Module – 2

Inheritance

Inheritance in Java is a feature that allows a class (child/subclass) to inherit properties and behaviors (methods and variables) from another class (parent/superclass).
It helps in code reusability, method overriding, and creating a hierarchy of classe

Single Inheritance

→ One child class inherits from one parent class.

Multilevel Inheritance

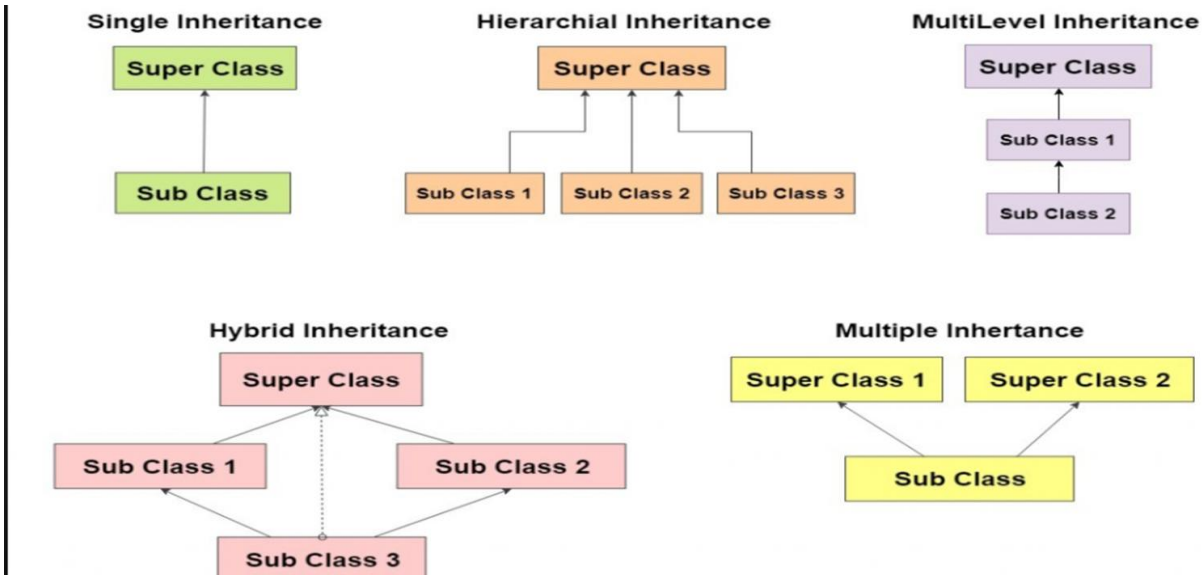
→ A class inherits from a class which itself inherits from another class.

Hierarchical Inheritance

→ Multiple child classes inherit from one parent class.

Multiple Inheritance

→ One class inherits from two or more classes (using interfaces in Java).



Single Inheritance :One child from one parent

```
class Animal {
   void sound() {
      System.out.println("Animal makes sound");
   }
}

class Dog extends Animal {
   void bark() {
      System.out.println("Dog barks");
   }
}
```

Multilevel Inheritance : Child from parent, and grandchild from child

```
class Animal {
   void sound() {
      System.out.println("Sound");
   }
}
class Dog extends Animal {
   void bark() {
      System.out.println("Bark");
   }
}
class Puppy extends Dog {
   void weep() {
      System.out.println("Weep");
   }
}
```

Hierarchical Inheritance: Two or more children from one parent

```
class Animal {
   void sound() {
      System.out.println("Sound");
   }
}

class Dog extends Animal {}
class Cat extends Animal {}
```

Multiple Inheritance (with Interface)

```
interface A {
   void show();
}

interface B {
   void display();
}

class C implements A, B {
   public void show() {
      System.out.println("Show A");
   }
   public void display() {
      System.out.println("Show B");
   }
}
```

Creating a Multilevel Hierarchy

In multilevel inheritance, a class inherits from a class, which itself inherits from another class.

Method Overriding in Java

Method must have the same name, return type, and parameters.

Example:

```
class Animal {
   void sound() {
      System.out.println("Animal makes sound");
   }
}
class Dog extends Animal {
   @Override
   void sound() {
      System.out.println("Dog barks");
   }
}
public class Test {
   public static void main(String[] args) {
      Animal a = new Dog();  // superclass reference
      a.sound();             // calls Dog's overridden method
   }
}
```

Using Abstract Classes

```
abstract class Animal {
    abstract void sound();  // abstract method

    void sleep() {          // regular method
        System.out.println("Animal sleeps");
    }
}
class Dog extends Animal {
    @Override
    void sound() {
        System.out.println("Dog barks");
    }
}
public class Test {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.sound();  // from Dog
        d.sleep();  // from Animal
    }
}
```

An interface in Java is a collection of abstract method

1. **Creating an Interface**

```
interface Animal {
   void sound(); // abstract method
}
```

2. **Implementing an Interface**

```
class Dog implements Animal {
   public void sound() {
      System.out.println("Dog barks");
   }
}
```

Use implements keyword

3. **Using Interface References**

```
public class Test {
   public static void main(String[] args) {
      Animal a = new Dog();  // Interface reference
      a.sound();             // Calls Dog's method
   }
}
```

Implementing Multiple Interfaces

Java allows a class to **implement multiple interfaces**.

```
interface A {
   void show();
}

interface B {
   void display();
}

class Demo implements A, B {
   public void show() {
      System.out.println("Showing A");
   }

   public void display() {System.out.println("Displaying B");
```

Package Fundamentals,
Packages and Member Access, Importing Packages

What is a Package?
A package is a group of related classes and interfaces.
It helps to organize code, avoid name conflicts, and control access.

Importing Packages

✅ **Syntax:**
import package_name.ClassName; **// import a specific class**
import package_name.*; **// import all classes in the package**

🔹 **Example:**

```
import mypack.MyClass;

public class Main {
   public static void main(String[] args) {
      MyClass obj = new MyClass();
      obj.show();
   }
}
```