**Internship Daily Report – Day 12**

Date: 16 June 2025

Topic: Inheritance and Its Types in Java

Objective:

To understand the concept of Inheritance in Java and explore its various types.

What We Learned:

1. What is Inheritance?

Inheritance is a core concept of object-oriented programming (OOP) in Java that allows a class (called a subclass or child class) to inherit fields and methods from another class (called a superclass or parent class). It promotes code reusability and a logical class hierarchy.

2. Types of Inheritance in Java:

**a. Single Inheritance:**

One class inherits from another.

Example:

class Animal {

void sound() {

System.out.println("Animal makes sound");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

**b. Multilevel Inheritance:**

A class is derived from a class that is also derived from another class.

Example:

class Animal {

void sound() {

System.out.println("Animal makes sound");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

class Puppy extends Dog {

void weep() {

System.out.println("Puppy weeps");

}

}

**c. Hierarchical Inheritance:**

Multiple subclasses inherit from a single superclass.

Example:

class Animal {

void sound() {

System.out.println("Animal makes sound");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Dog barks");

}

}

class Cat extends Animal {

void meow() {

System.out.println("Cat meows");

}

}

**d. Hybrid Inheritance (via Interface):**

Java does not support hybrid inheritance with classes directly due to ambiguity issues, but it can be achieved using interfaces.

Example:

interface A {

void methodA();

}

interface B {

void methodB();

}

class C implements A, B {

public void methodA() {

System.out.println("Method A");

}

public void methodB() {

System.out.println("Method B");

}

}

**Conclusion:**

Inheritance in Java is a powerful mechanism that helps in achieving code reusability and method overriding. Understanding its types helps in designing scalable and maintainable applications. While Java does not support multiple inheritance with classes, it achieves the same through interfaces, ensuring flexibility and avoiding ambiguity.