Ex:

```
class Hello{
int a=123; int b=0123; int c=ox123;
void show(){
system.out.println(a);
system.out.println(b);
system.out.println(c);
}
}
class Lab3{
Public static void main(String as[]){
Hello h=new Hello();
h.show();
}
}
```

Floating point literals:-

It is used to assign a value to float or double data type.
Two types of notation can be used to represent float value.
a) Standard notation
b) Exponential notation
float f1=10.0f
double d1=109e-5; -> 109*10^-5;
double d2 = 889E3; -> 889*10 per power 3;

Character literals:-

It is used to assign to character data type.
Char ch1='A';
Char ch2='3';
Char ch3='&';
* You have to provide single quotation for character literals;
* Every character enclosed in single quotation mark will have an ascii value as per ascii character set.
* Char ch1=' ' ; not ok, because in single quotation mark you have to provide at least one character. Error--> empty character literals.
* Char ch2='\n';
In this case of scope sequence character you can provide following character data types the \n will be considered as one character.
Ch1='\n'; => for new line
Ch2='\r'; => for carriage return
Ch3= '\t'; => for tab
Ch4= '\||'; =>
Ch5= '\f'; => for form feed
Ch5= 'Error! Hyperlink reference not valid.>

Char data type size two byte:-

While Loop:-

```
class Lab23{
public static void main(String as[]){
int i= Integer.parseInt(as[0]);
int i=1;
while(i<=10){
system.out.println(i*n);
i++;
}
}
}
```

Do While Loop:-

```
class Lab23{
public static void main(String as[]){
int n= Integer.parseInt(as[0]);
while(n<=10){
system.out.println(n);
n++;
}
system.out.println(n);
do{
system.out.println(n);
n++;
}
While(n<=10)
}
}
```


Q-> Different between while/ do-while?

While	do while
->In the case of while loop first condition Will be verified the statement will be executed.	->First statement will be executed then at last condition will be verified.
->If condition false no statement will be executed in while loop	-> In the case of do while loop at least one time statement if the condition is false also.

Unconditional control statement:-

a> Break
b> Continue

a> Break:-

```
class Arithmetic {
int a=10;
void sum(){
system.out.println(a+a);
}
void sum(int p){
system.out.println(p+p);
}
void sum(int p, double d){
system.out.println(p+d);
}
void sum(double d, int p){
system.out.println(d+p);
}
void sum(int a, int b, double d){
system.out.println(a+b+d);
}
}
public class Lab12{
public static void main(String as[]){
Arithmetic obj = new Arithmetic();
obj.sum();
obj.sum(10);
obj.sum(10,20);
obj.sum(10,20,30);
obj.sum(50,20,20);
obj.sum(20,10,50, 30);
}
}
```


Problem With Method Overloading:-

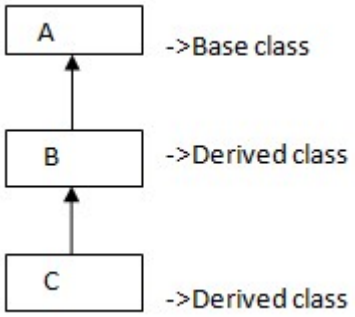
* If you have the requirement to add only one type of data inside the method then for every different number of arguments, you have to write new method or you can modifier the existing method.
* In JDK 1.5 sun has provided variable argument feature by which you can declared only one method that will handle different number of method calls.
* The value you are passing to var args will be passed as array of particular data type.
* The array can contain either zero (0) or more elements.

Example:-

```
void m1(Hai hai) {
system.out.println(“m1() start “);
system.out.println(Hai.x);
hai.x = hai.x+10;
system.out.printl;n(“m1() end “);
}
void m2 (int p)
{
system.out.println(“m2() start “);
system.out.println(p);
p=p+10;
system.out.println(p);
system.out.println(“m2 ends”);
}
}
public class Lab20 {
public static void mian (String as [ ]) {
Hello h =new Hello ();
int p=10;
system.out.println(“Before calling call by value method”);
system.out.println(p);
h.m2(p);
system.out.println(“After calling call by value method”);
system.out.println(p);
Hai hai = new Hai ();
system.out.println(“Before calling call by reference method”);
system.out.println(Hai.x);
system.out.println(“After calling call by reference method”);
system.out.println(Hai.x);
}
}
```


Q->Can I overload the main() method if yes the which main() method will be called by JVM.

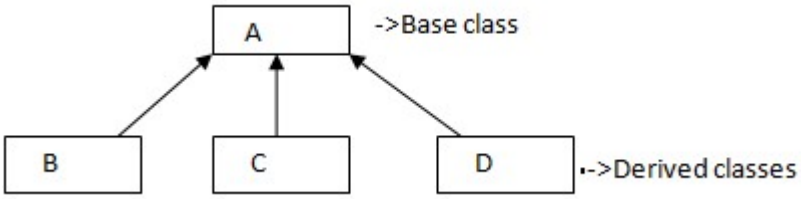
```
public class Lab21 {
public static void main()
{
system.out.println(“main()”);
}
public static void main(int a) {
system.out.println(“main (int a)”);
}
public static void main(String as []) {
system.out.println(“main (String as []”);
main();
main(10);
main(“bca”);
```



3> Hierarchical Inheritance:-

* Here one base can have many direct derived classes but all the derived class have same base class.

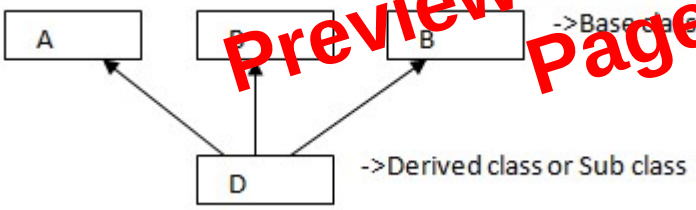
Example:



4> Multiple Inheritance:-

* Here one sub class can have many parent classes but all the parent class have only one sub class.

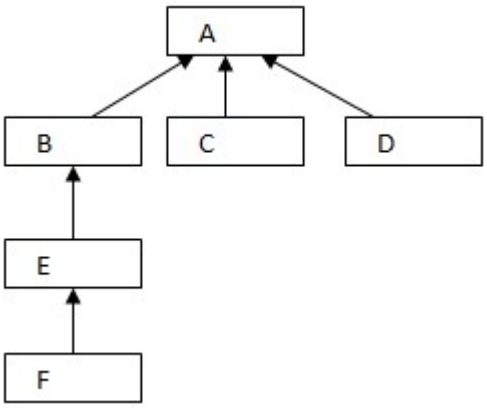
NOTE:- Multiple inheritance not allowed in java.



5> Hybrid Inheritance:-

* Combination of two or more inheritance is called as hybrid.
* Hybrid inheritance is not allowed in java.

Example:



Q->Different Between this and super?

This:-

* It is a reference variable which holds the current class object.
* It can be used for two purpose:-
(i) To access current class member that mean variable and object.
this.a; this.show();
(ii)To access current class constructor this is call default constructor (1 args, 2 args).
->It is instance reference variable can't use in static context.
->call to this must be the 1st statement inside current class constructor.

Super:-

* super is a reference variable which holds the super class and onject.
* It is use for two purpose:-
(i)To access immediate super class member
Super.a --->variable
(ii)super.show() --->method

* To call the super class constructor.
Super() -->default constructor
Super(10) --->1 args ''
Super(10, 20) --->2 args "
* super is a instance variable can't be used is static context.
* call to the super must be the 1st statement inside sub class constructor.

Access Modifiers / Visibility Modifiers

1> Private
2> Default
3> Protected
4> Public

1> Private:-

* Member declared with private can't be accessed outside the class.
* Member declared as default/ protected/ public you can accessed in same class, sub class and non-sub class defined in same package.
Example:-

```
class Superclass {
private int a=10;
int b=20;
protected int c=30;
public int d=40;
void showsuper() {
```

```
class Animal {
void sleeping() {
system.out.println(“Animal is sleeping”);
}
static void running() {
system.out.println(“Animal is running”);
}
void eating() {
system.out.println(“Animal is eating”);
}
}
class Dog extends Animal {
static void running() {
system.out.println(“Dog is running”);
}
void eating() {
system.out.println(“Dog is eating”);
}
void barking () {
system.out.println(“Dog is barking”);
}
}
public class Rtpoly {
public static void main (String as [ ] ) {
Animal a = null;
// a.sleeping() --->not ok
a.new Dog();
a.sleeping();
a.eating();
a.running();
// a.barking();
}
}
```


* In the above example we have assign sub class object (Dog) to super class (Animal) reference Variable, so here we are following the dynamic dispatch.

In the case of eating() method:-

* We are overriding eating() method in Dog class so we are following dynamic dispatch and method overriding for eating() method.
* So for eating method we are achieving the runtime polymorphism.
a.eating() -----> It is contain Dog object. It will call eating of Dog class.

In the case of sleeping () method:-

* We are not overriding sleeping () method in Dog class but same method will be inherited to Dog class.

```
Employee e = new Employee();
e.show();
}
}
```

Important Points:-

* Outer class member can be accessed inside inner class directly.
* Inner class member can't be accessed in outer class directly but same can be accessed with an object.
* Instance inner class can't have static declarations.
* Inner class name can be used inside the outer class but you can't inner class name outside the outer class directly.

Program:-

```
interface I1{
class Hello implements I1 {
class Hai {
}
interface I2 {
}
interface I3 {
}
private class Hai1 implements I1, I2, I3 {
}
protected class Hai2 extends Hai {
class X {
int a =10;
class Y {
system.out.println(a);
}
}
}
}
}
class Lab2 {
public static void main (String as [ ] ) {
Hello h = new Hello();
}
}
```

=> How many class file will be generated for Lab2

1> Hello $ Hai.class
2> Hello $ Hai2.class
3> Hello $ Hai2 $ X $ y.class
4> Hello $ Hai2 $ X.class
5> Hello $ Hai2 $ Y.class
6> Hello $ I2.class
7> Hello $ I3.class

* Anonymous class can be written inside the method or inside the class.
* If want to use any variable inside the Anonymous class than that variable must be declare as final.
* We can write instance block inside the animal's class but we can't write static blocks.
* We can't write the constructor for inside the Anonymous class as we know constructor name is same the class name but your Anonymous class don't have name.

Example:-

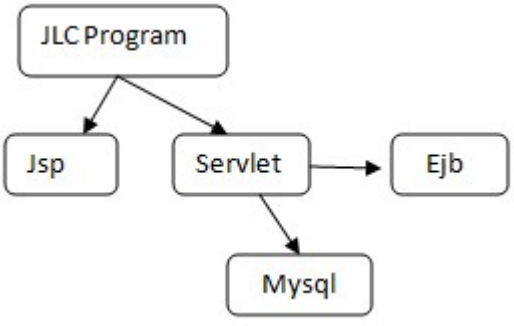
```
abstract class Ball {
abstract void hit () {
class Hello {
Ball b = new Ball () {
}
system.out.println("Instance Block of Anonymous class");
public void hit () {
system.out.println(" You hit);
}
};
void show () {
b.hit();
}
}
class Lab5 {
public static void main (String as [ ]) {
Hello h = new Hello();
h.show();
}
}
```


End Of Oops Concept

Package

Package:

* Package is the collection of classes.
* It is mainly used to organized file of your package.



(2) import static java.lang.math.*;

* in 2nd static import all the static method and static variable will be imported to your program.
* So using static import you can import static member to your program.
* Using to many static imports reduce the readability of your program.

Built-in package:-

* Package which are bundle with java software is called Built-in package.

List of Built-in package are:-

1> Java.lang;
2> Jaba.io;
3> Java.lang.reflect;
4> Java.util;
5> Java.sql;1> Java.lang:-

* Java.lang package is default package which is important in all the java program by default
* You can access all the classes from java.lang.package without importing it.
* List of class available in java.lang.package:-

1> Object
2> Char sequence
3> Byte
4> Integer
5> Short
6> Float
7> Double
8> Boolean
9> Character
10> Long
11> System
12> Process
13> Class
14> Class Loader
15> Runtime
16> Comparable
17> Comparator
18> String
19> String Buffer
20> String Builder

1> Object:-

* Object class is the default super class of all the java programs.
* You can access all the member of object class in your class as member of current class without extending it.

```
public static void main (String as [ ]) {
Hello h = new Hello();
system.out.println(“getClass()”);
system.out.println(h.getClass());
system.out.println(h.getclass().getName());
Hai h1 = new Hai ();
system.out.println(h1.getClass());
system.out.println(h1.getclass().getName());
Hello h2 = new Hai();
system.out.println(h2.getClass());
system.out.println(h2.getClass().getName());
system.out.println(“ hashCode”);
system.out.println(h1.hashCode());
system.out.println(hai.hashCode());
system.out.println(“tostring”);
system.out.println(h);
system.out.println(h.tostring());
system.out.println(Hai);
system.out.println(Hai.tostring ());
system.out.println(h1.tostring());
system.out.println(h1);
}
}
```


getClass():-

* It is the final method available in java,lang object class.
* You cannot overriding the getClass() method in your class.
* getClass() method will return the class name of invoking object.

hashCode:-

* It will return some hashCode integer number generated by JVM.
* whenever you are creating the object one hashCode will be generated by JVM forever object.
* This hashCode will be used by JVM to search the object when you have many object.
* If you want to generated your own hashCode then you can override the hashCode method in your class and write your own hashcode generation algorithm.

toString:-

* The return type of tostring() method is string.
* whenever you are printing the reference variable then JVM call the toString() method on the object

Example of using equals() method;-

```
package com.bca.lab2;
class Student {
int sid;
```

->prepare the list of unused obhect
->Hand over this list to gc.

Garbage Collector(GC):-

->It will call reufinalization() method and all unused object.
System.runfinalization();
rt.runfinalization();

Clean the memory:

runfinalize() method.
* in finalize() method you can write the code release the connection.
* call finalize() method.

Process of Garbage Collection:

=>Responsibility of JVM:-
* Detecting the unused object.
* Prepare the list of unused object.
* Whenever memory shortage problem will occure invoke runfinalzation() method.
* Call the gc() mehtod.
System.gc();
rt.gc();
runtime rt.gc();

=>Responsible of gc() :-
* gc () method is responsible to clean the memory.

=>Responsibility of runfinalization():-
* It will call finalize() method on all unused object .
* You can have two version of finalize() method one from object class and 2nd you can override finalize() method in your class and write the code for resource cleanup.

Example:-

```
class Hello {
public void finalize() {
system.out.println(“Hello – finalize()”);
}
}
class Hai {
void m1() {
system.out.println(“starts- m1()”);
Hello h = new Hello ();
Hello h1 = new Hello ();
Hello h2 = new Hello ();
Hello h3 = new Hello ();
```

This method you can use to set the properties.

7> Public static string getProperties (string_prop_value):-

This method will return the property value of specified property name.

8> setProperty (string_prop_name, string_prop_value):-

Using this method you can get the property value of specified property name.

Runtime class:-

* Runtime is a final class available in java.lang package.
* You can't create the object explicitly of runtime class.
* In runtime class all member are defined as now static.
* You can access the object of runtime class as follows:-

Runtime rt = Runtime.getRuntime();

* Runtime class default constructor is private.

Math class:-

* Math is final class available in java.lang. Package.
* All the member of math class are static.
* All the member of math class is strict.

Strict FP:-

* Strict FP stands for strict floating point.
* By default floating point calculation is based on operating system standards.
* If you are following as based calculations then you may get variation in result from one as to another.
* To get unified result among all the as you can use strict FP keyword.
* Strict FP in process to use IEEE standards for floating point calculations.

GOF(Gang of Fore) Design Pattern:-

* GOF design pattern is also called as code java design pattern.

List of core java design pattern:-

(i) singleton (v) command
(ii) factory (vi) decorator
(iii) obj sever (vii)Proxy
(iv) facode (viii) composite etc

(i) singleton:-

* Singleton design pattern ensure that only one object will be created for a class.
* Example of access which is implementing singleton design-pattern:-

* Zero or more (many).

Q->Can you define finally block between two catch block?

* No it should be last block of statement.

Q->How can be stop execution of finally block?

* using System.exit() method inside try block on catch block.
* finally block is mainly used to write resource cleanup code.
* we can write the resource cleanup code inside the finalized method also but it is recommended to write resource cleanup code in finally block for better performance.

Q-> Different between finalize and finally?

1> finalize():-

```
class Test {
connection con = null;
void get_con() {
con=om.getconnection(“Drive”);
}
public void finalize() {
conclose();
}
}
```


* Connection will be close vay-2 late
* You can handle limited no of connection.

2> finally:-

```
class Test {
connection con = null;
void get_con() {
try {
con = om.getconnection(“Drive”);
}
catch(Exception e) { }
finally{
conclose();
}
}
```

* Connection will be close immediately.
* You can handle any no. of request.