

Theeran P

24BCS298

CSE-A1

## **METHOD OVERLOADING AND METHOD OVERRIDING**

### **Aim:**

To understand and implement method overloading and method overriding.

### **PRE LAB EXERCISE**

#### **QUESTIONS**

##### **1. What is method overloading?**

- **Method overloading** means having **multiple methods with the same name** in the **same class**, but with **different parameters** (different number, type, or order of parameters).
- It is used to **perform similar operations in different ways**.

##### **2. What is method overriding?**

- **Method overriding** means a **subclass provides its own implementation** of a method that already exists in its **parent class**, with the **same method name, same parameters, and same return type**.
- It is used to **change or extend parent class behaviour**.

##### **3. Difference between overloading and overriding.**

Method Overloading	Method Overriding
Happens in <b>same class</b>	Happens in <b>parent-child classes</b>
Same method name, <b>different parameters</b>	Same method name, <b>same parameters</b>
Return type <b>can be different</b>	Return type must be <b>same (or covariant)</b>
Compile-time polymorphism	Runtime polymorphism
Does <b>not</b> require inheritance	<b>Requires</b> inheritance
Faster (resolved at compile time)	Slower (resolved at runtime)

## 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();
    }
}
```

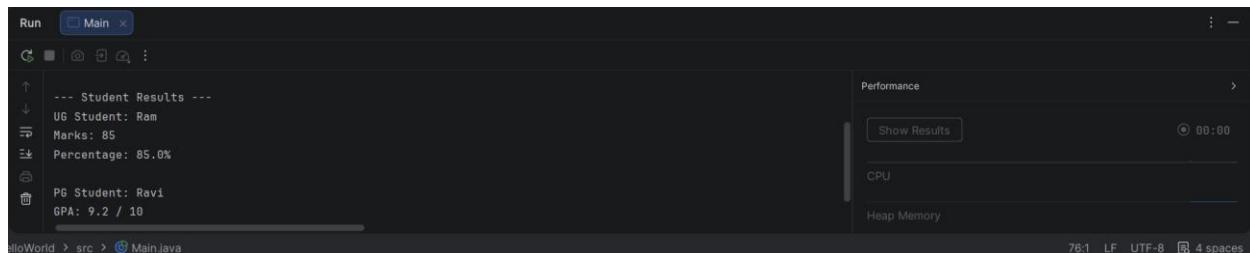
## OUTPUT:

Sample Input:

```
Enter UG Student Name: Ram  
Enter UG Student Marks (out of 100): 85  
Enter PG Student Name: Ravi  
Enter PG Student GPA (0-10): 9.2
```

Output:

```
--- Student Results ---  
UG Student: Ram  
Marks: 85  
Percentage: 85.0%  
PG Student: Ravi  
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

```

```

Run Main ×
C:\Users\User\.jdks\openjdk-25.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2025.3.2\lib\idea_rt.jar=50007" -Dfile.encoding=UTF-8 -Dsun.stdout.encoding=UTF-8
Enter two integers: 10 20
Sum of two integers: 30
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
Process finished with exit code 0

```

## POST LAB EXERCISE

1. Is return type important in method overloading and method overriding?
  - In method overloading, return type is not important because overloading depends on method name and parameter list.

- In method overriding, return type is important and must be the same or covariant as the parent class method.
2. Can you overload a method by changing only the return type?
- No, you cannot overload a method by changing only the return type because the method signature remains the same and it causes a compile time error.
3. Can static methods be overridden? Can they be overloaded?
- Static methods cannot be overridden because they belong to the class and not to objects.
  - Static methods can be overloaded by changing the parameter list.
4. Can a method be overridden if the parameter list is different?
- No, a method cannot be overridden if the parameter list is different.  
If the parameter list is different, it becomes method overloading and 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	
<b>Total</b>	<b>30</b>	
<b>Faculty Signature</b>		