Inheritance in Java

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

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

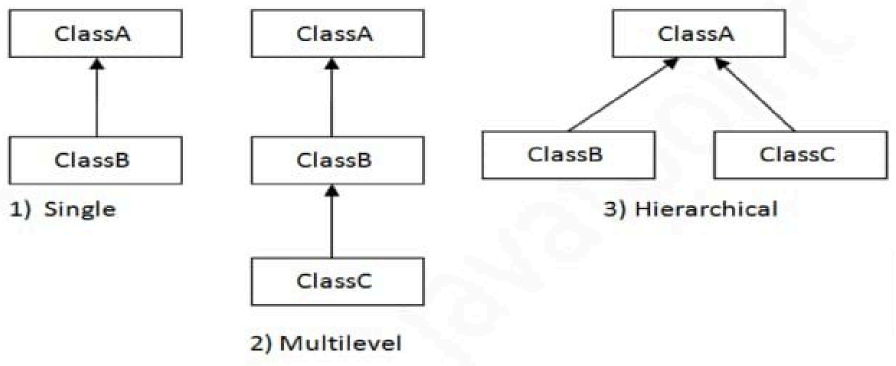
Class: A class is a group of objects which have common properties. It is a template or blueprint from which objects are created.
Sub Class/Child Class: Subclass is a class which inherits the other class. It is also called a derived class, extended class, or child class.
Super Class/Parent Class: Superclass is the class from where a subclass inherits the features. It is also called a base class or a parent class.
Reusability: 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
}
```

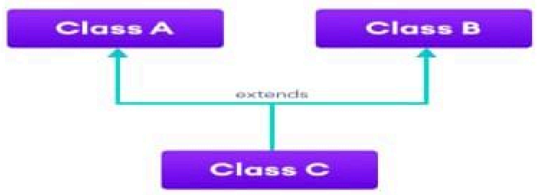
Types of inheritance in java

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

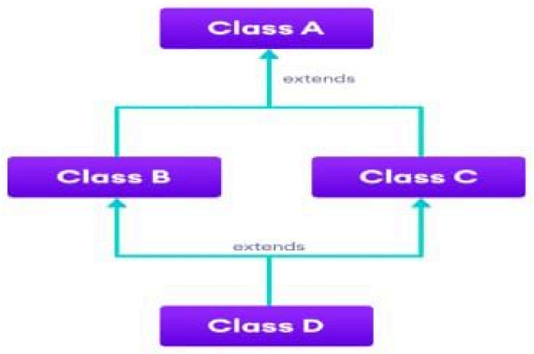
In java programming, multiple and hybrid inheritance is supported through interface only. Multiple inheritance is not supported in Java through class.



Java doesn't support multiple inheritance



Java doesn't support Hybrid Inheritance



Single Inheritance Example

When a class inherits another class, it is known as a *single inheritance*. In the example given below, Dog class inherits the Animal class, so there is the 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[])
{
```

Single Inheritance Example

When a class inherits another class, it is known as a *single inheritance*. In the example given below, Dog class inherits the Animal class, so there is the 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();
}
}
```

Multilevel Inheritance Example

When there is a chain of inheritance, it is known as multilevel inheritance. As you can see in the example given below, BabyDog class inherits the Dog class which again inherits the Animal class, so there is a 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();
}
}
```

Hierarchical Inheritance Example

When two or more classes inherits a single class, it is known as hierarchical inheritance. In the example given below, Dog and Cat classes inherits the Animal class, so there is 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
}}
```

Example 1: