot access non static members inside static method/block.
* But we can access static members inside a non static method/block/constructor
* static blocks are called at class loading time- useful for initialization.

example to understand static data is shared between objects

: count no of objects created for a class

**public** **class** Employee {

//global variables or class level variables

**public** String name;

**public** **int** id;

// static variable

**public** **static** **int** *count*=0;

//constructor

**public** Employee(String name,**int** id){

System.***out***.println("constructor is called first...");

**this**.name = name;

**this**.id= id;

++*count*;

}

**public** **void** displayDetails(){

System.***out***.println("name : " + name);

System.***out***.println("id : " + id);

// u can access static variable/method inside non static methods

System.***out***.println(*count*);

}

**public** **static** **void** timeSheet(){

// we cannot access non static variables inside static method/block

System.***out***.println("time sheet");

}

}

**public** **class** EmpTest {

**public** **static** **void** main(String[] args) {

Employee e1 = **new** Employee("Alan", 1);

// e1.name="Alan";

// e1.id=1;

e1.id = e1.id + 1;

// we can access non static variables/methods only with an object

e1.displayDetails();

// we can access static variables/methods without an object

System.***out***.println("count of objects::" + Employee.*count*);

Employee.*timeSheet*();

}

}

**final** : final is keyword which means it cannot be modified

final variables means values cannot be reassigned

final class – we cannot created subclass/child for final class

* cannot be inherited

final methods cannot be overridden.

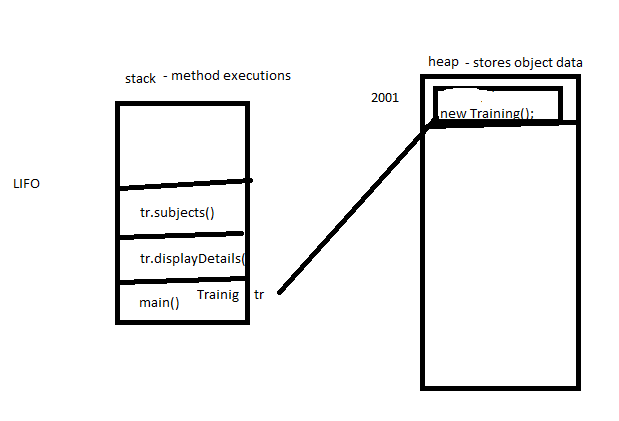
int i=10;

i=11;//valid

final int j=10;

j=11;//invalid

Stack and heap:



OOPS:

Encapsulation

Inheritance

Polymorphism

Abstraction

Abstract class and interface

DataTypes

Operators

loops