OOP Problems for Week 2 Lab Session (CSE222)

Objective:

To reinforce the understanding of classes, objects, and basic OOP principles in Java, including creating and instantiating classes, using access specifiers, and working with constructors and methods.

Problems

1. Create a Student Class

Write a program to create a Student class with the following attributes: name, ID, and CGPA. Add methods to:

- Input student details.
- Display the details of the student.

Expected Outcome:

Students will practice creating a class, defining attributes, and using methods to manipulate class data.

2. Define a Circle Class

Create a Circle class with:

- Attribute: radius.
- Methods to calculate the area and circumference of a circle.

Take the radius as input from the user and display the calculated area and circumference.

Expected Outcome:

Understand how to define attributes, methods, and perform calculations in a class.

3. Develop a BankAccount Class

Create a BankAccount class with:

- Attributes: accountNumber, accountHolderName, and balance.
- Methods to:
- Deposit an amount.

- Withdraw an amount (check if sufficient balance is available).
- Display the current balance.

Expected Outcome:

Students will learn about methods interacting with attributes and validating data.

4. Create a Car Class

Design a Car class with:

- Attributes: brand, model, year, and price.
- Method: displayDetails() to display all car information.

Instantiate two Car objects with different data and call the displayDetails() method for each.

Expected Outcome:

Practice creating multiple objects of a class and invoking their methods.

5. Define a Rectangle Class

Create a Rectangle class with:

- Attributes: length and width.
- Methods to:
- Calculate and return the area.
- Calculate and return the perimeter.

Test the class by creating objects with different dimensions.

Expected Outcome:

Understand the use of methods to perform calculations and return values.

6. Write a Book Class

Create a Book class with:

- Attributes: title, author, and price.
- Add two constructors:

- Default constructor to initialize attributes with default values.
- Parameterized constructor to initialize attributes with user-provided values.

Print the book details for objects created using both constructors.

Expected Outcome:

Understand constructor overloading and the difference between default and parameterized constructors.

7. Create a Person Class

Write a program to define a Person class with:

- Attributes: name and age.
- Methods:
- A method to calculate the year of birth based on the current year.
- A method to display the person’s details.

Test the class by creating objects and calling the methods.

Expected Outcome:

Use methods to process data and return calculated results.

8. Build an Employee Class

Develop an Employee class with:

- Attributes: employeeID, name, and salary.
- Methods to:
- Set the employee details.
- Calculate a bonus (10% of the salary) and return the total salary.

Test the class by creating objects and displaying the total salary.

Expected Outcome:

Understand how to implement methods with calculations based on class attributes.

9. Design a Triangle Class

Create a Triangle class with:

- Attributes: sideA, sideB, and sideC.
- Methods to:
- Calculate and return the perimeter.
- Check if the triangle is valid (sum of any two sides > the third side).

Test the class by taking side lengths as input and verifying its validity.

Expected Outcome:

Understand attribute validation and conditional logic within methods.

10. Create a TemperatureConverter Class

Write a TemperatureConverter class with:

- Methods to:
- Convert Celsius to Fahrenheit.
- Convert Fahrenheit to Celsius.

Take user input for temperature and the desired conversion, then display the result.

Expected Outcome:

Learn how to create utility classes and implement methods for specific operations.

Evaluation Criteria

- Class Structure: Proper definition of attributes and methods.
- Coding Standards: Use of meaningful variable names, indentation, and comments.
- Functionality: Code compiles without errors and performs as expected.
- Creativity: Additional methods or features added to enhance the functionality.