Inheritance in JAVA

JAVA support the concept of reusability as C++.The mechanism of driving a new class from existing class in inheritance.In JAVA a class that is inherited is called as a superclass.The class that does the inheriting is called as subclass.The subclass is specialized version of superclass.It inherits all the data members of superclass and also adds its own features to it.

In JAVA a class can be inherited as shown below

class subclassname extends superclassname

{

//declaration of data members

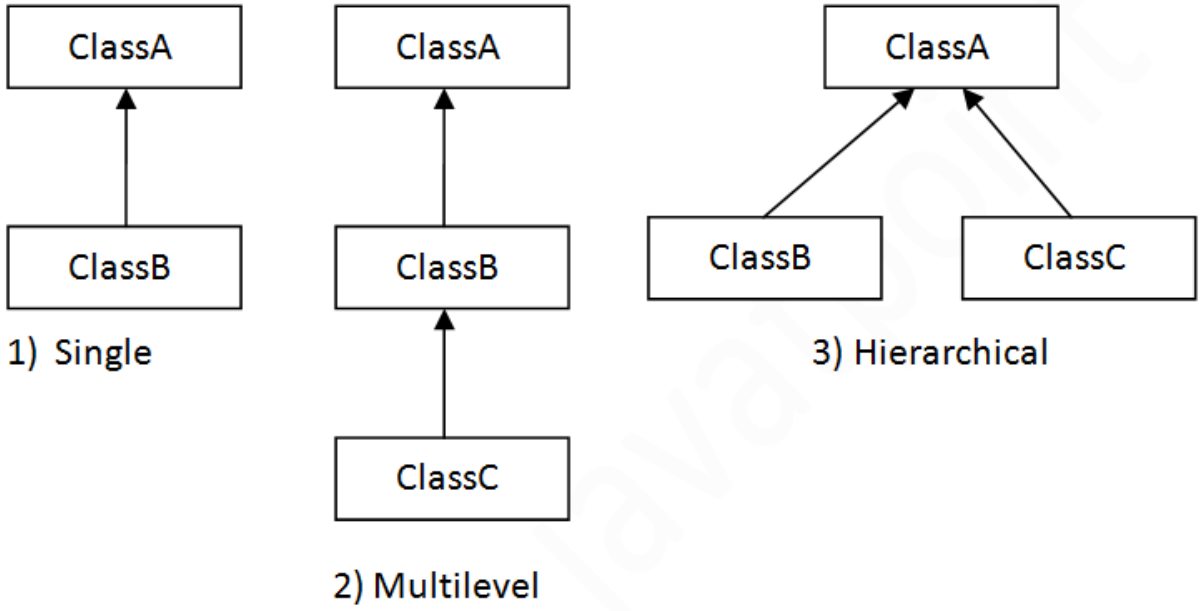
}

The extends is a keyword which is used to show that a subclass is extended from a superclass.

Types of Inheritance

1. Single Inheritance
2. Multilevel Inheritance
3. Hierarchical Inheritance
4. Multiple Inheritance **//JAVA does not support need to use Interface extends can be used with only one class**
5. Hybrid Inheritance **//Need to use Interface**

Types of inheritance supported by JAVA



e.g. Program to explain the concept of Single level Inheritance

```
class Room
{
Int length,breadth;
Room(int a,int b)
{
length=a;
breadth=b;
}
int area()
{
return(length*breadth);
}
}
class Bed extends Room
{
int height;
Bed(int a,int b,int c)
{
super(a,b);
height=c;
```

```
}
int volume()
{
retrun(length*bredth*height);
}
}
class test
{
public static void main(String args[])
{
Bed b=new Bed(10,20,40);
System.out.println("Area is " + b.area());
System.out.println("Volume  is "+ b.volume());
}
}
```

Super keyword gives access to the constructors,methods and variables of its superclass.The super keyword works with inheritance.

Program to explain Hierarchical Inheritance

```
import java.io.*;
class A
{
int rno;
void setrno(int a)
{
rno=a;
}
void show()
{
System.out.println("rno is "+rno);
}
}
class B extends A
{
int s1;
void setmarks(int x)
{
s1=x;
}
void show1()
{
```

```
System.out.println("subject marks "+s1);
}
}
class C extends A
{
int l1;
void getlab(int y)
{
l1=y;
}
void show2()
{
System.out.println("lab marks are "+l1);
}
}
class hirarc
{
public static void main(String args[])
{
B b=new B();
b.setrno(83);
b.setmarks(40);
C c=new C();
c.getlab(50);
b.show();
b.show1();
c.show2();
}
}
```

Program to explain Hybrid Inheritance in JAVA

```
import java.io.*;
class A
{
int rno;
void getrno(int z)
{rno=z;
}
void putrno()
```

```
{
System.out.println(" Roll No "+rno);
}
}
class B extends A
{
float m1,m2;
void getmarks(float a,float b)
{
m1=a;
m2=b;
}
void putmarks()
{
System.out.println(" The marks are ");
System.out.println(" Marks1 "+m1);
System.out.println(" Marks 2"+m2);
}
}

interface C
{
flaot sportmarks = 5.0;
void putsports();
}

class D extends B implements C
{
float total;
public void putsports()
{
System.out.println(" Sprots marks are "+sportmarks);
}

void show()
{
total = m1+m2+sportmarks;
putrno();
putmarks();
putsprots();
System.out.println(" Total is "+total);
}
```

```
}

class hybrid
{
public static void main(String args[])
{

D d=new D();
d.getrno(10);
d.getmarks(78,88);
d.show();
}

}
```

JAVA visibility modifiers (Access specifies)

The following chart explains the access specifiers supported by JAVA

Access	Public	Protected	Default	Private
From the same class	Yes	Yes	Yes	Yes
From any class in the same package	Yes	Yes	Yes	No
From any class outside the package	Yes	No	No	No
From a sub - class in the same package	Yes	Yes	Yes	No
From a sub - class outside the same package	Yes	Yes	Yes	No