Inheritance

Inheritance in Java is an important concept 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. 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.

The class which inherits the properties of other is known as **subclass** (derived class, child class) and the class whose properties are inherited is known as **superclass** (base class, parent 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 superclass fields and methods.

##### **Why use inheritance in java?**

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

##### **How to use inheritance in Java?**

The keyword used for inheritance is **extends**. Following is the syntax of extends keyword.

inheritance syntax

class Super {

.....

.....

}

class Sub extends Super {

.....

.....

}

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);

}

}

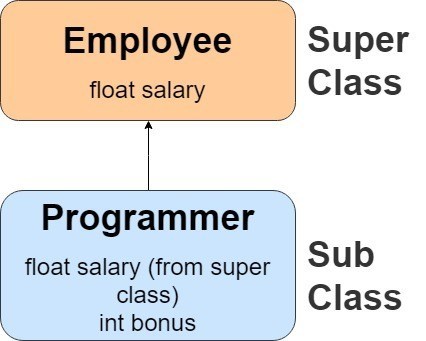
**Output**

Programmer salary is:40000.0  
Bonus of programmer is:10000

**Output**

Programmer salary is:40000.0  
Bonus of programmer is:10000

##### **Program Explanation-**

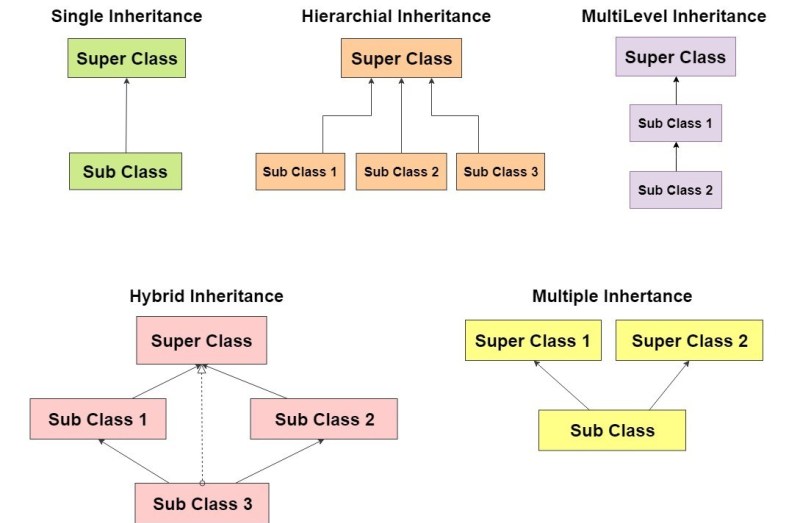


As displayed in the above figure, Programmer is the subclass and Employee is the superclass. Relationship between two classes is **Programmer IS-A Employee**. It means that Programmer is a type of Employee.

##### **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.



##### **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.

Single Inheritance

class Animal

{

void eat()

{

System.out.println("eating...");

}

}

class Dog extends Animal

{

void bark()

{

System.out.println("barking...");

}

}

class TestInheritance

{

public static void main(String args[])

{

Dog d=new Dog();

d.bark();

d.eat();

}

}

**Output**

barking…  
eating…

MultiLevel Inheritance

class Animal {

void eat() {

System.out.println("eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("barking...");

}

}

class BabyDog extends Dog {

void weep() {

System.out.println("weeping...");

}

}

class TestInheritance2 {

public static void main(String args[]) {

BabyDog d = new BabyDog();

d.weep();

d.bark();

d.eat();

}

}

**Output**

weeping…  
barking…  
eating…

Hierarchical Inheritance

class Animal {

void eat() {

System.out.println("eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("barking...");

}

}

class Cat extends Animal {

void meow() {

System.out.println("meowing...");

}

}

class TestInheritance3 {

public static void main(String args[]) {

Cat c = new Cat();

c.meow();

c.eat();

//c.bark();//C.T.Error

}

}

**Output**

meowing…  
eating…