**Inheritance in Java**

Inheritance:

1. Inheritance in Java is used to add additional features to an existing class.
2. Inheritance is used to extend the present class by adding some properties to it.
3. Inheritance is used to reuse the present tried and tested code so that you may not have to written them and compile them again.

Inheritance in Java is implemented by using the Extends Java keyword.

**Scenario1: Child class reference and Child class object** – This will allow to access all the methods of base class(parent class) and child class.

Example: ChildClass obj1=new ChildClass();

**Scenario2: Base Class reference and Base class object** – This will allow to access all the methods of base class only.

Example: Baseclass obj1=new Baseclass();

**Scenario3: Base class reference and child class object** – This will allow to accesss all the methods of base class and not child class.

Example: Baseclass obj1=new ChildClass();

**Scenario 4: Child class reference and BaseClass Object** – This will be invalid scenario because child cannot hold parent

class A

{

    int i;

    void methodOne()

    {

        System.out.println("From methodOne");

    }

}

class B extends A

{

    int j;

    void methodTwo()

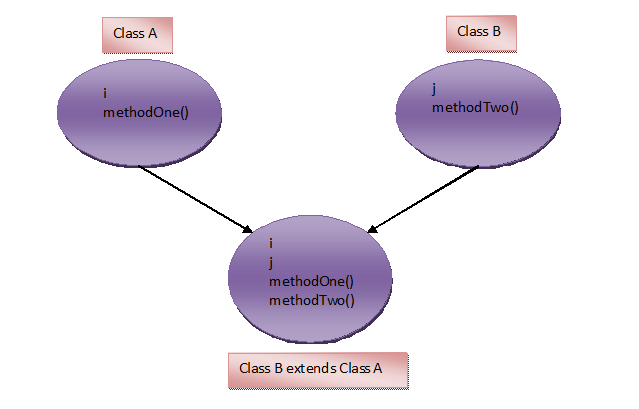
    {

        System.out.println("From methodTwo");

    }

}

Here, **Class A** has two members – **‘i’** and **‘methodOne()’**. **Class B** also has two members – **‘j’** and **‘methodTwo()’**. **Class B** is extending **Class A**. Therefore, members of **class A** are inherited to **Class B**. Now, **Class B** will have two additional members inherited from **class A** along with its members. Have a look at the below diagram,



**Points to remember:**

1)Constructors, SIB – Static Initialization Block and IIB – Instance Initialization Block of super class will not be ineriting to its subclass. But they are executed while creating an object of sub class. For example.

class A

{

    int i;

    static

    {

        System.out.println("Class A SIB");

    }

    {

        System.out.println("Class A IIB");

    }

    A()

    {

        System.out.println("Class A Constructor");

    }

}

class B extends A

{

    int j;

}

class MainClass

{

    public static void main(String[] args)

    {

        B b = new B();

    }

}

**Output :**  
Class A SIB  
Class A IIB  
Class A Constructor

2)Try to compile following program.

class A

{

    int i;

    A(int i)

    {

        System.out.println("Class A Constructor");

    }

}

class B extends A

{

    int j;

}

You will get a compile time error saying implicit default constructor A() is undefined for Class A. Compiler will force you to write constructor in Class B. Because, we are not defining constructor for Class B. So, compiler will be providing default constructor. In that default constructor, first statement is super() – it is a calling statement to default constructor of Class A. But it is not defined in Class A. Therefore you will get a compile time error. To avoid this error, write the constructor for sub class. From that constructor call super class constructor explicitly. See the below code.

class A

{

    int i;

    A(int i)

    {

        System.out.println("Class A Constructor");

    }

}

class B extends A

{

    int j;

    public B()

    {

        super(10);     //Explicitly Calling Class A constructor

        System.out.println("Class B Constructor");

    }

3)By default, every class is a sub class of java.lang.Object class. So, every class in java has properties inherited from Object class. Look at the below code,

class A

{

     //some satements

}

Compiler will treat the above code as,

class A extends Object

{

     //some satements

}

4)Any class can not extend itself i.e

class A extends A

{

     //It gives compile time error

}

5)We can call super class constructor explicitly through super() calling statement from sub class constructor and we can call other constructors of the same class through this() calling statement but, we can’t call sub class constructor from super class constructor.

class A

{

     A()

     {

          //B();      There is no statement in java to call subclass constructor

          System.out.println("Class A Constructor");

     }

}

class B extends A

{

     B()

     {

          super();        // calling statement to super class constructor

          System.out.println("Class B Constructor");

     }

}

**Effect of private, default, protected and public keyword on inheritance in java:**

**Private :**Private members can not be inherited to sub class.

**Default** : Default members can be inherited to sub class within package.

**protected** : protected members can be inherited to any sub class but usage of protected member is limited within package.

**public** : public members are inherited to all sub classes.

Types of Inheritance in Java



**1) Single Inheritance :** One class is extended by only one class.

**2) Multilevel Inheritance :** One class is extended by a class and that class is extended by another class thus forming chain of inheritance.

**3) Hierarchical Inheritance :** One class is extended by many class.

**4) Hybrid Inheritance :** It is a combination of above types of inheritance.

There exist one more type of inheritance – **Multiple Inheritance.**

**5) Multiple Inheritance :** One class extends more than one class.