SE 115 Introduction to Programming I

Lab No:	07
Topic:	Classes and Objects

Scenario 0: Write the program that demonstrates the Point - Circle class example from the WEEK 10 slides.

Scenario 1: Write a program that performs following instructions:

1. Create a class called **Student**:
    - **Do not** define any constructor manually.
    - Give it two class members: **name** and **age**. Decide their data types.
    - Implement a method called **printStudentInfo** that displays students' name and age.
2. In the main method (in a separate class, e.g., StudentDemo):
    - Create a Student object.
    - Call the printStudentInfo method and run the program.
3. Observe the output.
    - What are the values assigned to name and age?
    - Did you get any errors? Why?
    - Answer these questions in comment lines.
4. Extension**:**
    - Now **explicitly** define a **default constructor** in the Student class that sets default values for the data members you created and run the program.
    - Now define a **parameterized constructor** in the Student class **without removing** the default constructor you created.
    - Run the program again and compare the behavior. Justify your observation in comment lines. Why did you get an error or not in the main method?

Scenario 2: Your company has asked you to design a small banking system that allows customers to manage their accounts. Create a class named **BankAccount** with two **private data members: accountID and balance**. Provide a constructor that initializes these values when a new account is created. Since the data members are **private**, you should also provide **setter** and **getter** methods to access them.

Next, add methods for common banking operations:

- **deposit** method that increases the balance if the amount of money deposited is greater than zero.
- **withdraw** method that decreases the balance if the account has enough money, otherwise ask customers if they want to go into borrowing, and if they say yes, reduce their balance to negative, if not, display a warning message "Insufficient balance!"
- **accountDetails** method displays current information about the account; balance and accountID.
- Since this is a continuous application, **the program should terminate only if the customer wants to quit.** You can use the following template to create a menu for banking operations:
    - 1 → Deposit
    - 2 → Withdraw
    - 3 → Account Details
    - 4 → Exit

Finally, write a separate demo class to test your system. In the main method, create at least two **BankAccount** objects, perform deposits and withdrawals, and print the final balance of each account and accountID.

What if your customer wants to change accountID? Perform this situation and print the old and the new accountID.

Scenario 3: Define a class **Book** with two data members: **title and pages.** Then, provide a constructor to initialize these two fields to perform followings:

1. Write a method **addPages** that increases the number of pages.
2. Write a method **printBook()** that displays the title and page count.

In a separate class **BookDemo**;

1. create a Book object called **b1**.
2. Create another Book object called **b2**, that refers to the same object **(b1)**. Then, call addPages(20) using the second variable.
3. Print the data members' information of both objects.

In comments, explain why both show the updated number of pages and how many objects are actually created in memory?

As an extension, create a new Book object called **b3** with the same values as the first object (b1). Compare references of the two objects **by printing objects (e.g. System.out.println(b3)**. Compare field values of the two objects. In comments, explain why the reference comparison gives a different result than the field comparison.

Bonus Scenario: Create a class **Product** with two **private** data members: **name** and **stock**.

- Add a constructor to initialize these values.
- Add a method **buyOne** that decreases stock if available; otherwise prints **"Out of stock"**.
- If the entered product does not exist, print **"No such product!"**.
- Add a method **printInfo** that shows the product name and remaining stock.

In a separate class ProductDemo:

1. Ask the user how many products the store has and create an array of products with that size.
2. Use a loop to fill the array by asking the user for each product's name and stock.
3. Create a loop where the user can type a product name to buy it, or "Q" to quit.
    - Buying decreases stock by one, if available.
    - If there is no remaining selected product in the stock, print **"Out of stock!"**.
4. Attempt to buy a product again that is out of stock.
5. After the user quits, call the **printInfo** method for all products.