Experiment Number : 07 **Date:**

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 where **multiple methods have the same name** but **different parameter lists** (number, type, or order of parameters) within the **same class**.
It improves **readability** and **flexibility** of code.

- ✓ What is method overriding?

Method overriding occurs when a **child class provides its own implementation** of a method already defined in the **parent class**, using the **same method name and same parameters**.
It supports **runtime polymorphism**.

- ✓ Difference between overloading and overriding.

Method Overloading	**Method Overriding**
Occurs in the **same class**	Occurs in **parent–child classes**
Method name is same, **parameters differ**	Method name and parameters are **same**
Compile-time polymorphism	Runtime polymorphism
extends not required	Requires **inheritance**
Return type can differ	Return type must be **same or covariant**

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: Prashanth

Enter UG Student Marks (out of 100): 85

Enter PG Student Name: Prash

Enter PG Student GPA (0-10): 9.2

Output:

--- Student Results ---

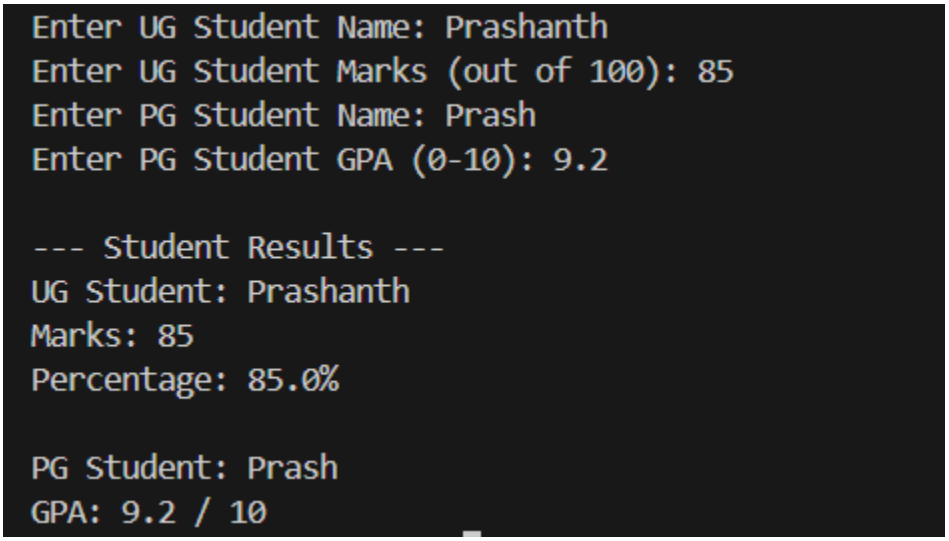
UG Student: Prashanth

Marks: 85

Percentage: 85.0%

PG Student: Prash

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

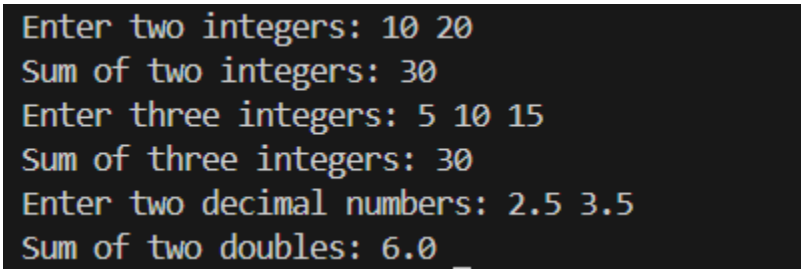
Enter two decimal numbers: 2.5 3.5

Output:

Sum of two integers: 30

Sum of three integers: 30

Sum of two doubles: 6.0



POST LAB EXERCISE

- ✓ Is return type important in method overloading and method overriding?
- ✓ • Method **overloading:** Return type alone is **not important**; overloading depends on the **parameter list**.
- ✓ • Method **overriding:** Return type **must be same or covariant** to maintain method compatibility.

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

Changing only the return type does **not** overload a method because the **method signature remains the same**, causing a **compile-time error**.

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

- Static **methods cannot be overridden** because they are **class-level** methods.
- Static **methods can be overloaded** by changing the parameter list.

- ✓ Can a method be overridden if the parameter list is different?4

  No.
  For overriding:

- Method name
- Parameter list
  must be **exactly the same**.
  If the parameter list differs, it becomes **method 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	