Experiment Number : 07 **Date: 05.02.2026**

METHOD OVERLOADING AND METHOD OVERRIDING:

Aim:

To understand and implement method overloading and method overriding.

PRE LAB EXERCISE:

QUESTIONS

- ✓ **What is method overloading?**

  Method overloading is a feature in which multiple methods have the same name but different parameter lists (type, number, or order of parameters) in the same class.

- ✓ **What is method overriding?**

  Method overriding is a feature in which a subclass provides its own implementation of a method that is already defined in its parent class.

- ✓ **Difference between overloading and overriding.**

Method Overloading	**Method Overriding**
Same method name, different parameters	Same method name, same parameters
Occurs in the same class	Occurs in different classes (parent-child)
Compile-time polymorphism	Runtime polymorphism
No inheritance required	Inheritance is required

IN LAB EXERCISE:

Objective:

To demonstrate compile-time and runtime polymorphism.

PROGRAMS:

1.Student Result System (Method Overriding)

Description:

- Base class Student has method displayResult().
- Subclasses UGStudent and PGStudent override the method to show different grading systems.

Code :

```
import java.util.Scanner;

// Base class
class Student {
    String name;

    void displayResult() {
        System.out.println("Student Result");
    }
}

// UG Student subclass
class UGStudent extends Student {
    int marks;

    UGStudent(String n, int m) {
        name = n;
        marks = m;
    }

    @Override
    void displayResult() {
        double percentage = (marks / 100.0) * 100;
        System.out.println("UG Student: " + name);
```

```
        System.out.println("Marks: " + marks);
        System.out.println("Percentage: " + percentage + "%");
    }
}

// PG Student subclass
class PGStudent extends Student {
    double gpa;

    PGStudent(String n, double g) {
        name = n;
        gpa = g;
    }

    @Override
    void displayResult() {
        System.out.println("PG Student: " + name);
        System.out.println("GPA: " + gpa + " / 10");
    }
}

// Main class
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);

        // Input for UG student
        System.out.print("Enter UG Student Name: ");
        String ugName = sc.nextLine();
        System.out.print("Enter UG Student Marks (out of 100): ");
        int ugMarks = sc.nextInt();
```

```
        sc.nextLine(); // consume newline

        // Input for PG student
        System.out.print("Enter PG Student Name: ");
        String pgName = sc.nextLine();
        System.out.print("Enter PG Student GPA (0-10): ");
        double pgGpa = sc.nextDouble();

        // Create objects
        Student s1 = new UGStudent(ugName, ugMarks);
        Student s2 = new PGStudent(pgName, pgGpa);

        System.out.println("\n--- Student Results ---");
        s1.displayResult();
        System.out.println();
        s2.displayResult();

        sc.close();
    }
}
```

OUTPUT:

```
Enter UG Student Name: Sanjanaa
Enter UG Student Marks (out of 100): 95
Enter PG Student Name: Rithanya
Enter PG Student GPA (0-10): 99

--- Student Results ---
UG Student: Sanjanaa
Marks: 95
Percentage: 95.0%

PG Student: Rithanya
GPA: 99.0 / 10
```

2. Calculator Program (Method Overloading)

Description:
Create a Calculator class with multiple add() methods to calculate:

- Addition of 2 integers
- Addition of 3 integers
- Addition of 2 double numbers

Code:

```
import java.util.Scanner;
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    int add(int a, int b, int c) {
        return a + b + c;
    }

    double add(double a, double b) {
        return a + b;
    }
}
public class Main {
    public static void main(String[] args) {
        Scanner sc = new Scanner(System.in);
        Calculator calc = new Calculator();

        System.out.print("Enter two integers: ");
        int x = sc.nextInt();
        int y = sc.nextInt();
        System.out.println("Sum of two integers: " + calc.add(x, y));
```

```
        System.out.print("Enter three integers: ");
        int p = sc.nextInt();
        int q = sc.nextInt();
        int r = sc.nextInt();
        System.out.println("Sum of three integers: " + calc.add(p, q, r));

        System.out.print("Enter two decimal numbers: ");
        double a = sc.nextDouble();
        double b = sc.nextDouble();
        System.out.println("Sum of two doubles: " + calc.add(a, b));

        sc.close();
    }
}
```

Output:

```
Enter two integers: 2
4
Sum of two integers: 6
Enter three integers: 5
10
15
Sum of three integers: 30
Enter two decimal numbers: 2.5
3.5
Sum of two doubles: 6.0
```

POST LAB EXERCISE:

- ✓ **Is return type important in method overloading and method overriding?**
    - In method overloading, return type alone is not important; the parameter list must be different.
    - In method overriding, the return type must be the same or covariant (compatible) with the parent method.

- ✓ **Can you overload a method by changing only the return type?**

No, a method cannot be overloaded by changing only the return type. The parameter list must be different.

- ✓ **Can static methods be overridden? Can they be overloaded?**
  - Static methods cannot be overridden because they belong to the class, not the object.
  - Static methods can be overloaded by changing the parameter list.

- ✓ **Can a method be overridden if the parameter list is different?**

No, a method cannot be overridden if the parameter list is different. It will be treated as method overloading instead.

Result:

Thus the method overloading and overriding concepts were implemented and executed successfully.

ASSESSMENT

Description	Max Marks	Marks Awarded
Pre Lab Exercise	**5**	
In Lab Exercise	**10**	
Post Lab Exercise	**5**	
Viva	**10**	
Total	**30**	
Faculty Signature		