TheKiranAcademy

Problem Statement: Implementing All Object-Oriented Programming Concepts in Java

Design and develop a comprehensive Java program that encompasses all the core Object-Oriented Programming (OOP) concepts. Create a system that models a library management system, integrating various aspects of OOP including classes, objects, inheritance, polymorphism, encapsulation, and abstraction.

Requirements:

1. **Class Hierarchy and Inheritance:**
   - Define a base class `LibraryItem` with attributes:
     - `title` (String): To store the title of the item.
     - `itemID` (int): To store the unique identifier for the item.
   - Implement two subclasses `Book` and `DVD` that inherit from `LibraryItem`. Add attributes specific to each type (e.g., `author` for books and `duration` for DVDs).

2. **Polymorphism:**
   - Create a method `displayInfo()` in the base class that displays information about the library item.
   - Override this method in the subclasses to display type-specific information.

3. **Encapsulation:**
   - Use appropriate access modifiers (private, protected, public) for class attributes and methods.
   - Implement getters and setters for the attributes of `LibraryItem`, `Book`, and `DVD` classes.

4. **Abstraction:**
   - Define an abstract class `LibraryMember` with attributes:
     - `memberID` (int): To store the unique identifier for the member.
     - `name` (String): To store the name of the member.

- Include an abstract method `borrowItem(LibraryItem item)` to be implemented by subclasses.

5. **Interfaces:**

- Create an interface `Reservable` with a method `reserveItem(LibraryItem item)` that can be implemented by classes that support item reservation.

6. **Main Program:**

- In the main program, create instances of `Book` and `DVD` classes, and demonstrate inheritance, polymorphism, and encapsulation by accessing and displaying item information using the `displayInfo()` method.

- Create instances of `LibraryMember` subclasses (e.g., `StudentMember`, `FacultyMember`) and demonstrate abstraction by calling the `borrowItem()` method.

Note:

- Design the classes with appropriate attributes, methods, and relationships to showcase the OOP concepts.

- Properly handle inputs, outputs, and interactions within the program.

- Use meaningful names for classes, methods, and attributes.

- Consider scenarios like item reservation and member-item interaction to demonstrate the effectiveness of interfaces and inheritance.

This problem statement aims to provide a comprehensive exercise in implementing and demonstrating all core OOP concepts using a library management system as the context.