**Java Inheritance and Polymorphism**  
  
Java OOPS- Object Oriented Programming System.  
  
Four fundamentals OOPS:  
  
i) Inheritance  
  
ii) Polymorphism  
  
iii) Abstraction  
  
iv) Encapsulation  
-----------------------------  
**i) Inheritance**  
   
> It is a process of Inheriting(reusing) the class members(Variables and Methods) from one class to another.  
  
> Non-static class members only can be Inherited.  
  
> The class where the class members are getting Inherited is called as Super class/Parent class/Base class.  
  
> The class to which the class members are getting Inherited is called as Sub class/Child class/derived class.  
  
> The Inheritance between Super class and Sub class is achieved using "extends" keyword.

Class A - Parent class

Class B- Child Class

Public class B extends Class A{

}

--------------------------------------------------------  
**How to create Static class members?**  
Using static Non-access modifier.  
  
**How to access Static class members?**  
Using Class Name we access Static class members.  
  
**How to access Non static class members?**  
Using Object/Instance we can access Non static class members.  
------------------------------------------------------  
Example for accessing Static and Non-static Class members:  
   
package package1;  
  
public class Class1 {  
//Declare Static variables  
static int a =10, b=20;  
//Declare Non-static variables  
int c=30, d=40;  
//Create Static a method with returning a value  
public static int add(){  
int result = a+b;  
return result;      
}  
//Create Static a method without returning any value  
public static void multiply(){  
System.out.println(a\*b);  
}  
//Create a Non static method with returning a value  
public int add2(){  
int res = c + d;  
return res;  
}  
//Create a Non static method without returning any value   
public void multiply2(){  
System.out.println(c\*d);      
}  
public static void main (String [] args){  
//Access Static Class Members(Using Class Name)  
int x = Class1.add();  
System.out.println(x);//30  
System.out.println(Class1.a);//10  
Class1.multiply();//200  
  
//Access Non static class members(Using Object)   
Class1 obj = new Class1();  
int y = obj.add2();  
System.out.println(y);//70  
System.out.println(obj.c);//30  
obj.multiply2();//1200  
}  
}  
-----------------------------  
Three types Inheritance  
  
**1) Single Inheritance**  
  
**Example:**  
public Class ClassB extends Class A {  
.  
}  
-----------------------  
**2) Multi level Inheritance**  
  
**Example:**  
public Class ClassB extends ClassA {  
  
public Class ClassC extends ClassB {

C Child2 Class-🡪B(child1 class) -🡪 A (Base class)  
----------------------------------  
**iii) Multiple Inheritance (\* Java doesn't support)**  
Example:  
  
public class ClassB extends ClassA {  
public class ClassB extends ClassD {

.  
}  
}  
--------------------------------------------  
Example for Inheritance:  
   
**Class 1:**  
public class ClassA {  
int a =10;  
int b =20;  
public void add(){  
System.out.println(a+b);  
}  
  
public static void main(String[] args) {  
ClassA objA = new ClassA();  
System.out.println(objA.a);//10  
objA.add();//30  
}  
}  
----------------------------  
**Class 2:**  
public class ClassB extends ClassA{  
int a =100;  
int b =200;  
public void add(){  
System.out.println(a+b);      
}  
public static void main(String[] args) {  
ClassB objB = new ClassB();  
objB.add();//300  
System.out.println(objB.a);//100  
}  
}  
----------------------------  
**Class 3:**  
public class ClassC extends ClassB {  
int a =1;  
int b=2;  
public void add(){  
System.out.println(a+b);  
}  
  
public static void main(String[] args) {  
ClassC objC = new ClassC();  
System.out.println(objC.a);  
objC.add();  
}  
}  
--------------------------------------  
Example 2:  
package package1;  
  
public class ClassX {  
protected int a =10;  
protected int b =20;  
  
protected void add(){  
System.out.println(a+b);  
}  
public static void main(String[] args) {  
ClassX objX = new ClassX();  
System.out.println(objX.a);  
objX.add();  
}  
}  
---------------------------------  
package package2;  
  
import package1.ClassX;  
  
public class ClassZ  extends ClassX{  
public static void main(String[] args) {  
ClassZ objZ = new ClassZ();  
objZ.add();  
System.out.println(objZ.a);  
}  
}  
-----------------------------  
**ii) Polymorphism**  
   
Existence of Object behavior in many forms  
  
There are two types of Polymorphism  
  
1) Compile Time Polymorphism / Method Overloading  
  
2) Run Time Polymorphism or Method Overriding  
-----------------------------------  
1) Compile Time Polymorphism / Method Overloading  
   
If two are more methods having same name in the same class but they differ in following ways.  
  
a) Number of Arguments  
  
b) DataType of Arguments  
-----------------------------  
Example for Method OverLoading:  
   
public class MethodOverLoading {  
public void add(int a, int b){  
System.out.println(a+b);      
}  
  
public void add(int a, int b, int c){  
System.out.println(a+b+c);      
}  
  
public void add(double a, double b){  
System.out.println(a+b);      
}  
  
public void add(double a, double b, double c){  
System.out.println(a+b+c);      
}  
public static void main(String[] args) {  
MethodOverLoading obj = new MethodOverLoading();  
obj.add(10, 20);  
obj.add(10, 20, 30);  
obj.add(1.234, 2.456);  
obj.add(1.234, 2.456, 3.567);  
}  
}  
-----------------------------------------  
2) Run Time Polymorphism or Method Overriding  
   
If two are more methods with same name available in the Super class and Sub class.  
  
Example for Method OverRiding:  
**Class 1:**  
  
public class ClassNew1 {  
public void myMethod(){  
System.out.println("Selenium for Test Automation");  
}  
public static void main(String[] args) {  
ClassNew1 obj = new ClassNew1();  
obj.myMethod();  
}  
}  
-------------------------  
**Class 2:**  
   
public class ClassNew2 extends ClassNew1{  
public void myMethod(){  
System.out.println("UFT for Test Automation");  
}  
public static void main(String[] args) {  
ClassNew2 obj = new ClassNew2();  
obj.myMethod();//UFT for Test Automation  
  
ClassNew1 obj2= new ClassNew1();  
obj2.myMethod();//Selenium for Test Automation  
}  
}