INHERITANCE IN JAVA

- **Inheritance** is an important pillar of OOP(Object-Oriented Programming).
- It is the mechanism in Java by which one class is allowed to inherit the features(fields and methods) of another class.
- In Java, **Inheritance** means creating new classes based on existing ones.
- A class that inherits from another class can reuse the methods and fields of that class. In addition, you can add new fields and methods to your current class as well.

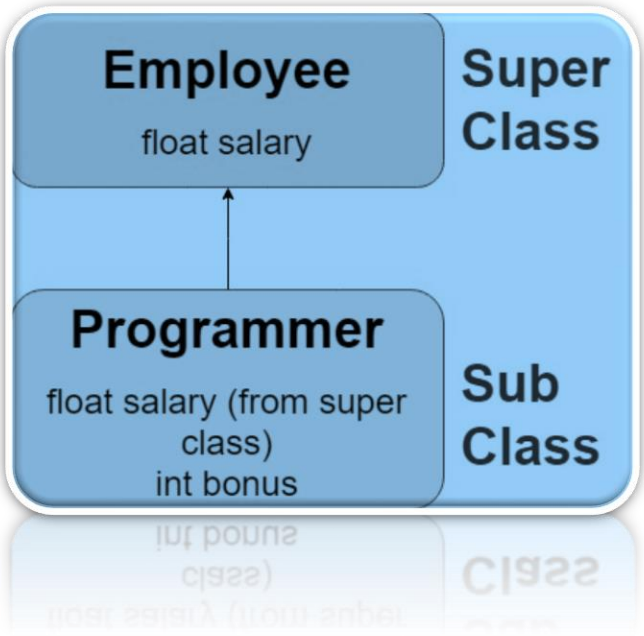
Terms used in Inheritance

- **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.

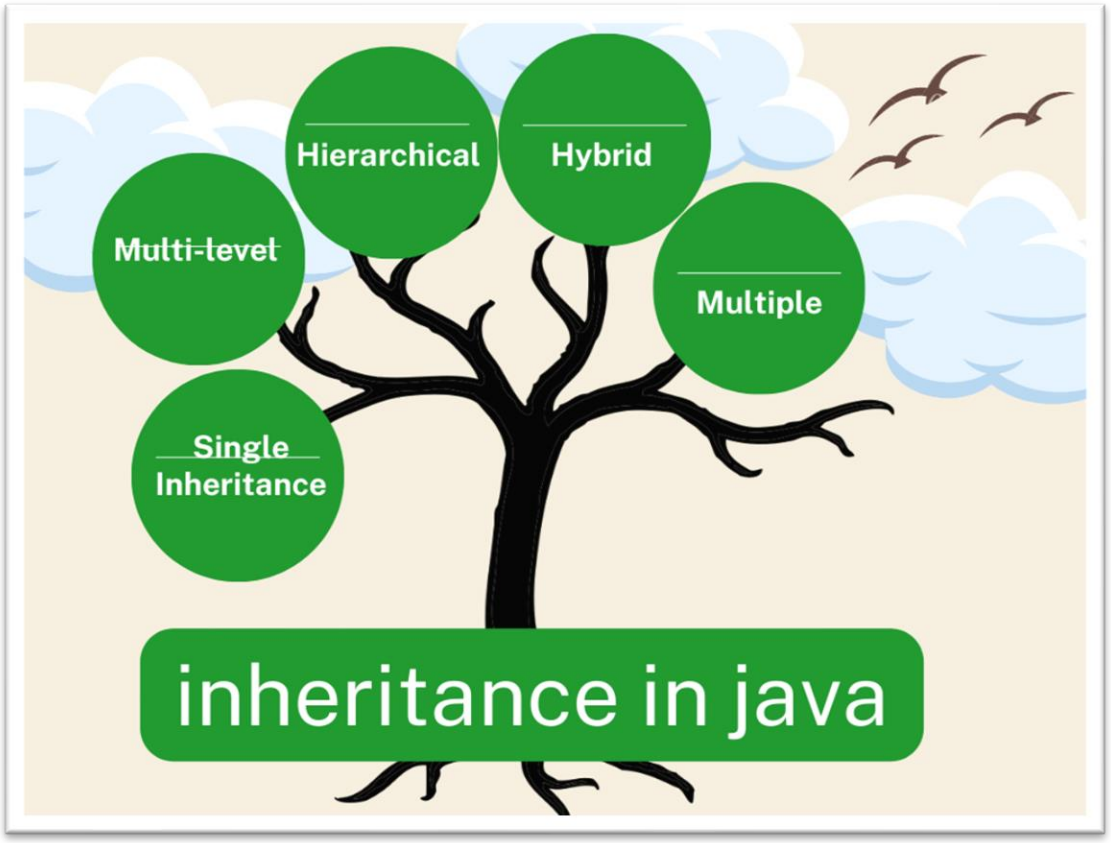
The syntax of Java Inheritance:

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

Example of inheritance:



Types of inheritance :

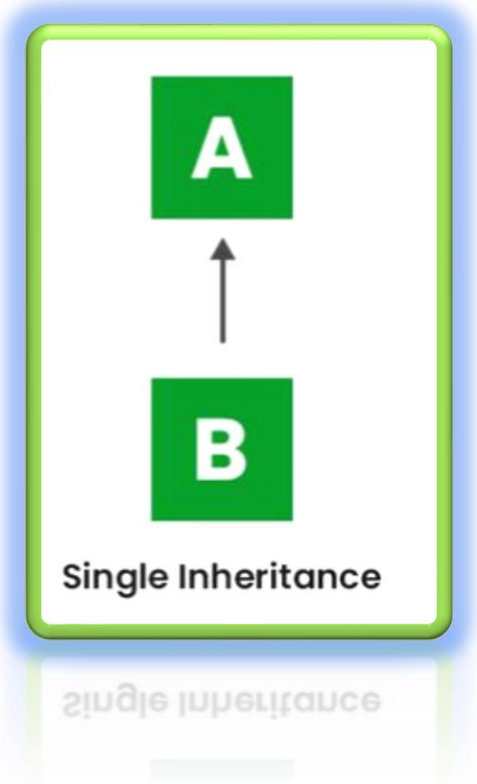


1. **Single Inheritance**
2. **Multilevel Inheritance**
3. **Hierarchical Inheritance**
4. **Multiple Inheritance**
5. **Hybrid Inheritance**

Let's Disscuss one by one:

1. Single Inheritance

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



Example:

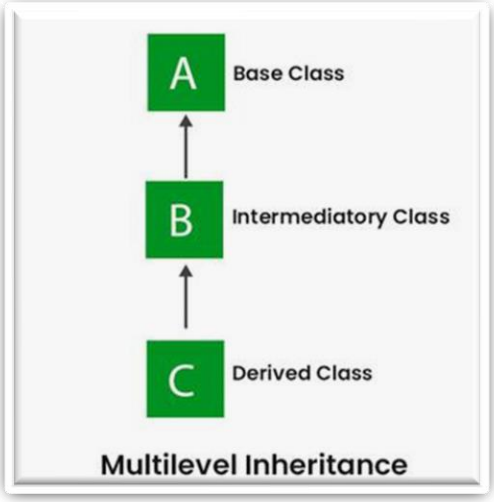
```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class TestInheritance{
public static void main(String args[]){
Dog d=new Dog();
d.bark();
d.eat();
}}
```

Output:

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

2.Multilevel Inheritance

In **Multilevel Inheritance**, a derived class will be inheriting a base class, and as well as the derived class also acts as the base class for other classes. In the below image, 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.



Example:

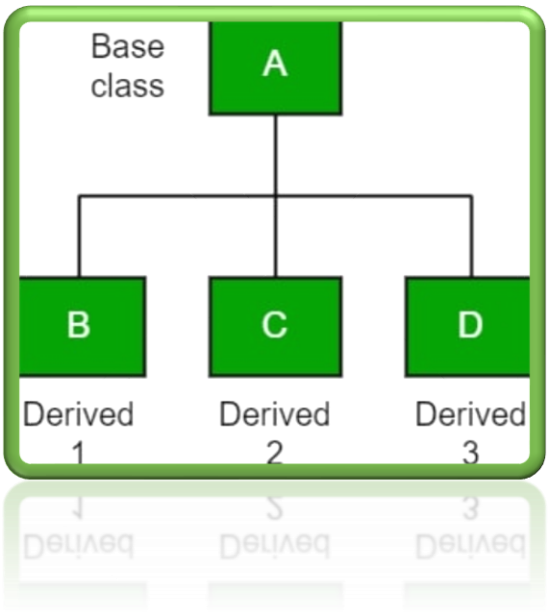
```
class Animal{
void eat(){System.out.print("eating...");}
}
class Dog extends Animal{
void bark(){System.out.print("barking...");
}
}
class BabyDog extends Dog{
void weep(){System.out.print("weeping...");
}
}
class TestInheritance2{
public static void main(String args[]){
BabyDog d=new BabyDog();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

Output:

```
Weeping... barking...
eating...
```

3. Hierarchical Inheritance

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



Example:

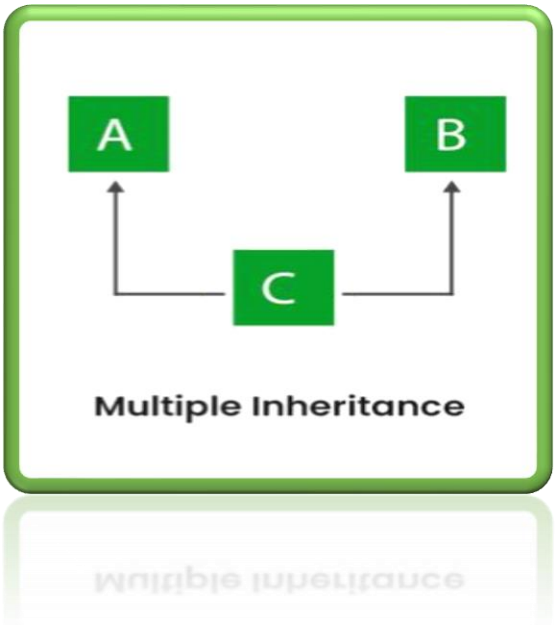
```
class Animal{
void eat(){System.out.println("eating...");}
}
class Dog extends Animal{
void bark(){System.out.println("barking...");}
}
class Cat extends Animal{
void meow(){System.out.println("meowing...");}
}
class TestInheritance3{
public static void main(String args[]){
Cat c=new Cat();
c.meow();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

Output:

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

4.Multiple Inheritance

In Multiple inheritances, one class can have more than one superclass and inherit features from all parent classes. Please note that Java does not support multiple inheritances with classes. In Java, we can achieve multiple inheritances only through Interfaces. In the image below, Class C is derived from interfaces A and B.



Example:

```
interface One {
public void print_tw();
}
interface Two {
public void print_for();
}
interface Three extends One, Two {
public void print_tw();
}
```

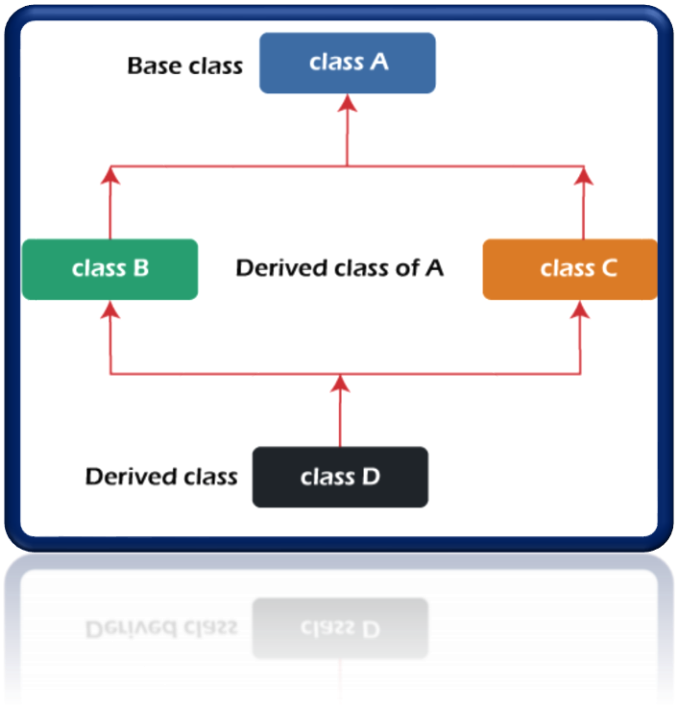
```
interface Three extends One, Two {
public void print_tw();
}
class Child implements Three {
@Override public void print_tw()
{
        System.out.println("Tws");
}
public void print_for() { System.out.println("for"); }
}
// Drived class
public class Main {
public static void main(String[] args)
{
Child c = new Child();
c.print_Raman();
c.print_Kumar();
c.print_Raman();
}
}
```

Output:

```
Raman
Kumar
Raman
```

5.Hybrid inheritance

It is a mix of two or more of the above types of inheritance. Since Java doesn't support multiple inheritances with classes, hybrid inheritance involving multiple inheritance is also not possible with classes. In Java, we can achieve hybrid inheritance only through Interfaces if we want to involve multiple inheritance to implement Hybrid inheritance.



Advantages Of Inheritance in Java:

- Code Reusability
- Abstraction
- Class Hierarchy:
- Polymorphism

Disadvantages of Inheritance in Java:

- Complexity
- Tight Coupling