DAY 6 PRACTICE QUESTIONS

1.Calculate total item sold and total revenue in Online Store

Question: You are developing an online store system. The store needs to keep track of the total number of items sold and the total revenue generated. Each Item has a name, price, and a method to sell the item. When an item is sold, the total number of items sold and the total revenue should be updated.

Which members of the Item class should be static and why?

Write a complete Java program that demonstrates the use of static members to keep track of the total items sold and total revenue.

Test Case 1: Basic Sale

Steps:

Create an Item instance for a laptop with a price of $1000.

Sell 2 laptops.

Check if Item.getTotalItemsSold() returns 2.

Check if Item.getTotalRevenue() returns 2000.0.

Expected Result:

Total Items Sold: 2

Total Revenue: $2000.0

Test Case 2: Multiple Item Sale

Steps:

Create an Item instance for a laptop with a price of $1000.

Create an Item instance for a smartphone with a price of $500.

Sell 3 laptops.

Sell 5 smartphones.

Check if Item.getTotalItemsSold() returns 8.

Check if Item.getTotalRevenue() returns 5500.0.

Expected Result:

Total Items Sold: 8

Total Revenue: $5500.0

Test Case 3: No Sales

Steps:

Create an Item instance for a laptop with a price of $1000.

Check if Item.getTotalItemsSold() returns 0.

Check if Item.getTotalRevenue() returns 0.0.

Expected Result:

Total Items Sold: 0

Total Revenue: $0.0

Test Case 4: Mixed Item Sale

Steps:

Create an Item instance for a laptop with a price of $1000.

Create an Item instance for a smartphone with a price of $500.

Sell 1 laptop.

Sell 4 smartphones.

Check if Item.getTotalItemsSold() returns 5.

Check if Item.getTotalRevenue() returns 3000.0.

Expected Result:

Total Items Sold: 5

Total Revenue: $3000.0

Test Case 5: Large Quantity Sale

Steps:

Create an Item instance for a laptop with a price of $1000.

Sell 100 laptops.

Check if Item.getTotalItemsSold() returns 100.

Check if Item.getTotalRevenue() returns 100000.0.

Expected Result:

Total Items Sold: 100

Total Revenue: $1000 00.0

2. University Student Management System

Question: You are developing a University Student Management System. The university needs to keep track of the total number of students enrolled and the total number of courses offered. Each Student has a name and an ID, and each Course has a title and a code. The system should be able to enroll students and add new courses, updating the total counts accordingly.

Which members of the Student and Course classes should be static and why?

Write a complete Java program that demonstrates the use of static members to keep track of the total number of students enrolled and the total number of courses offered.

Test Case 1: Basic Student Enrollment

Steps:

Create 2 Student instances with names and IDs.

Check if Student.getTotalStudentsEnrolled() returns 2.

Expected Result:

Total Students Enrolled: 2

Test Case 2: Basic Course Offering

Steps:

Create 2 Course instances with titles and codes.

Check if Course.getTotalCoursesOffered() returns 2.

Expected Result:

Total Courses Offered: 2

Test Case 3: Mixed Enrollment and Offering

Steps:

Create 3 Student instances.

Create 2 Course instances.

Check if Student.getTotalStudentsEnrolled() returns 3.

Check if Course.getTotalCoursesOffered() returns 2.

Expected Result:

Total Students Enrolled: 3

Total Courses Offered: 2

Test Case 4: No Enrollment or Offering

Steps:

Check if Student.getTotalStudentsEnrolled() returns 0.

Check if Course.getTotalCoursesOffered() returns 0.

Expected Result:

Total Students Enrolled: 0

Total Courses Offered: 0

Test Case 5: Reset and Recount

Steps:

Create 5 Student instances.

Create 3 Course instances.

Check if Student.getTotalStudentsEnrolled() returns 5.

Check if Course.getTotalCoursesOffered() returns 3.

Expected Result:

Total Students Enrolled: 5

Total Courses Offered: 3

3.Bank Account Management

Question: Create a Java program that simulates a bank account. The account should have a private balance attribute and methods to deposit and withdraw money. Ensure that the balance cannot be accessed or modified directly from outside the class.

Test Case 1: Initial Balance Test

Title: Verify initial balance setup

Steps: Create a BankAccount object with an initial balance of 1000.0.

Expected Result: The balance should be 1000.0.

Test Case 2: Deposit Test

Title: Verify deposit functionality

Steps: Deposit 500.0 into the account.

Expected Result: The balance should be 1500.0.

Test Case 3: Withdraw Test

Title: Verify withdraw functionality

Steps: Withdraw 200.0 from the account.

Expected Result: The balance should be 1300.0.

Test Case 4: Overdraw Test

Title: Verify withdraw functionality for overdraw

Steps: Withdraw 2000.0 from the account.

Expected Result: The balance should remain 1300.0 and no money should be withdrawn.

Test Case 5: Negative Deposit Test

Title: Verify deposit functionality for negative amount

Steps: Deposit -500.0 into the account.

Expected Result: The balance should remain 1300.0 and no money should be deposited.

4.Student Grade Management

Question: Create a Java program to manage student grades. The program should have a private attribute for the student's grade and methods to set and get the grade. Ensure the grade cannot be set to an invalid value (e.g., less than 0 or greater than 100).

Test Case 1: Valid Grade Test

Title: Verify valid grade setup

Steps: Set the student's grade to 85.

Expected Result: The grade should be 85.

Test Case 2: Invalid High Grade Test

Title: Verify invalid high grade setup

Steps: Set the student's grade to 110.

Expected Result: The grade should remain unchanged, and an error message should be displayed.

Test Case 3: Invalid Low Grade Test

Title: Verify invalid low grade setup

Steps: Set the student's grade to -10.

Expected Result: The grade should remain unchanged, and an error message should be displayed.

Test Case 4: Edge Case Low Grade Test

Title: Verify edge case low grade setup

Steps: Set the student's grade to 0.

Expected Result: The grade should be 0.

Test Case 5: Edge Case High Grade Test

Title: Verify edge case high grade setup

Steps: Set the student's grade to 100.

Expected Result: The grade should be 100.

5.Employee Information Management

Question: Create a Java program that encapsulates employee information. The program should have private attributes for the employee's name and salary, and provide public methods to set and get these attributes. Ensure that the salary cannot be set to a negative value.

Test Case 1: Valid Name Test

Steps: Set the employee's name to "John Doe".

Expected Result: The name should be "John Doe".

Test Case 2: Valid Salary Test

Steps: Set the employee's salary to 50000.0.

Expected Result: The salary should be 50000.0.

Test Case 3: Invalid Negative Salary Test

Steps: Set the employee's salary to -10000.0.

Expected Result: The salary should remain unchanged, and an error message should be displayed.

Test Case 4: Edge Case Zero Salary Test

Steps: Set the employee's salary to 0.

Expected Result: The salary should be 0.

Test Case 5: Change Name Test

Steps: Change the employee's name to "Jane Smith".

Expected Result: The name should be "Jane Smith".