Experiment Number : 07 **Date: 05-02-26**

METHOD OVERLOADING AND METHOD OVERRIDING

Aim:

To understand and implement method overloading and method overriding.

PRE LAB EXERCISE

QUESTIONS

- ✓ What is method overloading?

  Answer: Method overloading is when **multiple methods in the same class have the same name but different parameters** (different number, type, or order of arguments). It is a form of **compile-time polymorphism**.
- ✓ What is method overriding?

  Answer: Method overriding occurs when a **child class provides its own implementation of a method already defined in the parent class** with the **same method name and parameters**. It is a form of **run-time polymorphism**.
- ✓ Difference between overloading and overriding.

  Answer:

Method Overloading	**Method Overriding**
Same class	Different classes (parent–child)
Same method name, different parameters	Same method name and same parameters
Compile-time polymorphism	Run-time polymorphism
Does not need inheritance	Requires inheritance
Improves readability	Provides specific behavior

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:

Sample Input:

```
Enter UG Student Name: Rohitha
Enter UG Student Marks (out of 100): 99
Enter PG Student Name: Neha
Enter PG Student GPA (0-10): 9.2
```

Output:

```
--- Student Results ---
```

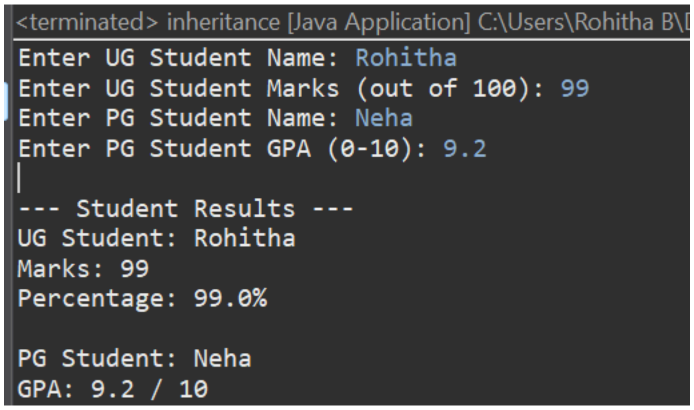
UG Student:Rohitha

Marks: 99

Percentage: 99.0%

PG Student: Ravi

GPA: 9.2 / 10



```
<terminated> inheritance [Java Application] C:\Users\Rohitha B\
Enter UG Student Name: Rohitha
Enter UG Student Marks (out of 100): 99
Enter PG Student Name: Neha
Enter PG Student GPA (0-10): 9.2

--- Student Results ---
UG Student: Rohitha
Marks: 99
Percentage: 99.0%

PG Student: Neha
GPA: 9.2 / 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:

Sample Input:

```
Enter two integers: 10 20
```

Enter three integers: 5 10 15

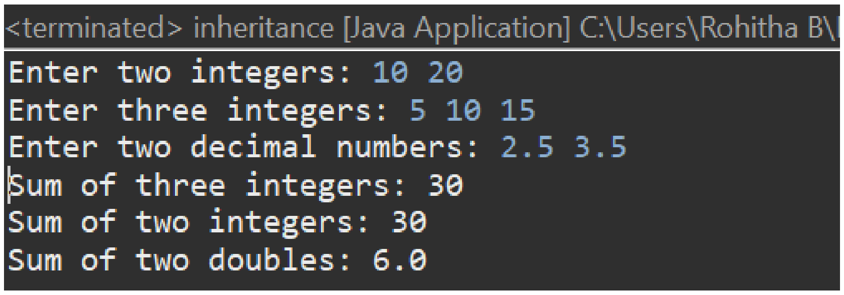
Enter two decimal numbers: 2.5 3.5

Output:

Sum of two integers: 30

Sum of three integers: 30

Sum of two doubles: 6.0



```
<terminated> inheritance [Java Application] C:\Users\Rohitha B\
Enter two integers: 10 20
Enter three integers: 5 10 15
Enter two decimal numbers: 2.5 3.5
Sum of three integers: 30
Sum of two integers: 30
Sum of two doubles: 6.0
```

POST LAB EXERCISE

- ✓ Is return type important in method overloading and method overriding?

  Answer:

  Method Overloading:

  Return type alone is **NOT important —** parameters must be different. Return type by itself cannot distinguish overloaded methods.

  Method Overriding:

  Return type must be **same or covariant (subclass type)**.

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

  Answer: **No.**

  Changing only the return type does NOT overload a method. The parameter list must be different.

- ✓ Can static methods be overridden? Can they be overloaded?

  Answer:

  1. Static methods **CANNOT be overridden** (they belong to the class, not object).

  2.Static methods **CAN be overloaded**.

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

  Answer: **No.**

  For overriding, the method name AND parameter list must be exactly the same.

  If parameters differ, it becomes **overloading**, not overriding.

  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	

ROHITHA B

24BCS229