Library Management System

Object-Oriented Programming Project Documentation

---

1) Project Overview

Project Title: Library Management System using Java (OOP)

Project Description

The Library Management System is a console-based Java application designed to automate basic library operations such as book management, member registration, book searching, borrowing, and returning. The system follows Object-Oriented Programming principles like encapsulation, modularity, and abstraction to ensure clean and maintainable code.

Objectives

- To manage books and library members efficiently
- To implement OOP concepts in a real-world scenario
- To reduce manual work involved in library record keeping
- To provide a menu-driven, user-friendly interface

Setup Instructions (Using VS Code)

Software Requirements

- **Operating System:** Windows
- **Programming Language:** Java
- **Java Version:** JDK 8 or above (JDK 19 used)
- **IDE: Visual Studio Code (VS Code)**

Required VS Code Extensions

- **Extension Pack for Java** (by Microsoft)
    - Includes:
        - Language Support for Java™
        - Debugger for Java
        - Java Test Runner

---

2) Installation & Execution Steps (VS Code)

1. **Install Java JDK**
    - Download and install JDK

- Set JAVA_HOME and update system PATH

2. **Install Visual Studio Code**
   - Open VS Code
   - Install **Extension Pack for Java**
3. **Open Project Folder**
   - Open VS Code
   - Click **File → Open Folder**
   - Select the *Library Management System* project folder
4. **Verify Project Files**
   Ensure the following files are present:
   - Book.java
   - Member.java
   - Library.java
   - LibrarySystem.java
5. **Run the Application**
   - Open LibrarySystem.java
   - Click **Run** ▶ above the main() method
     OR
   - Right-click inside the editor → **Run Java**
6. **View Output**
   - Program output is displayed in the **VS Code Terminal / Output Console**

---

3) Code Structure

Project File Hierarchy

Library Management System

```
├── Book.java
├── Member.java
├── Library.java
└── LibrarySystem.java
```

Class Responsibilities

- **Book:** Stores book details and availability

- **Member:** Stores member details and borrowed book info
- **Library:** Handles operations like add, search, borrow, return
- **LibrarySystem:** User interface and menu control

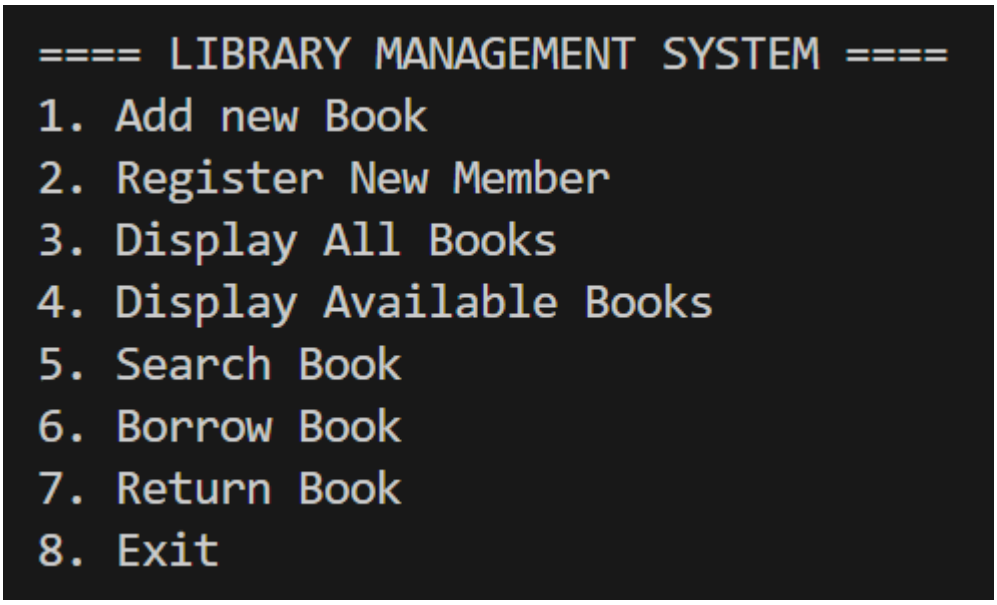
---

4) Visual Documentation

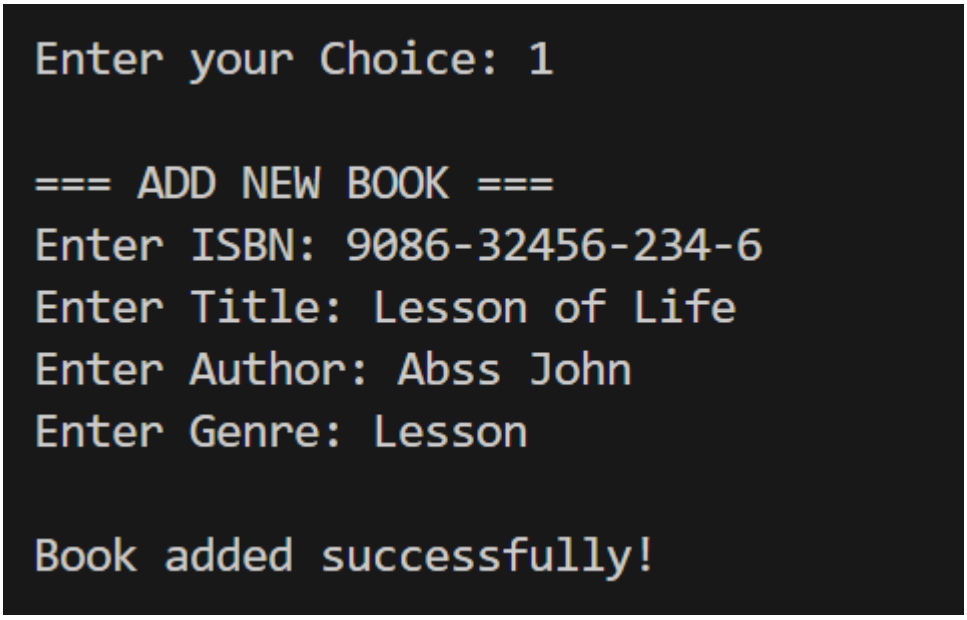
Screenshots to Include (Very Important)

You should add screenshots of:

1. Main menu display



2. Adding a new book



3. Registering a member

```
Enter your Choice: 2

=== REGISTER NEW MEMBER ===
Enter Member ID: M002
Enter Name: Samruddhi
Enter Contact: 8905432776

Member registered successfully
```

4. Displaying all books

```
=== ALL BOOKS ===
ISBN: 978-0-262-03384-8
Title: Introduction to Algorithms
Author: Thomas Cormen
Genre: Computer Science
Status: Available
----------------------------------------
ISBN: 978-0-262-03399-1
Title: Effective Java
Author: Joshua Bloch
---------
ISBN: 978-0-56619-909-4
Title: Clean Code
Author: Robert Martin
Genre: Programming
Status: Available
----------------------------------------
ISBN: 9086-32456-234-6
Title: Lesson of Life
Author: Abss John
Genre: Lesson
Status: Available
----------------------------------------
```

5. Searching books

```
Enter your Choice: 5

===== SEARCH BOOKS ====
Enter Search keyword: Code

Search Results:
ISBN: 978-0-56619-909-4
Title: Clean Code
Author: Robert Martin
Genre: Programming
Status: Available
----------------------------------------
```

6. Borrowing a book

```
Enter your Choice: 6

===== BORROW BOOK ====
Enter Member ID: M001
Enter Book ISBN: 978-0-262-03399-1

Book borrowed successfully!
Member: Alice Jhonson
Book: Effective Java
```

7. Returning a book

```
Enter your Choice: 7

===== RETURN BOOK ====
Enter Member ID: M001
Enter Book ISBN: 978-0-262-03399-1

Book returned successfully!
```

5) Technical Details

OOP Concepts Used

- **Encapsulation:**
  All class variables are private and accessed using getters/setters.
- **Abstraction:**
  Library operations are abstracted through methods.
- **Modularity:**
  Each class handles a specific responsibility.

Data Structures Used

- ArrayList<Book> → Store books
- ArrayList<Member> → Store members
- HashMap<String, LocalDate> → Track borrowed books & due dates

Algorithms

- **Linear Search** for searching books by title or author
- **Validation checks** for availability before borrowing

6) Testing Evidence

Sample Test Cases

Test Case	Input	Expected Output
Add Book	Valid book details	Book added successfully
Search Book	Keyword “Java”	Matching Java books
Borrow Book	Available book	Book borrowed
Borrow Book	Already borrowed	Error message
Return Book	Valid ISBN	Book returned successfully
Invalid Choice	Menu input 10	Invalid choice message

7) UML & System Architecture (Mandatory)

Class Diagram (Text Description)

Book

- isbn
- title
- author
- genre
- available

Member

- memberId
- name
- contact
- borrowedBooks

Library

- books
- members

LibrarySystem

- main()

System Architecture

- User interacts with **LibrarySystem**
- LibrarySystem communicates with **Library**
- Library manages **Book** and **Member** objects

---

8) Conclusion

The Library Management System successfully demonstrates the use of Object-Oriented Programming concepts in Java. The project provides a structured, scalable, and user-friendly solution for basic library operations. It can be further enhanced by adding features like fine calculation, database integration, and graphical user interface.

---

9) Future Enhancements

- Fine calculation for late returns

- Database (MySQL) integration
- GUI using JavaFX or Swing
- Admin and User roles