File Name: Java_Inheritance_Types_of_Inheritance.pdf

Types of Inheritance in Java

1. Introduction

Inheritance in Java can be categorized into **different types** based on how classes inherit from each other. Understanding these types helps model real-world relationships and reuse code efficiently.

---

2. Single Inheritance

In **single inheritance**, **one child class inherits from one parent class**.

```
// Parent class
class Vehicle {
    void start() {
        System.out.println("Vehicle starts");
    }
}

// Child class
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving");
    }
}

// Main class
public class TestSingleInheritance {
    public static void main(String[] args) {
        Car c = new Car();
        c.start();  // Parent method
        c.drive();  // Child method
    }
}
```

Output:

```
Vehicle starts
Car is driving
```

---

3. Multilevel Inheritance

In **multilevel inheritance**, a class inherits from a **child class**, forming a chain.

```
// Grandparent class
class Animal {
    void eat() {
        System.out.println("Animal eats");
    }
}

// Parent class
class Mammal extends Animal {
    void walk() {
        System.out.println("Mammal walks");
    }
}

// Child class
class Dog extends Mammal {
    void bark() {
        System.out.println("Dog barks");
    }
}

// Main class
public class TestMultilevelInheritance {
    public static void main(String[] args) {
        Dog d = new Dog();
        d.eat();   // Grandparent method
        d.walk();  // Parent method
        d.bark();  // Child method
    }
}
```

Output:

```
Animal eats
Mammal walks
Dog barks
```

---

4. Hierarchical Inheritance

In **hierarchical inheritance**, **multiple child classes inherit from a single parent class**.

```
// Parent class
class Vehicle {
    void start() {
        System.out.println("Vehicle starts");
    }
}

// Child class 1
class Car extends Vehicle {
    void drive() {
        System.out.println("Car is driving");
    }
}

// Child class 2
class Bike extends Vehicle {
    void ride() {
        System.out.println("Bike is riding");
    }
}

// Main class
public class TestHierarchicalInheritance {
    public static void main(String[] args) {
        Car c = new Car();
        Bike b = new Bike();
        c.start();
        c.drive();
        b.start();
        b.ride();
    }
}
```

Output:

```
Vehicle starts
Car is driving
Vehicle starts
Bike is riding
```

---

5. Multiple Inheritance (via Interfaces)

Java **does not allow multiple inheritance** with classes to avoid ambiguity (Diamond Problem). Interfaces can be used instead.

```
interface Engine {
    void start();
}

interface Horn {
    void honk();
}

class Car implements Engine, Horn {
    public void start() {
        System.out.println("Car starts");
    }
    public void honk() {
        System.out.println("Car honks");
    }
}

public class TestMultipleInheritance {
    public static void main(String[] args) {
        Car c = new Car();
        c.start();
        c.honk();
    }
}
```

Output:

```
Car starts
Car honks
```

---

6. Summary Table

Type	Description	Example Scenario
Single	One child inherits one parent	Car → Vehicle
Multilevel	Chain of inheritance	Dog → Mammal → Animal

Type	Description	Example Scenario
Hierarchical	Multiple children inherit one parent	Car & Bike → Vehicle
Multiple (via Interface)	One class inherits multiple interfaces	Car implements Engine & Horn

---

7. Points to Remember

1. Java **does not support multiple inheritance with classes**.
2. Interfaces can be used to achieve multiple inheritance safely.
3. Use inheritance to **reuse code** and **model relationships**.
4. Avoid deep inheritance chains to maintain **readable and maintainable code**.