

OOP-4

Inheritance in JAVA

Inheritance in Java is a mechanism in which one object acquires all the properties and behaviors of a 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 the parent class. Moreover, you can add new methods and fields in your current class also.
- Inheritance represents the **IS-A relationship** which is also known as a *parent-child* relationship.

Why use inheritance in java

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

Terms used in Inheritance

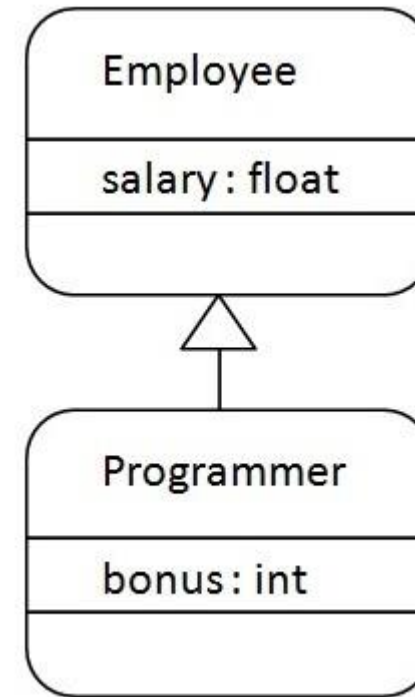
- **Class:** A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
- **Sub Class/Child Class:** Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
- **Super Class/Parent Class:** Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
- **Reusability:** As the name specifies, reusability is a mechanism which facilitates you to reuse the fields and methods of the existing class when you create a new class. You can use the same fields and methods already defined in the previous class.

The syntax of Java Inheritance

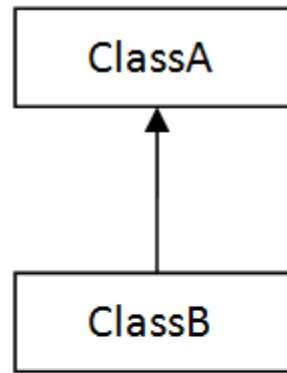
```
class Subclass-name extends Superclass-name
{
    //methods and fields
}
```

Java Inheritance Example

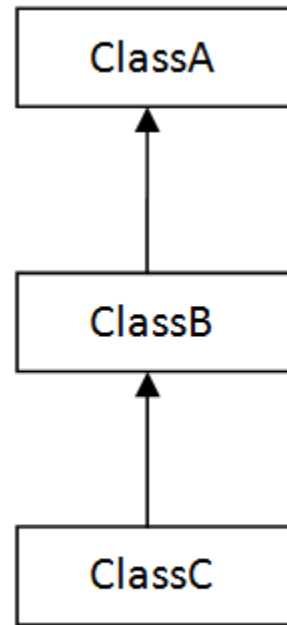
```
class Employee{  
    float salary=40000;  
}  
class Programmer extends Employee{  
    int bonus=10000;  
}  
public static void main(String args[]){  
    Programmer p=new Programmer();  
    System.out.println("Programmer salary is:"+p.salary);  
    System.out.println("Bonus of Programmer is:"+p.bonus);  
}
```



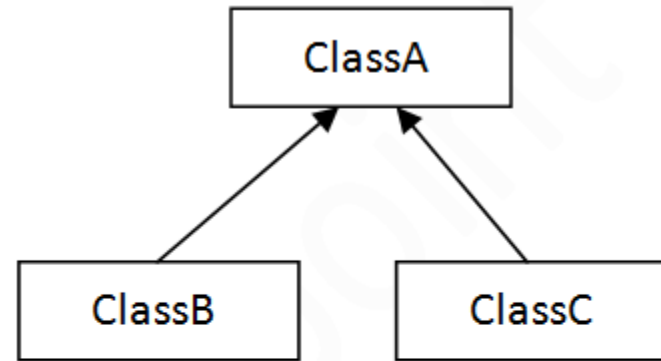
Types of inheritance in java



1) Single

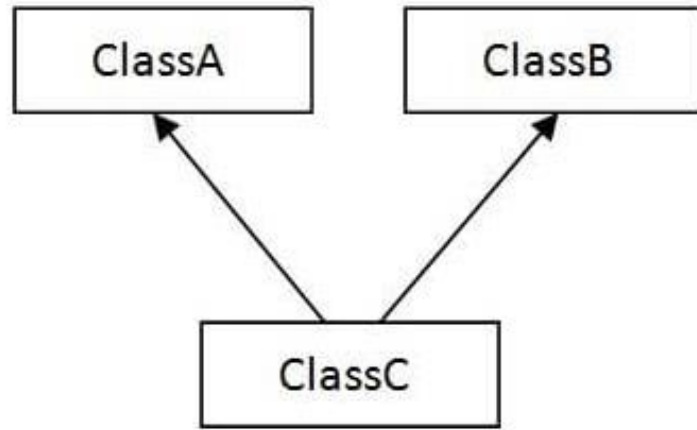


2) Multilevel

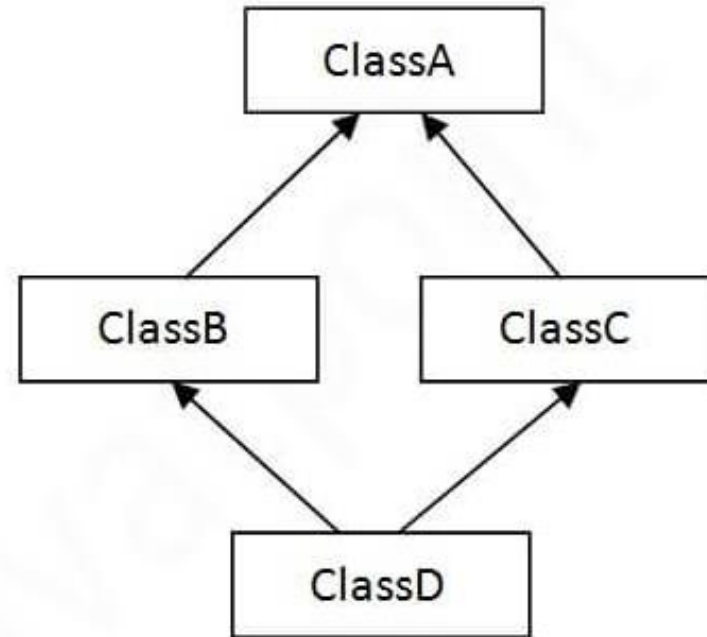


3) Hierarchical

Continued...



4) Multiple



5) Hybrid

Single Inheritance Example

- **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 Example

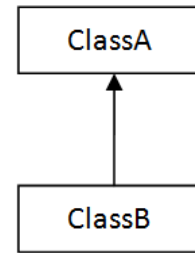
- **class** Animal{
- **void** eat(){System.out.println("eating...");}
- }
- **class** Dog **extends** Animal{
- **void** bark(){System.out.println("barking...");}
- }
- **class** BabyDog **extends** Dog{
- **void** weep(){System.out.println("weeping...");}
- }
- **class** TestInheritance2{
- **public static void** main(String args[]){
- BabyDog d=new BabyDog();
- d.weep();
- d.bark();
- d.eat();
- }}

Output:

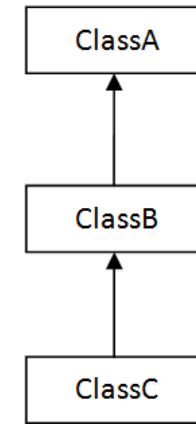
```
weeping...  
barking...  
eating...
```


Hierarchical Inheritance Example

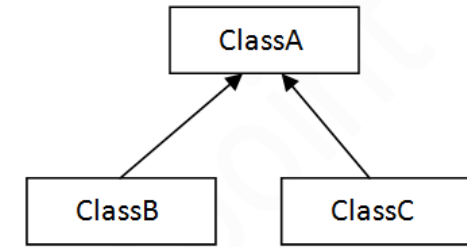
```
class Animal{
    void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
    void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
    void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
    public static void main(String args[]){
        Cat c=new Cat();
        c.meow();
        c.eat();
        //c.bark();//C.T.Error
    }
}
```



1) Single



2) Multilevel



3) Hierarchical

Output:

```
meowing...
eating...
```

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 the same method and you call it from child class object, there will be ambiguity to call the 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.

Continued...

```
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?
    }
}
```