Aror University of Art, Architecture, Design & Heritage Sukkur

Department of AI-Multimedia and Gaming

BS-AI and MMG, Fall 2024 Batch

_____________ Lab 08: Polymorphism and Encapsulation Course: Object Oriented Programming

Date: 8 April, 2025

—--------------------—--------------------—---------------------—--------------------—--------

1. Polymorphism: Method Overloading (also called **Compile-time polymorphism**)

Concept: Method overloading allows multiple methods with the **same name** but **different parameters** in the same class.

Task 1: Calculator Operations

Create a Calculator class with overloaded add() methods:

- add(int a, int b) → returns sum of two integers.
- add(double a, double b) → returns sum of two doubles.
- add(String a, String b) → returns concatenated string.

Test Case:

```
Calculator calc = new Calculator();
System.out.println(calc.add(5, 3));        // Output: 8
System.out.println(calc.add(2.5, 3.7));    // Output: 6.2
System.out.println(calc.add("Hello", "World")); // Output: HelloWorld
```

Task 2: Employee Bonus Calculation

Create an Employee class with overloaded calculateBonus() methods:

- calculateBonus(int salary) → returns 10% of salary.
- calculateBonus(int salary, int extraHours) → returns 10% of salary + 500 per extra hour.

Test Case:

```
Employee emp = new Employee();
System.out.println(emp.calculateBonus(50000));       // Output: 5000
System.out.println(emp.calculateBonus(50000, 3));    // Output: 6500
```

2. Polymorphism: Method Overriding (Also called Run Time Polymorphism in Java)

Concept: Method overriding allows a **subclass** to provide a different implementation of a method already defined in its **parent class**.

Task 1: Animal Sounds

Create:

- A parent class Animal with a method makeSound() → prints "Animal makes a sound."
- Two subclasses Dog and Cat that override makeSound() to print "Dog barks" and "Cat meows" respectively.

Test Case:

```
Animal myAnimal = new Animal();
Animal myDog = new Dog();
Animal myCat = new Cat();

myAnimal.makeSound(); // Output: Animal makes a sound.
myDog.makeSound();    // Output: Dog barks.
myCat.makeSound();    // Output: Cat meows.
```

Task 2: Bank Interest Rates

Create:

- A parent class Bank with a method getInterestRate() → returns 0.
- Subclasses SBI, ICICI, and HDFC that override getInterestRate() to return 7.5, 8.2, and 6.8 respectively.

Test Case:

```
Bank sbi = new SBI();
Bank icici = new ICICI();
Bank hdfc = new HDFC();

System.out.println("SBI Interest: " + sbi.getInterestRate());   // Output: 7.5
System.out.println("ICICI Interest: " + icici.getInterestRate()); // Output: 8.2
System.out.println("HDFC Interest: " + hdfc.getInterestRate());  // Output: 6.8
```

3. Encapsulation in Java

Concept: Encapsulation binds data (variables) and methods together while **hiding internal details** using private fields and public getters/setters.

Task 1: Student Record System

Create a Student class with:

- Private fields: name, rollNumber, grade.
- Public getter/setter methods.
- A method displayDetails() that prints student info.

Test Case:

```
Student student = new Student();
student.setName("Alice");
student.setRollNumber(101);
student.setGrade('A');

student.displayDetails();
// Output:
// Name: Alice
// Roll Number: 101
// Grade: A
```

Task 2: Temperature Converter

Create a Temperature class with:

- A private field celsius.
- Public methods:
    - setCelsius(double c) → sets temperature in Celsius.
    - getFahrenheit() → converts Celsius to Fahrenheit (F = C × 9/5 + 32).

Test Case:

```
Temperature temp = new Temperature();
temp.setCelsius(25);
System.out.println("Fahrenheit: " + temp.getFahrenheit());
// Output: Fahrenheit: 77.0
```