**1. Single Inheritance**

**Definition:**  
A child class inherits from only one parent class.

**Why use:**  
To reuse code and extend functionality of a single parent.

**Example:**

class Animal {

void eat() {

System.out.println("Eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Barking...");

}

}

public class Main {

public static void main(String[] args) {

Dog d = new Dog();

d.eat(); // from Animal

d.bark(); // from Dog

}

}

**2. Multilevel Inheritance**

**Definition:**  
Inheritance chain with more than two levels.

**Why use:**  
To build specialized classes step-by-step.

**Example:**

class Animal {

void eat() {

System.out.println("Eating...");

}

}

class Dog extends Animal {

void bark() {

System.out.println("Barking...");

}

}

class Puppy extends Dog {

void weep() {

System.out.println("Weeping...");

}

}

public class Main {

public static void main(String[] args) {

Puppy p = new Puppy();

p.eat(); // from Animal

p.bark(); // from Dog

p.weep(); // from Puppy

}

}

**3. Hierarchical Inheritance**

**Definition:**  
Multiple child classes inherit from the same parent class.

**Why use:**  
To share common behavior across many classes.

**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...");

}

}

public class Main {

public static void main(String[] args) {

Dog d = new Dog();

d.eat();

d.bark();

Cat c = new Cat();

c.eat();

c.meow();

}

}

**4. Multiple Inheritance (via Interfaces)**

**Definition:**  
A class inherits from multiple sources (only possible in Java using interfaces).

**Why:**  
To combine different behaviors from different sources.

**Example:**

interface Printable {

void print();

}

interface Showable {

void show();

}

class Demo implements Printable, Showable {

public void print() {

System.out.println("Printing...");

}

public void show() {

System.out.println("Showing...");

}

}

public class Main {

public static void main(String[] args) {

Demo d = new Demo();

d.print();

d.show();

}

}

**5. Hybrid Inheritance**

**Definition:**  
Combination of two or more types of inheritance (like single + multiple).  
In Java, you can only do this via interfaces (not with classes).

**Example (Hierarchical + Multiple):**

class Animal {

void eat() {

System.out.println("Eating...");

}

}

interface Pet {

void beFriendly();

}

class Dog extends Animal implements Pet {

public void beFriendly() {

System.out.println("Wagging tail...");

}

}

class Cat extends Animal implements Pet {

public void beFriendly() {

System.out.println("Purring...");

}

}

public class Main {

public static void main(String[] args) {

Dog d = new Dog();

d.eat();

d.beFriendly();

Cat c = new Cat();

c.eat();

c.beFriendly();

}

}