Lab 27

Self Practice Solution: Write two programs based on method overloading and overriding

1. Method Overloading:

Example 1: Method Overloading for Adding Numbers

class Calculator {

int add(int a, int b) {

return a + b;

}

double add(double a, double b) {

return a + b;

}

// Method overloading with three parameters

int add(int a, int b, int c) {

return a + b + c;

}

}

public class MethodOverloadingDemo {

public static void main(String[] args) {

Calculator calc = new Calculator();

int sum1 = calc.add(5, 3); // Calls the int version of add()

double sum2 = calc.add(2.5, 3.7); // Calls the double version of add()

int sum3 = calc.add(2, 4, 6); // Calls the three-parameter version of add()

System.out.println("Sum1: " + sum1);

System.out.println("Sum2: " + sum2);

System.out.println("Sum3: " + sum3);

}

}

In this example, we have overloaded the `add` method with different parameter lists, allowing it to handle both integers and doubles and varying numbers of parameters.

2. Method Overriding:

Example 2: Method Overriding in Inheritance

class Animal {

void makeSound() {

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

}

}

class Dog extends Animal {

@Override

void makeSound() {

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

}

}

public class MethodOverridingDemo {

public static void main(String[] args) {

Animal myAnimal = new Dog();

myAnimal.makeSound(); // Calls the overridden makeSound() method in Dog

}

}

In this example, the `makeSound` method is overridden in the `Dog` subclass. When we create an instance of `Dog` and call `makeSound`, it calls the overridden method in the `Dog` class.

3. Method Overloading and Overriding Together:

Example 3: Method Overloading and Overriding Combined

class Shape {

void draw() {

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

}

void draw(String color) {

System.out.println("Drawing a shape with color: " + color);

}

}

class Circle extends Shape {

@Override

void draw() {

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

}

void draw(int radius) {

System.out.println("Drawing a circle with radius: " + radius);

}

}

public class MethodOverloadingOverridingDemo {

public static void main(String[] args) {

Circle circle = new Circle();

circle.draw(); // Calls the overridden draw() method in Circle

circle.draw("red"); // Calls the overloaded draw(String) method in Shape

circle.draw(5); // Calls the overloaded draw(int) method in Circle

}

}

In this example, we have a superclass `Shape` with an overloaded `draw` method and a subclass `Circle` that overrides the `draw` method and also overloads it with a different parameter. This demonstrates both method overloading and overriding together.

These examples illustrate how method overloading and overriding work in Java, allowing you to create methods with the same name but different behaviors based on the parameters and the class hierarchy.