Java GUI Project Assignment

Simple Library Management System (LMS)

Develop a Java application with a graphical user interface using NetBeans, following the MVC architecture. This project will implement object-oriented programming concepts such as encapsulation, inheritance, polymorphism, abstraction, method overloading, aggregation, composition, and exception handling.

Features and Functional Requirements

1. Login System:

- Implement a login system that authenticates users based on predefined credentials:
    - Admin: Full access to manage members and books.
    - Member: Limited access to view and search for books.
- Create a login screen with fields for username and password.

2. Member Management:

- Add, Update, and Delete Members: Admin users can manage member information.
- View Member Details: Each member can have attributes such as memberID, name, and contactInfo.
- MembershipCard: Each member has a MembershipCard, showing a composition relationship. This class contains fields such as cardNumber and expirationDate.

3. Book Management:

- Add, Update, and Delete Books: Admin users can manage book information.
- Search Books: Provide method overloading for searching books:
- searchBooks(String title)
- searchBooks(String title, String author)
- View Book Details: Each book can have attributes such as bookID, title, author, and yearPublished.

4. Aggregation:

- Implement an aggregation relationship between Book and Member. For example, a book can be associated with one or more members who have borrowed it, without tightly coupling the two classes.

5. Exception Handling:

- Use exception handling to manage scenarios such as invalid login attempts, empty fields during member/book entry, and attempts to access features without appropriate permissions.

Additional Requirements

Architecture

- MVC Design Pattern: Use the Model-View-Controller design pattern to separate the application logic.
- Model Layer: Include classes for Member, MembershipCard, and Book.
- View Layer: Manage GUI components for login and member/book management.
- Controller Layer: Handle interactions between the Model and View layers.

Object-Oriented Programming Concepts

- Encapsulation: Use private fields with public getters and setters in all classes.
- Inheritance: Consider creating a base class User for both Admin and Member (if needed).
- Polymorphism: Implement polymorphic behavior, for example, by defining a method displayDetails() in both Member and Book classes and overriding it as necessary.
- Abstraction: Create an abstract class or interface for common behaviors, if applicable.
- Aggregation: Show how members can be associated with books without being dependent on each other.
- Composition: Use the MembershipCard class within the Member class.

Graphical User Interface

- Login Screen: A simple login interface to authenticate users.
- Main Menu: Displays features based on the user role after login.
- Screens for Member and Book Management: Allow users to add, update, delete, and search for members and books.
- Details Display: Display member or book details using the displayDetails() method.

Documentation

- UML Diagrams
    - Use Case Diagram: Represent actions available to Admin and Member roles.
    - Class Diagram: Show relationships between Member, MembershipCard, and Book.
- ER diagram
- User Manual: Describe the system's features with screenshots of the GUI and instructions for each role.

Submission Requirements

1. Source Code: Submit all Java files, organized into Model, View, and Controller packages.
2. Demonstration limited to 10 minutes.
3. Report in PDF format.
    a. UML diagrams.
    b. ER diagram
    c. User Manual: Describing the system’s features with screenshots.
4. Executable Application: Provide a runnable .jar file
5. Submission date : 13th week of the semester.

Grading Criteria

1. Functionality (20 marks): Accurate implementation of all features.
2. OOP Concepts (20 marks): Effective use of encapsulation, inheritance, polymorphism, abstraction, overloading, aggregation, and composition.
3. User Interface (10 marks): GUI design, usability, and navigation.
4. Architecture (10 marks): Adherence to MVC principles.
5. Demonstration (20 marks)
6. Documentation (20 marks): Completeness and clarity of diagrams and user manual.