Lab Sheet: Object-Oriented Programming in Java – Class and Object

Course: Introduction to Programming Language II (Java)
Topic: Class, Object, Constructor, and Methods
Duration: 2 hours

1. Objectives

By the end of this lab, students will be able to:

- Understand the concept of classes and objects in Java.
- Define and use constructors (default and parameterized).
- Write and call methods.
- Use objects to access class data and methods.
- Differentiate between instance and static members.

2. Theoretical Background

What is a Class?. A **class** is a blueprint or template for creating objects. It defines data members (variables) and methods (functions).

```java
class ClassName {
    // Data members (variables)
    // Methods (functions)
}
```

What is an Object?. An **object** is an instance of a class that occupies memory.

```java
ClassName obj = new ClassName();
```

3. Example 1: Basic Class and Object

```java
class Student {
    String name;
    int age;

    void displayInfo() {
        System.out.println("Name: " + name);
        System.out.println("Age: " + age);
    }
}

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();
        s1.name = "Ashraful";
        s1.age = 22;
        s1.displayInfo();
    }
}
```

Output:

```
Name: Ashraful
Age: 22
```

4. Example 2: Using Constructors

```
class Student {
    String name;
    int age;

    // Default constructor
    Student() {
        name = "Unknown";
        age = 0;
    }

    // Parameterized constructor
    Student(String n, int a) {
        name = n;
        age = a;
    }

    void display() {
        System.out.println(name + " is " + age + " years old.");
    }
}

public class Main {
    public static void main(String[] args) {
        Student s1 = new Student();
        Student s2 = new Student("Paran", 22);
        s1.display();
        s2.display();
    }
}
```

Output:

```
Unknown is 0 years old.
Paran is 22 years old.
```

5. Example 3: Method with Parameters and Return Type

```
class Calculator {
    int add(int a, int b) {
        return a + b;
    }

    void showSquare(int n) {
        System.out.println("Square of " + n + " = " + (n * n));
    }
}

public class Main {
    public static void main(String[] args) {
        Calculator calc = new Calculator();
        int sum = calc.add(10, 20);
        System.out.println("Sum = " + sum);
        calc.showSquare(5);
    }
}
```

6. Example 4: Static vs Instance Methods

```
class MathUtils {
    void greet() {
        System.out.println("Hello from an instance method!");
    }

    static void info() {
        System.out.println("This is a static method.");
    }
}

public class Main {
    public static void main(String[] args) {
        MathUtils obj = new MathUtils();
        obj.greet();
        MathUtils.info();
    }
}
```

7. Example 5: Multiple Objects and Constructor Overloading

```
class Rectangle {
    int length, width;

    Rectangle() {
        length = 1;
        width = 1;
    }

    Rectangle(int l, int w) {
        length = l;
        width = w;
    }

    int area() {
        return length * width;
    }
}

public class Main {
    public static void main(String[] args) {
        Rectangle r1 = new Rectangle();
        Rectangle r2 = new Rectangle(10, 5);
        System.out.println("Area of r1 = " + r1.area());
        System.out.println("Area of r2 = " + r2.area());
    }
}
```

8. Example 6: Object as Parameter

```
class Box {
    int height, width, depth;

    Box(int h, int w, int d) {
        height = h;
        width = w;
        depth = d;
    }

    boolean isEqual(Box b) {
        return (height == b.height && width == b.width && depth == b.depth)
            ↪ ;
    }
}

public class Main {
    public static void main(String[] args) {
        Box b1 = new Box(10, 20, 30);
        Box b2 = new Box(10, 20, 30);
        Box b3 = new Box(5, 10, 15);
        System.out.println("b1 == b2? " + b1.isEqual(b2));
        System.out.println("b1 == b3? " + b1.isEqual(b3));
    }
}
```

9. Additional Examples

Example 7: Using `this` Keyword.

```
class Person {
    String name;
    int age;

    // Parameterized constructor using 'this'
    Person(String name, int age) {
        this.name = name;  // 'this' refers to current object
        this.age = age;
    }

    void introduce() {
        System.out.println("Hi, I'm " + this.name + " and I'm " + this.age
            ↪ + " years old.");
    }
}
```

```java
public class Main {
    public static void main(String[] args) {
        Person p = new Person("Alice", 25);
        p.introduce();
    }
}
```

Output:

```
Hi, I'm Alice and I'm 25 years old.
```

Example 8: Method Overloading.

```java
class Printer {
    void print(String text) {
        System.out.println("Text: " + text);
    }

    void print(int number) {
        System.out.println("Number: " + number);
    }

    void print(String text, int times) {
        for (int i = 0; i < times; i++) {
            System.out.println(text);
        }
    }
}

public class Main {
    public static void main(String[] args) {
        Printer p = new Printer();
        p.print("Hello");
        p.print(100);
        p.print("Java", 3);
    }
}
```

Output:

```
Text: Hello
Number: 100
Java
Java
Java
```

Example 9: Returning an Object from Method.

```java
class Counter {
    int value;

    Counter(int value) {
        this.value = value;
    }

    Counter increment() {
        return new Counter(this.value + 1);
    }
    void display() {
        System.out.println("Count: " + value);
    }
}

public class Main {
    public static void main(String[] args) {
        Counter c1 = new Counter(5);
        Counter c2 = c1.increment();
        c1.display();
        c2.display();
    }
}
```

Output:

```
Count: 5
Count: 6
```

Example 10: Array of Objects.

```java
class Book {
    String title;
    String author;

    Book(String title, String author) {
        this.title = title;
        this.author = author;
    }

    void display() {
        System.out.println(title + " by " + author);
    }
}

public class Main {
    public static void main(String[] args) {
        // Array of 3 Book objects
        Book[] library = new Book[3];

        library[0] = new Book("Java Programming", "James Gosling");
        library[1] = new Book("Clean Code", "Robert Martin");
        library[2] = new Book("Design Patterns", "Gang of Four");

        System.out.println("Library Catalog:");
        for (Book b : library) {
            b.display();
        }
    }
}
```

Output:

```
Library Catalog:
Java Programming by James Gosling
Clean Code by Robert Martin
Design Patterns by Gang of Four
```

Example 11: Static Variables (Class-level Data).

```java
class Employee {
    String name;
    static int totalEmployees = 0;  // Shared among all objects

    Employee(String name) {
        this.name = name;
        totalEmployees++;
    }

    void show() {
        System.out.println(name + " (Employee #" + totalEmployees + ")");
    }

    static void showTotal() {
        System.out.println("Total Employees: " + totalEmployees);
    }
}

public class Main {
    public static void main(String[] args) {
        Employee e1 = new Employee("Rahim");
        Employee e2 = new Employee("Karim");
        Employee e3 = new Employee("Fahim");

        e1.show();
        e2.show();
        e3.show();
        Employee.showTotal();
    }
}
```

Output:

```
Rahim (Employee #3)
Karim (Employee #3)
Fahim (Employee #3)
Total Employees: 3
```

10. Lab Tasks

Task 1: Car Class

Create a class `Car` with:

- Fields: `brand`, `model`, `year`
- Methods: `displayDetails()`
- Constructors: default & parameterized

Expected Output:

```
Car: Toyota Corolla (2022)
Car: Tesla Model 3 (2024)
```

Task 2: Bank Account
Write a class `BankAccount` that has:

- Fields: `accountNumber`, `holderName`, `balance`
- Methods: `deposit()`, `withdraw()`, `showBalance()`

Demonstrate deposit and withdraw operations.

Task 3: Circle
Create a class `Circle` with:

- Field: `radius`
- Methods: `area()` and `circumference()`

Take input from user using `Scanner`.

Task 4: Employee
Design a class `Employee` with overloaded constructors to initialize employee data in different ways. Use `this` keyword properly.

Task 5 (Challenge): Student Management System

- Store details of 3 students.
- Use an array of objects: `Student[] students = new Student[3];`
- Display all students' information.

11. Questions for Practice

1. What is the difference between a class and an object?
2. Can we have multiple constructors in a class? How?
3. What is the role of the `this` keyword?
4. What happens if you don't define a constructor?
5. What is the difference between a static method and an instance method?