Inheritance in Java

Inheritance in java is a mechanism in which one object acquires all the properties and behaviors of parent object.

The idea behind inheritance in java is that you can create new classes that are built upon existing classes. When you inherit from an existing class, you can reuse methods and fields of parent class, and you can add new methods and fields also.

Inheritance represents the **IS-A relationship**, also known as parent-child relationship.

**Why use inheritance in java**

1. For Method Overriding (so runtime polymorphism can be achieved).
2. For Code Reusability.

**Syntax of Java Inheritance**

class Subclass-name extends Superclass-name

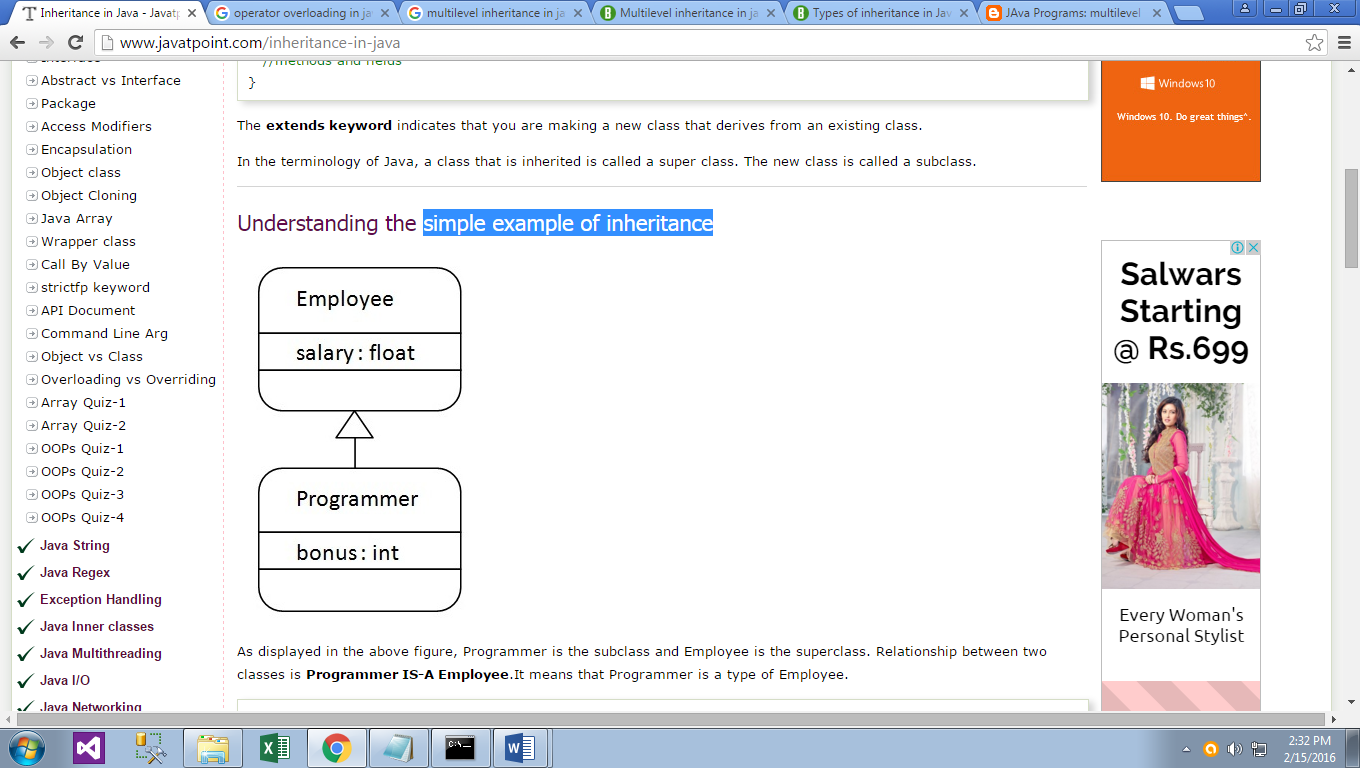
{

//methods and fields }

The **extends** keyword indicates that you are making a new class that derives from an existing class.

In the terminology of Java, a class that is inherited is called a super class. The new class is called a subclass.

Simple example of inheritance



In the above figure, Programmer is the subclass and Employee is the superclass. Relationship between two classes is Programmer IS-A Employee.It means that Programmer is a type of Employee.

class Employee{

float salary=40000;

}

class Programmer extends Employee{

int bonus=10000;

public static void main(String args[]){

Programmer p=new Programmer();

System.out.println("Programmer salary is:"+p.salary);

System.out.println("Bonus of Programmer is:"+p.bonus);

}}

**Output:**

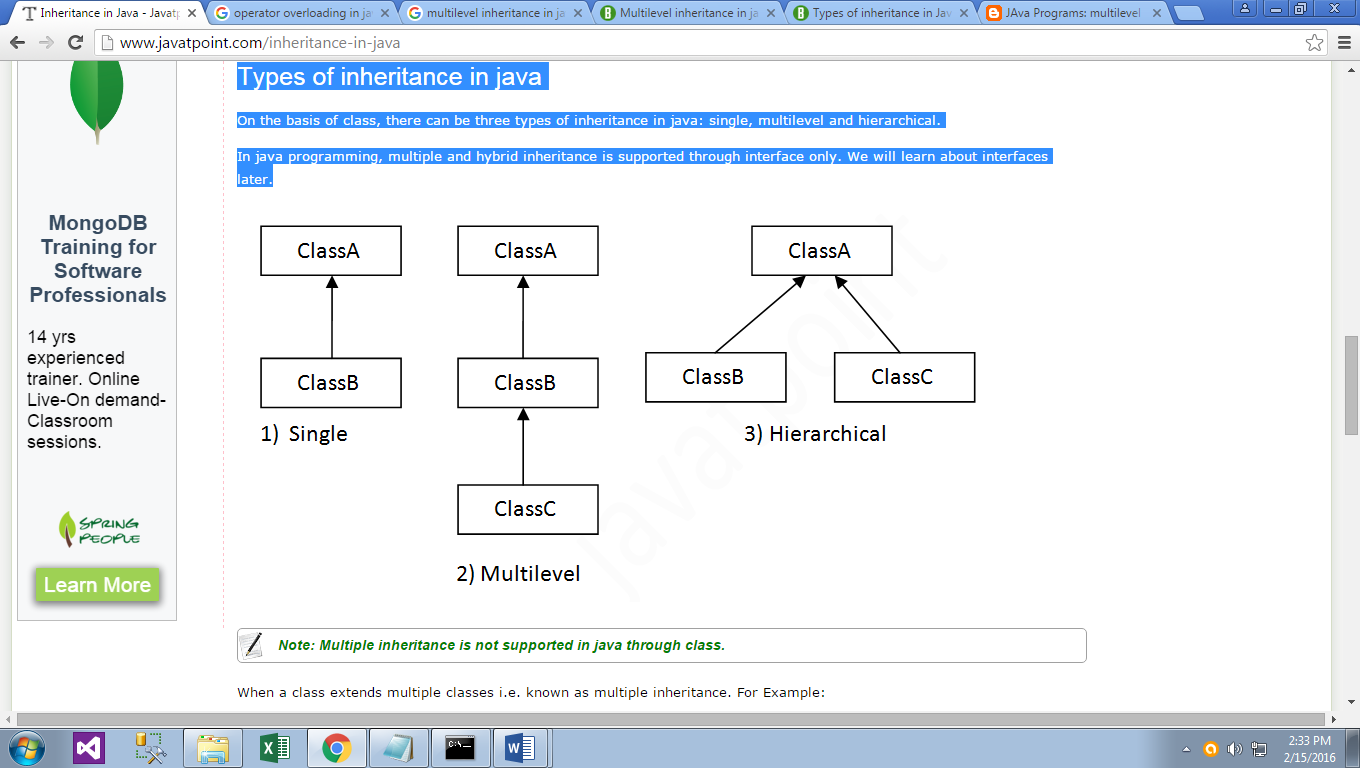
Programmer salary is:40000.0

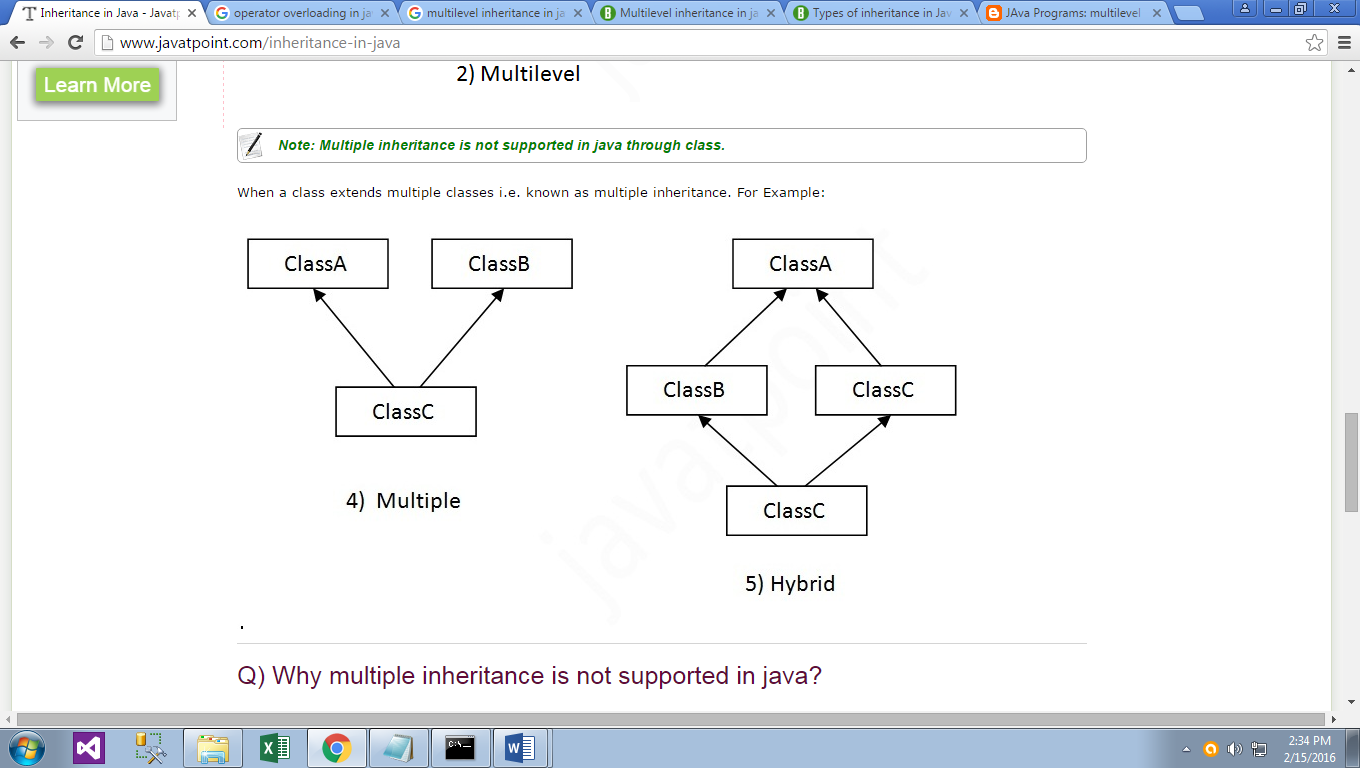
Bonus of programmer is:10000

**Types of inheritance in java**

On the basis of class, there can be three types of inheritance in java: single, multilevel and hierarchical.

In java programming, multiple and hybrid inheritance is supported through interface only. We will learn about interfaces later.





**Multilevel Example:**

class students

{

private int sno;

private String sname;

public void setstud(intno,String name)

{

sno = no;

sname = name;

}

public void putstud()

{

System.out.println("Student No : " + sno);

System.out.println("Student Name : " + sname);

}

}

class marks extends students

{

protected int mark1,mark2;

public void setmarks(int m1,int m2)

{

mark1 = m1;

mark2 = m2;

}

public void putmarks()

{

System.out.println("Mark1 : " + mark1);

System.out.println("Mark2 : " + mark2);

}

}

class finaltot extends marks

{

private int total;

public void calc()

{

total = mark1 + mark2;

}

public void puttotal()

{

System.out.println("Total : " + total);

}

public static void main(String args[])

{

finaltot f = new finaltot();

f.setstud(100,"Nithya");

f.setmarks(78,89);

f.calc();

f.putstud();

f.putmarks();

f.puttotal();

}}

**Output:**

Student No : 100

Student Name :Nithya

Mark1 : 78

Mark2 : 89

Total : 167

**Q) Why multiple inheritance is not supported in java?**

To reduce the complexity and simplify the language, multiple inheritance is not supported in java.

Consider a scenario where A, B and C are three classes. The C class inherits A and B classes. If A and B classes have same method and you call it from child class object, there will be ambiguity to call method of A or B class.

Since compile time errors are better than runtime errors, java renders compile time error if you inherit 2 classes. So whether you have same method or different, there will be compile time error now.

class A{

void msg(){System.out.println("Hello");}

}

class B{

void msg(){System.out.println("Welcome");}

}

class C extends A,B{//suppose if it were

Public Static void main(String args[]){

C obj=new C();

obj.msg();//Now which msg() method would be invoked?

} }

**Output:**

Compile Time Error