**Inheritance in Java**

Inheritance is an important pillar of OOP (Object Oriented Programming). It is the mechanism in java by which one class is allow to inherit the features (fields and methods) of another class.  
**Important terminology:**

* **Super Class:** The class whose features are inherited is known as super class (or a base class or a parent class).
* **Sub Class:** The class that inherits the other class is known as sub class (or a derived class, extended class, or child class). The subclass can add its own fields and methods in addition to the super class fields and methods.
* **Reusability:** Inheritance supports the concept of “reusability”, i.e. when we want to create a new class and there is already a class that includes some of the code that we want, we can derive our new class from the existing class. By doing this, we are reusing the fields and methods of the existing class.

**How to use inheritance in Java**

The keyword used for inheritance is **extends**. Syntax :

|  |
| --- |
| class derived-class extends base-class |
| { |
| //methods and fields |
| } |

**Example:** In below example of inheritance, class Bicycle is a base class, class MountainBike is a derived class which extends Bicycle class and class Test is a driver class to run program.

//Java program to illustrate the

// concept of inheritance

// base class

class Bicycle

{

    // the Bicycle class has two fields

    public int gear;

    public int speed;

    // the Bicycle class has one constructor

    public Bicycle(int gear, int speed)

    {

        this.gear = gear;

        this.speed = speed;

    }

// the Bicycle class has three methods

    public void applyBrake(int decrement)

    {

        speed -= decrement;

    }

    public void speedUp(int increment)

    {

        speed += increment;

    }

    // toString() method to print info of Bicycle

    public String toString()

    {

        return("No of gears are "+gear

                +"\n"

                + "speed of bicycle is "+speed);

    }

}

// derived class

class MountainBike extends Bicycle

{

    // the MountainBike subclass adds one more field

    public int seatHeight;

    // the MountainBike subclass has one constructor

    public MountainBike(int gear,int speed,

                        int startHeight)

    {

        // invoking base-class(Bicycle) constructor

        super(gear, speed);

        seatHeight = startHeight;

    }

    // the MountainBike subclass adds one more method

    public void setHeight(int newValue)

    {

        seatHeight = newValue;

    }

    // overriding toString() method

    // of Bicycle to print more info

    @Override

    public String toString()

    {

        return (super.toString()+

                "\nseat height is "+seatHeight);

    }

}

// driver class

public class Test

{

    public static void main(String args[])

    {

        MountainBike mb = new MountainBike(3, 100, 25);

        System.out.println(mb.toString());

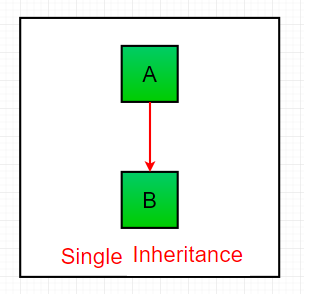
    }

}

**Types of Inheritance in Java**

Below are the different types of inheritance which is supported by Java.

1. **Single Inheritance :** In single inheritance, subclasses inherit the features of one superclass. In image below, the class A serves as a base class for the derived class B.

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/inheritance1.png)

//Java program to illustrate the

// concept of single inheritance

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class one

{

    public void print\_geek()

    {

        System.out.println("Geeks");

    }

}

class two extends one

{

    public void print\_for()

    {

        System.out.println("for");

    }

}

// Driver class

public class Main

{

    public static void main(String[] args)

    {

        two g = new two();

        g.print\_geek();

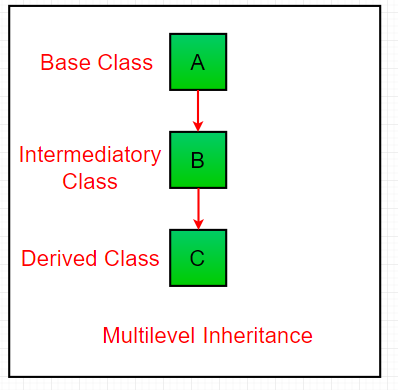
        g.print\_for();

        g.print\_geek();

    }

}

**Multilevel Inheritance :** In Multilevel Inheritance, a derived class will be inheriting a base class and as well as the derived class also act as the base class to other class. In below image, the class A serves as a base class for the derived class B, which in turn serves as a base class for the derived class C. In Java, a class cannot directly access the [grandparent’s members](https://www.geeksforgeeks.org/g-fact-91/).

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/inheritance3.png)

// Java program to illustrate the

// concept of Multilevel inheritance

import java.util.\*;

import java.lang.\*;

import java.io.\*;

class one

{

    public void print\_geek()

    {

        System.out.println("Geeks");

    }

}

class two extends one

{

    public void print\_for()

    {

        System.out.println("for");

    }

}

class three extends two

{

    public void print\_geek()

    {

        System.out.println("Geeks");

    }

}

// Drived class

public class Main

{

    public static void main(String[] args)

    {

        three g = new three();

        g.print\_geek();

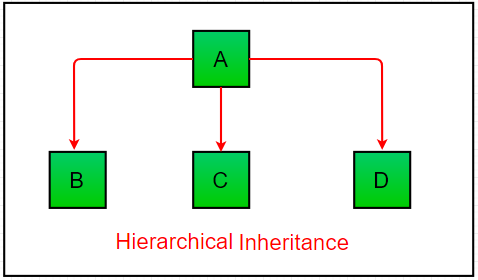
        g.print\_for();

        g.print\_geek();

    }

}

3. **Hierarchical Inheritance :** In Hierarchical Inheritance, one class serves as a superclass (base class) for more than one sub class. In below image, the class A serves as a base class for the derived class B,C and D.

[](https://cdncontribute.geeksforgeeks.org/wp-content/uploads/inheritance4.png)

// Java program to illustrate the

// concept of Multiple inheritance

import java.util.\*;

import java.lang.\*;

import java.io.\*;

interface one

{

    public void print\_geek();

}

interface two

{

    public void print\_for();

}

interface three extends one,two

{

    public void print\_geek();

}

class child implements three

{

    @Override

    public void print\_geek() {

        System.out.println("Geeks");

    }

    public void print\_for()

    {

        System.out.println("for");

    }

}

// Drived class

public class Main

{

    public static void main(String[] args)

    {

        child c = new child();

        c.print\_geek();

        c.print\_for();

        c.print\_geek();

    }

}