Lab Sheet: Understanding Static Variables, Objects, and Methods in Java

Objectives

- To understand the concept of static variables and methods in Java.
- To learn how to use static members in a Java program.
- To implement and practice using static variables and methods in various scenarios.

Introduction

In Java, the `static` keyword can be applied to variables and methods. Static members belong to the class rather than instances of the class, meaning they are shared among all instances.

Static Variables

- **Static Variable**: A variable that is shared among all instances of a class. It is also known as a class variable. There is only one copy of the static variable, regardless of how many objects are created from the class.

Static Methods

- **Static Method**: A method that can be called without creating an instance of the class. It can access static variables and other static methods directly but cannot access non-static variables and methods directly.

Exercise 1: Static and Non-Static Variables

1. Create a class called `Counter` with the following:
    - A static variable `staticCounter`.
    - A non-static variable `nonStaticCounter`.
    - Create a method called `setStaticCount(int num)` to set the value of the `staticCounter`
    - Create a method called `setNonStaticCount(int num)` to set the value of the `nonStaticCounter`
2. Create a main class to instantiate multiple `Counter` objects and observe the values of `staticCounter` and `nonStaticCounter`.

    Create two instances/objects of Counter class. (counter1 and counter2)

    - Then set the `nonStaticCounter` value of the **counter1** object to 30;
    - Then set the `staticCounter` value of the **counter1** object to 45;
    - Then set the `nonStaticCounter` value of the **counter2** object to 65;
    - Then set the `staticCounter` value of the **counter2** object to 75;
    - Print the value of the `staticCounter` and `nonStaticCounter` of both counter1 and counter2

Exercise 2: Static Methods

1. Create a class called `MathOperations` with the following:
    - A static method `add(int a, int b)` that returns the sum of two numbers.
    - A non-static method `multiply(int a, int b)` that returns the product of two numbers.
2. In the main method, demonstrate calling the static method without creating an instance and calling the non-static method by creating an instance.

Exercise 3: Static and Non-Static Methods in a Banking Application

Objective:

To understand the use of static and non-static methods in a practical scenario by creating a simple banking application.

Instructions:

1. Create a class called `BankAccount` with the following:
    - A static variable `totalAccounts` to keep track of the total number of bank accounts created.
    - A static method `initialize()` to increment the `totalAccounts` by one
    - A non-static variable `balance` to store the balance of an individual account.
    - A static method `getTotalAccounts()` that returns the total number of bank accounts.
    - A non-static method `deposit(double amount)` that adds the specified amount to the account balance.
    - A non-static method `withdraw(double amount)` that deducts the specified amount from the account balance if there are sufficient funds.
    - A non-static method `getBalance()` that returns the current balance of the account.
    - A constructor that initializes the balance to a specified amount and increments the `totalAccounts` counter.

2. In the main method, demonstrate the following:
    - Create multiple `BankAccount` objects.
    - Call method `initialize()` to increment the number of account
    - Deposit and withdraw money from the accounts. (using each object)
    - Display the current balance of each account.
    - Display the total number of bank accounts created.