Library Management System - README

Introduction

This Java program is a comprehensive Library Management System designed to manage library members, books, and facilitate various library-related operations. It's a command-line interface (CLI) application that provides functionalities for both librarians and library members.

This project was made in IntellijIde using maven to run this program I just have to play the play button

Using the Library Management System

The system supports two user roles: librarian and member. Below are instructions for both roles:

As a Librarian

Choose option 1 when prompted for your role.

As a librarian, you can perform the following functions:

- Register a Member: Register a new library member.
- Remove a Member: Remove an existing library member.
- Add a Book: Add a new book to the library.
- Remove a Book: Remove an existing book from the library.
- View all Members: View a list of all registered library members.
- View all Books: View a list of all books in the library.
- Return to the main menu: Select option 7 to return to the main menu.

As a Member

Choose option 2 when prompted for your role.

Enter your name and phone number (if you're a registered member).
As a member, you can perform the following functions:

- List Available Books: View a list of books available for borrowing.
- List My Books: View a list of books you have borrowed.
- Issue a Book: Borrow a book from the library.
- Return a Book: Return a book to the library.
- Pay Fines: Pay any fines incurred .Fine is only charged once a book is returned till then the member always has Fine=rs 0,Once He/she returns book fine is imposed.

    - Return to the main menu: Select option 6 to return to the main menu.

Features

- Member Limit: Members can borrow up to two books at a time.
- Fine Calculation: Fines are calculated based on the return time of books.
- User-Friendly Interface: The CLI interface is designed to be user-friendly and intuitive.

Exiting the Program

To exit the program, select option 3 from the main menu. This will terminate the Library Management System application.There is no other way to terminate a program if you give any value that should be an Integer and you gave an string the program will not ext it will ask you again for use

Error Handling

User Input Validation:

- The code uses input validation techniques to ensure that user-provided values (e.g., integers for main menu choices, librarian choice,member choice book IDs,age,NumberofCopies) are of the correct data type and within acceptable ranges.
- It checks if the input is an integer using hasNextInt() and provides error messages for invalid inputs. This prevents the program from crashing due to non-integer inputs.

Member Existence Check:

- When a member attempts to log in (option 2 in the menu), the program checks if the provided member name and phone number match the records in the library using the memberExists method. If the member does not exist, an error message is displayed, and the program allows the user to exit.

Book Existence Check:

- When a member tries to issue or return a book, the code checks whether the specified book exists in the library's records. If the book is not found, an error message is displayed.

Handling Member Registration and Removal:

- When registering a member (option 1 for librarians), the code checks if the member already exists based on name and phone number. If a duplicate is found, the code doesn't add the member, preventing duplicate entries.
- When removing a member (option 2 for librarians), the code checks if the specified member exists. If not, an error message is displayed.

Handling Book Addition and Removal:

  - When adding a book (option 3 for librarians), there's no explicit error handling for book addition, but the code prevents duplicate book IDs from being added.
  - When removing a book (option 4 for librarians), the code checks if the specified book exists. If not, an error message is displayed.

Handling Book Issuance and Return:
  - When a member issues a book, the code checks if the book exists and if the member has not exceeded their book limit. If the book is not found or the limit is exceeded, appropriate error messages are displayed.
  - When returning a book, the code checks if the book exists, and if it does, it calculates fines based on the time the book was issued. If the book is not found, an error message is displayed.

Handling Invalid Menu Choices:
  - Throughout the program, there are checks to ensure that menu choices are valid integers within the expected range. If an invalid choice is entered, the code displays an error message and reprompts for input.

String Types are sometime Deliberately used without error handling
- name could be supposed as ELizabeth-2
- phone is a string may be you want to give your mobile has +91,0129-

Encapsulation & Class Relationships

Member Class:
  - In the Member class, we have encapsulated member-specific data and methods. For example:
    - Private attributes like myBooks, myBookCount, fine, day are used to store member-specific information, and they are not directly accessible from outside the class.
    - Methods like listMyBooks, issueBook, returnBook, and payFine provide controlled access to these attributes and perform operations on them.

Library Class:
  - The Library class also demonstrates encapsulation. It encapsulates information related to the library's members, books, and their interactions.
  - Private attributes like fourEntryTupleArray, threeEntryTupleArray, twoEntryTupleCount, threeEntryTupleCount, member_id, and bookid are used to store library-specific information and are not directly accessible from outside the class.
  - Methods like registerMembers, removeMembers, addBooks, removeBooks, viewAllMembers, viewAllBooks, and memberExists provide controlled access to these attributes and implement the library's functionality.

Class Relationships: Class relationships define how classes interact with each other. In your code, there are two primary classes, Library and Member, and they exhibit several types of relationships:

Association:

- The Member class associates with the Library class to perform library-related operations. Members can list available books, issue books, return books, and pay fines through the Library class. This is an example of a "uses-a" relationship, where a Member uses a Library to access library services.

Composition/Aggregation:

- In the Library class, you have arrays of FourEntryTuple and ThreeEntryTuple objects to store member and book data. This can be seen as a composition or aggregation relationship, where a Library class contains or is composed of multiple instances of these tuple classes.

A video of working of Assignment 👍:

https://drive.google.com/file/d/1pwHjZAKEhBPs-kg6y54_CCzsFfTwaz10/view?usp=sharing

Thanks For reading this Readme