**\*\*\*LIBRARY MANAGEMENT SYSTEM**

**DOCUMENTATION\*\*\***

**\*\*\*1. PROJECT OVERVIEW: \*\*\***

Library Management System is a Java-based application system which is designed to manage librarians, members, and the books in the library. It provides an interface to dynamically add librarians and members, issue and return books, and handle exceptions for various error scenarios such as when a member or user is not found, or a book is already issued.

**Key Aspects:**

* **Dynamic Management:** Add multiple librarians and members without any predefined limits.
* **Book Issuing & Returning:** Allows members to borrow and return books.
* **Custom Exceptions:** Handles specific errors such as MemberNotFoundException, UserNotFoundException, and BookAlreadyIssuedException.
* **Menu-Driven Program:** The main class runs as a menu-driven program for ease of use.

**\*\*\*2. SYSTEM REQUIREMENTS: \*\*\***

To run the Library Management System, you will need the following:

* **Java Development Kit (JDK):** You need to have JDK 8 or higher installed on your computer. This allows you to run Java applications. You can download it from [Oracle's website](https://www.oracle.com/java/technologies/javase-jdk8-downloads.html).
* **Integrated Development Environment (IDE):** While optional, using an IDE makes it easier to write and run your code. Some popular IDEs for Java include:
  + IntelliJ IDEA
  + Eclipse
* **Operating System:** The application is platform-independent and can run on:
  + Windows
  + macOS
  + Linux

**\*\*\*3. USAGE: \*\*\***

* The Library Management System provides a menu-driven interface for managing library operations. Upon running the application, users will see a menu with the following options:
* **Add Librarian**: Allows the user to add librarian. No restriction on adding number of librarians.
* **Add Members:** Allows the user to add member. No restriction on adding number of members.
* **Add Book:** Allows the existing librarians to add a book to the library.
* **Remove Book:** Allows the existing librarians to remove an existing book.
* **Borrow Book:** Allows the existing member to borrow a book from the library.
* **Return Book:** Allows the existing member to return the book borrowed from the library.
* **View Books:** Allows the member to view all the available books in the library.
* **View Borrowed List:** Allows only the librarians to view the books borrowed from the library along with the borrower details.
* **Exit:** Allows the user to exit the program.

**\*Option Descriptions:**

* **Add Librarian:** This option lets you input librarian details, such as librarian ID and librarian name, to store them in the

ArrayList.

* **Add Member:** This option lets you input member details, such as member ID and member name, to add them to the ArrayList.
* **Add Book:** This option takes the librarian ID as input, if found then adds the book by taking book ID, title, author and number of copies as input.
* **Remove Book:** This option takes the librarian ID as input, if found then enables to remove book by taking the book ID as input. Else raises error message or “BookNotFoundException”.
* **Borrow Book:** This option takes the member ID as input, if found then takes book ID to borrow the book.
* **Return Book:** This option takes the member ID as input. If found, then takes the book ID to be returned as input, if found then marks the book as returned. Else raises “MemberNotFoundException” or “BookNotFoundException”.
* **View Books:** This option takes member ID input. If found, then takes librarian ID as input, if found then displays all the

available books in the library. Else raises “MemberNotFoundException” or an error message.

* **View Borrowed List:** This option takes librarian ID as input. If found, then allows viewing the books borrowed along with borrower details.
* **Exit:** The program exists after this option is chosen.

**\*\*\*4. PACKAGE STRUCTURE: \*\*\***

com.Library

MemberNotFoundException

BookNotFoundException

BookAlreadyIssuedException

Librarian

Book

Member

LibraryManagementSystem

UserActions

LibraryActions

Classes

Interfaces

CustomExceptions

**\*\* com.Library: \*\***

* The root package containing the core classes and subpackages.

A**) com.Librarian.Interfaces Description:**

**I)LibraryActions:** This interface contains viewAllBooks (), addBook (), removeBook (), viewBorrowedBooks (), findBookById () abstract methods.

**II)UserActions:** This interface contains viewAllBooks (), borrowBook (), returnBook () abstract methods.

B**) com.Librarian.Classes Description:**

I)**Book:** This class contains information about the book, such as id, title, author, availability, total copies, available copies and a list of borrowed members. It has borrowCopy (), returnCopy (), and display () methods along with appropriate getters and setters.

**II)Librarian:** This class implements LibraryActions interface and contains fields such as librarianId, name, and list of managedBooks.

**III)Member:** This class implements UserActions and contains fields such as memberId, name, and a list of borrowedBooks.

**IV)LibraryManagementSystem:** This class contains the main method and acts as a driver class by creating objects and calling appropriate methods from the required classes. The main method contains menu from which user is prompted to choose an option to experience the library management system.

**C) com.Library.CustomExceptions description:**

**I)MemberNotFoundException:** This exception is raised when the input Id isn’t matched with the Ids in the member list.

**II)BookNotFoundException:** This exception is raised when the input Id isn’t matched with the book Ids in the list.

**III)BookAlreadyIssuedException:** This exception is raised when the user retries to borrow the same book.

**\*\*\*5. HOW TO RUN THE PROJECT: \*\*\***

1. **Open Your IDE**: Launch your preferred IDE (IntelliJ IDEA, Eclipse, etc.).
2. **Import the Project**:
   * If using version control, clone the repository or download the project files.
   * Open the project folder in your IDE.
3. **Compile the Code**: Ensure all classes compile without any errors.
4. **Run the Program**:
   * Locate the LibraryManagementSystem class.
   * Execute the main method.
   * Follow the on-screen menu prompts to interact with the system.

**\*\*\*6. CONCLUSION: \*\*\***

* **Summary:** The Library Management System project is designed to efficiently manage the library operations. It allows for the dynamic addition of librarians and members, as well as the management of books, including borrowing and returning. The system incorporates user-friendly interfaces and custom exceptions to handle errors gracefully. Overall, this project not only enhances the user experience in libraries.
* **Reflection:** I got an idea of how to work on projects, got to know the documentation preparation for the project besides efficient managing of resources and organising things.