Library Management System

**Overview :**

The Library Management System (LMS) is a Java-based application designed to manage a library's book collection. The system consists of three main classes: Library, Book, and Controller, along with a user interface class named View. It provides functionalities for adding, removing, updating, and fetching details of books in the library.

**Main Features :**

Library Setup: During application startup, the librarian enters the library's name, address, and pincode, which are stored for future reference.

Book Management:

**Add Book**: The librarian can add new books to the library by providing details like book name, author, and price.

**Remove Book:** The librarian can remove books from the library by entering the book name.

**Update Book:** The librarian can update existing book information (name, author, or price) by searching for the book and then modifying the desired field.

**Get Book:** The librarian can search for a book by name and view its details (name, author, price).

**Technical Implementation :**

The application uses classes to model real-world entities:

Library: Stores information about the library (name, address, pincode) and a collection of books.

Book: Represents a book with attributes like name, author, and price.

A Controller class handles user interactions and delegates tasks to the appropriate model classes (Library and Book).

A View class manages the user interface, taking input from the librarian and displaying messages on the console.

**Limitations :**

This is a command-line application, lacking a graphical user interface (GUI).

It doesn't handle user authentication or authorization.

Error handling could be improved to provide more informative messages in case of invalid inputs or unexpected situations.

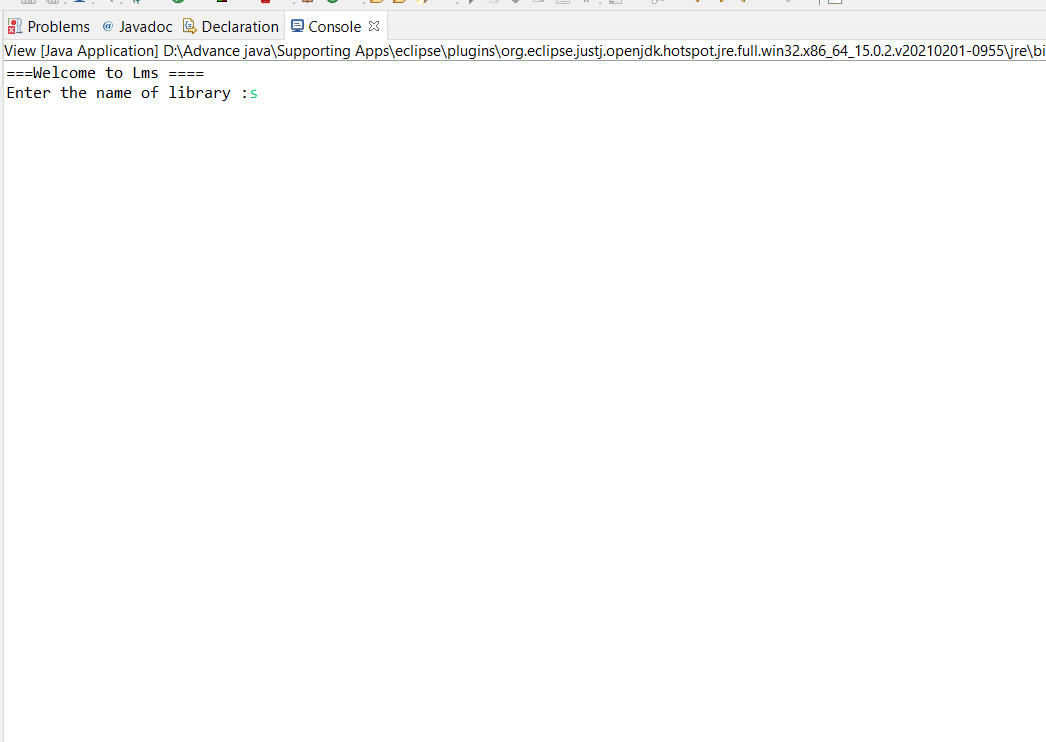
**System Requirements :**

Java Runtime Environment (JRE) installed on your computer.

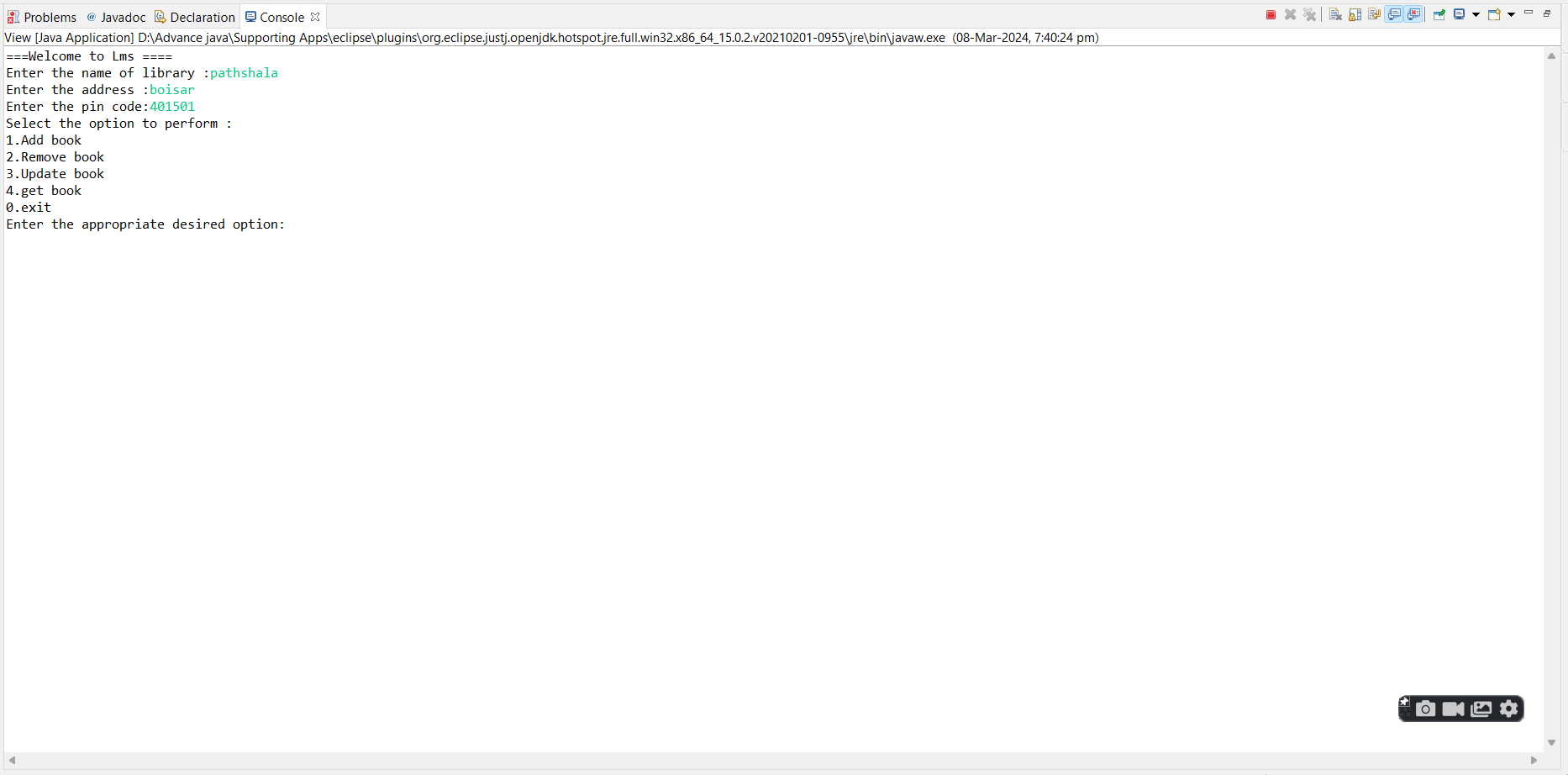
**Get Started**

**User Interaction :**

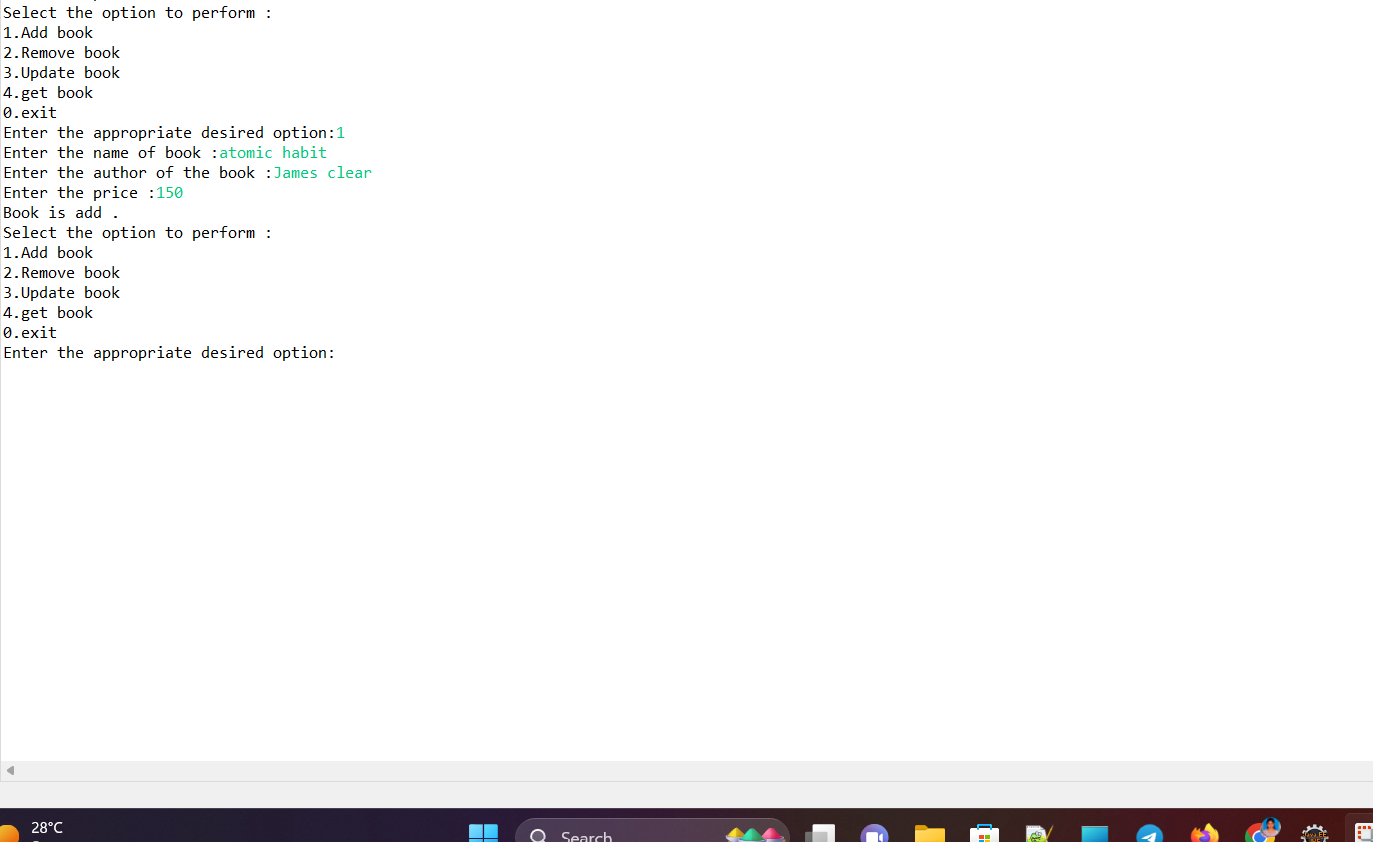
The application will greet you with a welcome message and prompt you to enter the library's details during the first run:



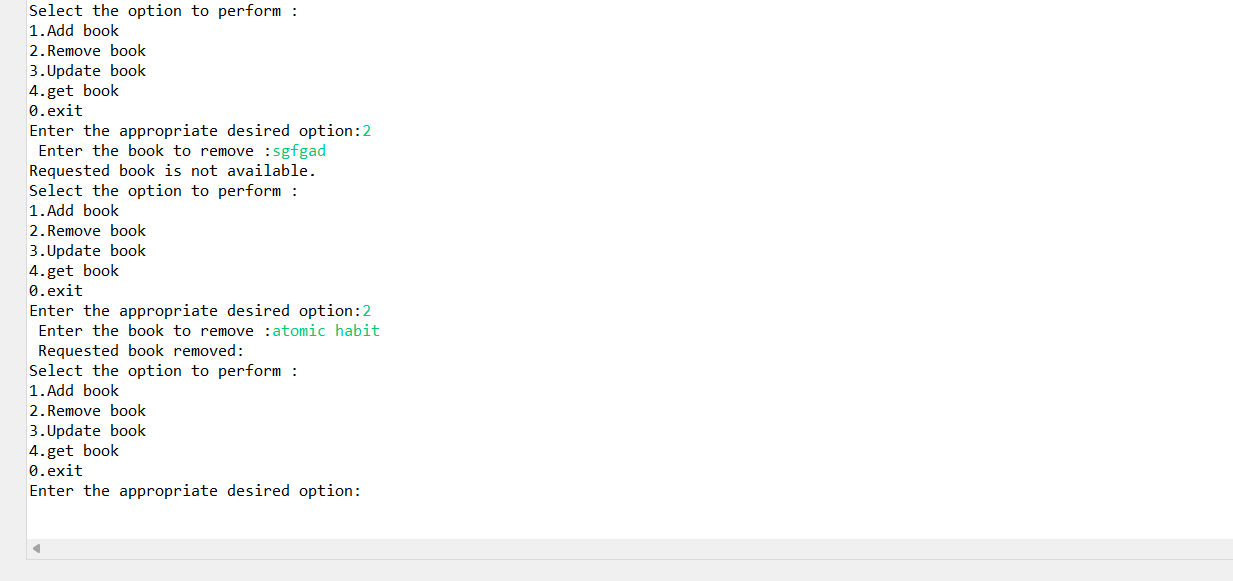
**Once the library details are set, the main menu appears:**



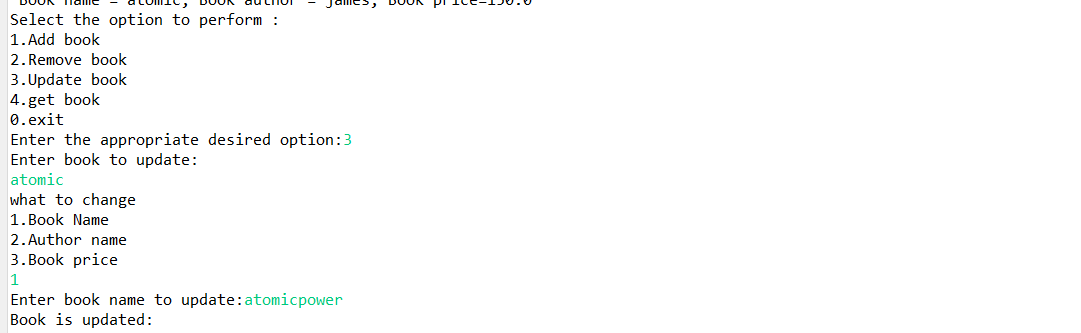
**The application will prompt you to enter details for the new book :**



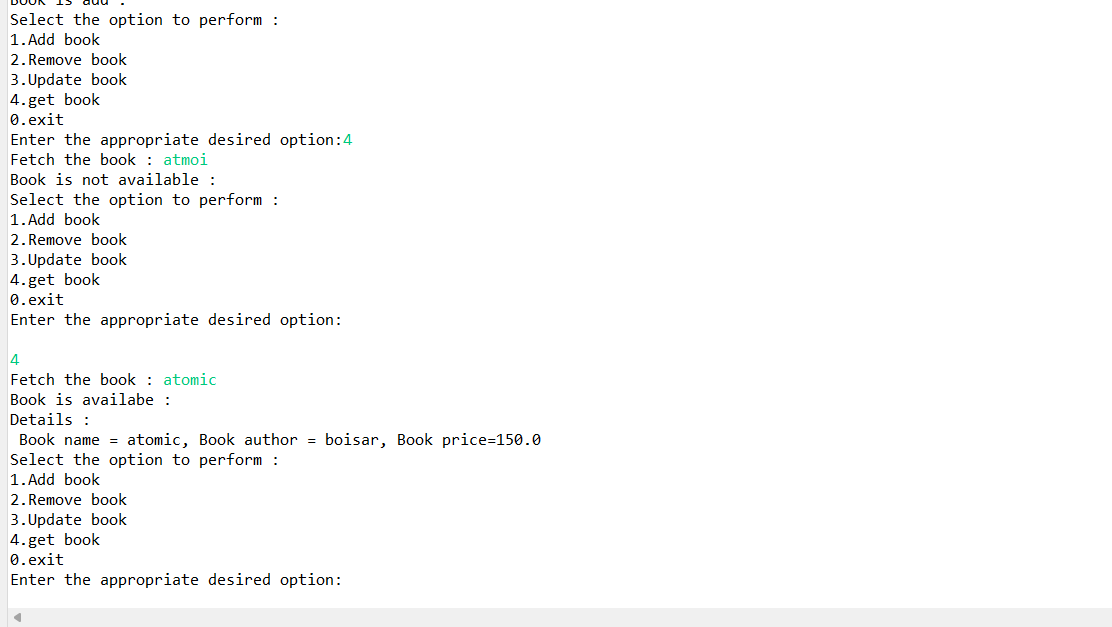
**Choose option 2: "Remove book"**

****

**Choose option 3: "Update book"**

****

**Choose option 4: "Get book" :**

****

**Data Structures:**

**Library:** Represented by the Library class.

**Fields:**

**LibraryName (String):** Name of the library.

**LibraryAddress (String):** Address of the library.

**pincode (int):** Pincode of the library's location**.**

**books (List<Book>):** Collection of books belonging to the library (relationship explained later).

**Book: Represented by the Book class.**

**Fields:**

**bookname (String):** Name of the book.

**bookauthor (String**): Author of the book.

**bookprice (double):** Price of the book**.**

**List:** Used by the Library class to store books (List<Book> books).

This is an interface implemented by various concrete list classes like ArrayList (likely used here).

It provides functionalities for adding, removing, and accessing elements in a sequential order.

Relationships:

**One-to-Many (Library and Book):**

A single Library object can have a collection of many Book objects.

This is reflected by the List<Book> books field within the Library class.

The books list stores references to individual Book objects, each representing a book in the library**.**

**Real-world examples and use cases to illustrate how this Library Management System (LMS) application can be applied in practical scenarios:**

**Small Library Management:**

A local community library can use this application to manage its collection of books. The librarian can add new books entering details like title, author, and price. They can search for existing books by title and update details if needed (e.g., update price after a discount sale). Removing books from the collection when they become damaged or are no longer relevant is also possible.

**School Library Management:**

A school librarian can utilize this application to manage the school library's book collection. Adding new books for different age groups and subjects is straightforward. Searching for specific books by title or author can help students and teachers locate resources quickly. Removing outdated or damaged books from the collection can be easily tracked.

**Personal Book Collection Management:**

Book enthusiasts can use this application to manage their personal libraries. Adding new books they acquire, including details like author and purchase price, allows for easy organization. They can search for specific books within their collection and update information if needed (e.g., mark a book as loaned to a friend).

Overall, this LMS application provides a foundational structure for managing book collections. It can be a valuable tool for small libraries, personal libraries, or educational settings where basic organization and tracking of books are needed

**Visual representation of the Library Management System's architecture :**

Start

-> User interacts with View (text-based menu)

-> Choose option:

|-> Add Book

| -> Enter book details (name, author, price)

| -> Controller adds book to Library

|-> Remove Book

| -> Enter book name

| -> Controller removes book from Library (if found)

|-> Update Book

| -> Enter book name to update

| -> Find book (Controller)

| -> If found:

| -> Choose update option (name, author, price)

| -> Update book details (Controller)

| -> Else: Book not found

|-> Get Book

| -> Enter book name

| -> Controller searches for book in Library

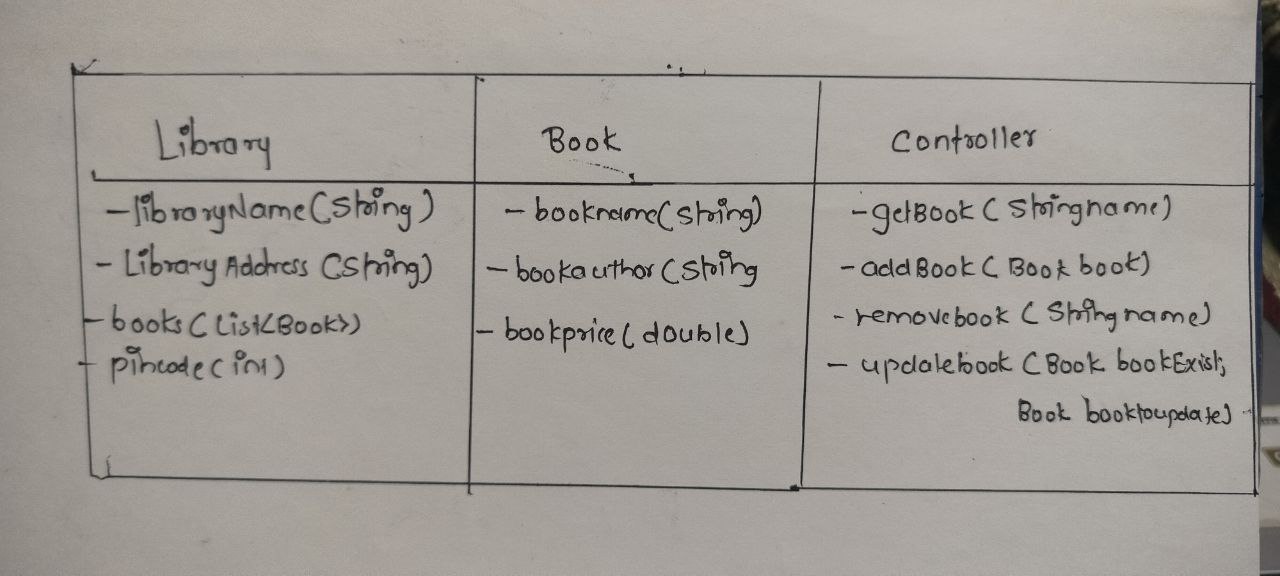
| -> If found: Display book details

| -> Else: Book not found

+-> Exit program

End

**Visual representation of the Library Management System's architecture :**



**Data Flow Example (Adding a Book):**

User enters details: The user provides information for a new book (name, author, price) through the application interface (likely a console menu in this case).

Data passes to Controller: This information is passed to the Controller class (assumed to be present).

Controller creates Book object: The Controller creates a new Book object with the provided details.

Book object added to Library: The Controller calls the addBook method of the Library class, passing the newly created Book object as an argument.

Library updates book list: The Library class adds the Book object to its internal List<Book> collection.