Object Oriented Programming using Java

Assignment - 5

Inheritance & Polymorphism

1. Define a class Person with properties name and dob (String) and a member function personDetails() to display the details of a person. Create a subclass Employee with an additional property comapnyName and employeeId and a method employeeDetails() to display employee details. Create objects of both Person and Employee class and display their details.
2. Create class called Rectangle with two data members length and breadth and two member function inputData(), to initialize the data and area() to compute the area of a rectangle. Create a subclass called Square with one data member called length and overrides both the methods of the Rectangle class. Create object of both the class and display their areas.
3. Design an employee hierarchy with multiple levels of inheritance. There are three types of employees: Employee (base class), Manager (inherits from Employee), and Director (inherits from Manager). Each employee has a name, employee ID, and a method to display their details. You can use constructor or setter methods to initialize the data members.
   Data Members:
   - Employee: name (String), employeeId (int)
   - Manager: teamSize (int)
   - Director: department (String)

   Member Functions:
   - Employee: displayDetails()
   - Manager: displayDetails() (overrides the base class method)
   - Director: displayDetails() (overrides the manager's method)
4. Develop an online shopping system with hierarchical inheritance. Create a base class Product and two derived classes Electronics and Clothing. Each class should have specific properties and methods. You can use constructor or setter methods to initialize the data members.
   Data Members:
   - Product: productName (String), price (double)
   - Electronics: brand (String), warrantyPeriod (int)
   - Clothing: size (String), material (String)

   Member Functions:
   - Product: displayDetails()

- Electronics: displayDetails()
- Clothing: displayDetails()

5. Create an abstract class Bank that has abstract method getROI(). Create three classes SBI, PNB, BOI inherited from Bank. Create a driver class that prints the rate of interest of each bank using super class memory reference.
6. Define an abstract class Shape with two data members sideOne and sideTwo, one parameterized constructor to display which corresponding child class object is getting created, two abstract methods setData(int, int) to set the sides and calculateArea() to calculate the area of a shape. Create two child class Rectangle and Triangle inherited from Shape and use suitable method to display their areas.
7. Develop an employee hierarchy using abstract classes. Design a base class Employee with related properties and two abstract methods calculateSalary() and displayDetails(). Derive two classes Manager and Engineer from Employee. Implement the abstract methods in the derived classes. Use constructors to initialize the data memebrs.
   Data Members:
   - Employee: name (String), employeeId (int), baseSalary (double)
   - Manager: teamSize (int), projectManaged (int)
   - Engineer: programmingLanguage (String), yearsOfExperience (int)

   Member Functions:
   - Employee: double calculateSalary(), void displayDetails() (Abstract Methods)
   - Manager:
     i. calculateSalary(): baseSalary + temSize * 1000 + projectManaged * 2000
     ii. displayDetails(): Id, Name, Base Salary, Team Size, Number of Projects Managed and Total salary
   - Engineer:
     i. calculateSalary(): baseSalary + yearsOfExperience * 500
     ii. displayDetails(): Id, Name, Programming Languages, Years of Experience, Base Salary and Total salary
8. Define an interface Calculator which has the basic methods add(), sub( ), mul() and div(). Define a concrete class named as DemoCalculator that implements the interface. Define the driver class, which create object reference of the interface Calculator and perform all basic operation of the calculator.
9. Design a banking system using Java interface and inheritance. Create an interface Account with methods displayBalance() and withdraw(). Derive two classes SavingsAccount and CurrentAccount from Account. Implement additional methods and properties for each account type.
   - Account:

        - displayBalance()
        - withdraw(double amount)
        - displayAccountDetails()
    - SavingsAccount
        - Data Members: accountNumber (String), balance (double), interestRate (double)
        - Member Functions: displayBalance(), withdraw(double amount), displayAccountDetails()
    - CurrentAccount
        - Data Members: accountNumber (String), balance (double), overdraftLimit (double)
        - Member Functions: displayBalance(), withdraw(double amount), displayAccountDetails()

10. Create an abstract class Employee with eid, name, salary and an abstract method getSalary(). Create an interface Printable with a method printDetails(). Then, create a concrete class Manager that extends Employee and implements Printable. The Manager class should implement both getSalary() and printDetails(). In the main method, create a Manager object and call both methods.