**Multiple Inheritance**

1. **Create a Java program that demonstrates multiple inheritance using interfaces.**

// Define two interfaces, InterfaceA and InterfaceB, each with one method.

// Create a class Demo that implements both interfaces and provides implementations for the methods.

1. **Write a Java program to show multiple inheritance using interfaces and a class.**

// Define an interface Animal with a method makeSound().

// Define another interface Pet with a method play().

// Create a class Dog that implements both Animal and Pet, and provide implementations for makeSound() and play().

**Hybrid Inheritance**

1. **Create a Java program that demonstrates hybrid inheritance using classes and interfaces.**

// Define a class Base with a method show().

// Define an interface InterfaceX with a method display().

// Create a class Derived that extends Base and implements InterfaceX, providing implementations for both show() and display().

1. **Write a Java program to demonstrate hybrid inheritance with a concrete class, an abstract class, and an interface.**

// Define an abstract class Shape with an abstract method draw().

// Define an interface Colorable with a method color().

// Create a class Circle that extends Shape and implements Colorable, providing implementations for both draw() and color().

**Multiple Inheritance Using Interfaces**

**Program 1:**

// Define InterfaceA

interface InterfaceA {

void methodA();

}

// Define InterfaceB

interface InterfaceB {

void methodB();

}

// Implement both interfaces in Demo class

class Demo implements InterfaceA, InterfaceB {

@Override

public void methodA() {

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

}

@Override

public void methodB() {

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

}

public static void main(String[] args) {

Demo demo = new Demo();

demo.methodA();

demo.methodB();

}

}

**Program 2:**

// Define Animal interface

interface Animal {

void makeSound();

}

// Define Pet interface

interface Pet {

void play();

}

// Create Dog class that implements both Animal and Pet interfaces

class Dog implements Animal, Pet {

@Override

public void makeSound() {

System.out.println("The dog barks.");

}

@Override

public void play() {

System.out.println("The dog plays fetch.");

}

public static void main(String[] args) {

Dog dog = new Dog();

dog.makeSound();

dog.play();

}

}

**Hybrid Inheritance**

**Program 1:**

// Define Base class

class Base {

void show() {

System.out.println("Show method in Base class");

}

}

// Define InterfaceX

interface InterfaceX {

void display();

}

// Create Derived class that extends Base and implements InterfaceX

class Derived extends Base implements InterfaceX {

@Override

public void display() {

System.out.println("Display method from InterfaceX");

}

public static void main(String[] args) {

Derived derived = new Derived();

derived.show();

derived.display();

}

}

**Program 2:**

// Define an abstract class Shape

abstract class Shape {

abstract void draw();

}

// Define an interface Colorable

interface Colorable {

void color();

}

// Create Circle class that extends Shape and implements Colorable

class Circle extends Shape implements Colorable {

@Override

void draw() {

System.out.println("Drawing a Circle.");

}

@Override

public void color() {

System.out.println("Coloring the Circle.");

}

public static void main(String[] args) {

Circle circle = new Circle();

circle.draw();

circle.color();

}

}