1. Method Overloading

Question 1: Calculate Area of Shapes

```
class Shape {
    // Overloaded method to calculate area of rectangle
    public void area(int length, int breadth) {
        System.out.println("Area of Rectangle: " + (length * breadth));
    }

    // Overloaded method to calculate area of square
    public void area(int side) {
        System.out.println("Area of Square: " + (side * side));
    }

    // Overloaded method to calculate area of circle
    public void area(double radius) {
        System.out.println("Area of Circle: " + (Math.PI * radius * radiu
    }
}

public class Main {
    public static void main(String[] args) {
        Shape shape = new Shape();
        shape.area(5, 10);  // Rectangle
        shape.area(4);      // Square
        shape.area(7.0);    // Circle
    }
}
```

---

2. Method Overriding

Question 2: Employee and Manager

```
class Employee {
    public int calculateSalary() {
        return 5000;
    }
}
```

```
class Manager extends Employee {
    @Override
    public int calculateSalary() {
        return 8000;
    }
}

public class Main {
    public static void main(String[] args) {
        Employee emp = new Employee();
        Manager mgr = new Manager();

        System.out.println("Employee Salary: " + emp.calculateSalary());
        System.out.println("Manager Salary: " + mgr.calculateSalary());
    }
}
```

---

3. Types of Inheritance

Question 3: Multi-Level Inheritance

```
class Vehicle {
    public void startEngine() {
        System.out.println("Engine started.");
    }
}

class Car extends Vehicle {
    public void openTrunk() {
        System.out.println("Trunk is opened.");
    }
}

class ElectricCar extends Car {
    @Override
    public void startEngine() {
        System.out.println("Electric car starts silently.");
    }
}

public class Main {
    public static void main(String[] args) {
        ElectricCar tesla = new ElectricCar();
        tesla.startEngine();    // Calls overridden method in ElectricCar
        tesla.openTrunk();      // Calls method from Car class
    }
}
```

Question 4: Hierarchical Inheritance

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

class Dog extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Bark");
    }
}

class Cat extends Animal {
    @Override
    public void makeSound() {
        System.out.println("Meow");
    }
}

public class Main {
    public static void main(String[] args) {
        Animal dog = new Dog();
        Animal cat = new Cat();

        dog.makeSound();  // Dog sound
        cat.makeSound();  // Cat sound
    }
}
```

Question 5: Hybrid Inheritance

```
interface Employee {
    void work();
}

interface Student {
    void study();
}

class Person {
    public void displayInfo() {
        System.out.println("Person details.");
    }
}

class Intern extends Person implements Employee, Student {
    @Override
    public void work() {
        System.out.println("Working as an Employee.");
    }

    @Override
    public void study() {
        System.out.println("Studying as a Student.");
    }
}

public class Main {
    public static void main(String[] args) {
        Intern intern = new Intern();
        intern.displayInfo();
        intern.work();
        intern.study();
    }
}
```

Question 6: Multiple Inheritance Using Interfaces

```
interface Flyable {
    void fly();
}

interface Swimmable {
```

```java
    void swim();
}

class Duck implements Flyable, Swimmable {
    @Override
    public void fly() {
        System.out.println("Duck is flying.");
    }

    @Override
    public void swim() {
        System.out.println("Duck is swimming.");
    }
}

public class Main {
    public static void main(String[] args) {
        Duck duck = new Duck();
        duck.fly();   // Calls fly method
        duck.swim();  // Calls swim method
    }
}
```