**Library Management System**

**1. Abstract**

The Library Management System is a console-based Java program designed to manage books and members efficiently.It allows users to add books, register members, borrow and return books, search books, and view library statistics.The program also supports saving and loading data, ensuring information is preserved between sessions.This project demonstrates object-oriented programming, file handling,and basic data management concepts in Java.

**2. Introduction**

Libraries are essential for storing and managing books systematically.  
Manually maintaining a library can be **time-consuming and error-prone**.  
The **Library Management System in Java** automates the process and provides a **user-friendly console interface** for library management.

**Key Features:**

* Add, display, and search books (by ID, title, or author)
* Register members and track borrowed books
* Borrow and return books with borrow limits and due date fine calculation
* View statistics such as total books, borrowed books, and total members
* Persistent storage using files

**3. Existing System**

Manual library management using paper records and ledgers.

**Problems:**

* Time-consuming to record and search books manually
* Difficult to track which books are borrowed or returned
* Error-prone due to human mistakes
* No easy way to calculate fines for late returns

**4 .Proposed System**

Automated Library Management System in Java.

**Advantages:**

* Add, display, search, borrow, and return books efficiently
* Maintain a member database to track who borrowed which book
* Enforce borrow limits and calculate fines automatically
* Save and load data from files for persistent storage
* Reduces manual errors and saves time

**5. Software Requirements**

* **Operating System:** Windows 7/8/10 or Linux
* **Programming Language:** Java (JDK 8 or higher)
* **IDE / Editor:** VS Code, Eclipse, or Notepad++
* **Compiler / Runtime:** Java JDK
* **Others:** Command Prompt / Terminal

**6.Hardware Requirements**

* **Processor:** Minimum Intel i3 or equivalent
* **RAM:** Minimum 4 GB
* **Storage:** Minimum 500 MB free space
* **Display:** Monitor with at least 1024x768 resolution
* **Others:** Keyboard, Mouse, Internet (optional)

**6.Program:**

import java.io.\*;

import java.time.\*;

import java.time.temporal.ChronoUnit;

import java.util.\*;

class Book implements Serializable {

int id;

String title;

String author;

String category;

boolean isAvailable = true;

LocalDate borrowDate;

int borrowedBy = -1; // member ID

Book(int id, String title, String author, String category) {

this.id = id;

this.title = title;

this.author = author;

this.category = category;

}

void displayBook() {

System.out.printf("ID: %d | Title: %s | Author: %s | Category: %s | Available: %s\n",

id, title, author, category, isAvailable ? "Yes" : "No");

}

}

class Member implements Serializable {

int id;

String name;

ArrayList<Integer> borrowedBooks = new ArrayList<>();

Member(int id, String name) {

this.id = id;

this.name = name;

}

void displayMember() {

System.out.println("Member ID: " + id + " | Name: " + name + " | Books Borrowed: " + borrowedBooks.size());

}

}

public class LibraryManagementSystem {

static ArrayList<Book> library = new ArrayList<>();

static ArrayList<Member> members = new ArrayList<>();

static Scanner sc = new Scanner(System.in);

static final String BOOK\_FILE = "books.dat";

static final String MEMBER\_FILE = "members.dat";

public static void main(String[] args) {

loadData();

if (!adminLogin()) return;

int choice;

do {

System.out.println("\n=== LIBRARY MANAGEMENT SYSTEM ===");

System.out.println("1. Add Book");

System.out.println("2. Display All Books");

System.out.println("3. Search Book (by ID/Title/Author)");

System.out.println("4. Add Member");

System.out.println("5. Display Members");

System.out.println("6. Borrow Book");

System.out.println("7. Return Book");

System.out.println("8. View Statistics");

System.out.println("9. Save & Exit");

System.out.print("Enter your choice: ");

choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1 -> addBook();

case 2 -> displayBooks();

case 3 -> searchBook();

case 4 -> addMember();

case 5 -> displayMembers();

case 6 -> borrowBook();

case 7 -> returnBook();

case 8 -> showStatistics();

case 9 -> {

saveData();

System.out.println("✅ Data saved. Exiting...");

}

default -> System.out.println("❌ Invalid choice! Try again.");

}

} while (choice != 9);

}

// ---------- ADMIN LOGIN ----------

static boolean adminLogin() {

System.out.println("=== ADMIN LOGIN ===");

System.out.print("Username: ");

String user = sc.nextLine();

System.out.print("Password: ");

String pass = sc.nextLine();

if (user.equals("admin") && pass.equals("1234")) {

System.out.println("✅ Login successful!");

return true;

} else {

System.out.println("❌ Invalid credentials!");

return false;

}

}

// ---------- BOOK MANAGEMENT ----------

static void addBook() {

System.out.print("Enter Book ID: ");

int id = sc.nextInt(); sc.nextLine();

System.out.print("Enter Title: ");

String title = sc.nextLine();

System.out.print("Enter Author: ");

String author = sc.nextLine();

System.out.print("Enter Category: ");

String category = sc.nextLine();

library.add(new Book(id, title, author, category));

System.out.println("📚 Book added successfully!");

}

static void displayBooks() {

if (library.isEmpty()) {

System.out.println("❌ No books available.");

return;

}

System.out.println("\n--- Book List ---");

for (Book b : library) b.displayBook();

}

static void searchBook() {

System.out.println("Search by: 1.ID 2.Title 3.Author");

int opt = sc.nextInt(); sc.nextLine();

boolean found = false;

switch (opt) {

case 1 -> {

System.out.print("Enter Book ID: ");

int id = sc.nextInt();

for (Book b : library)

if (b.id == id) { b.displayBook(); found = true; }

}

case 2 -> {

System.out.print("Enter Title keyword: ");

String key = sc.nextLine().toLowerCase();

for (Book b : library)

if (b.title.toLowerCase().contains(key)) { b.displayBook(); found = true; }

}

case 3 -> {

System.out.print("Enter Author keyword: ");

String key = sc.nextLine().toLowerCase();

for (Book b : library)

if (b.author.toLowerCase().contains(key)) { b.displayBook(); found = true; }

}

}

if (!found) System.out.println("❌ No matching books found.");

}

// ---------- MEMBER MANAGEMENT ----------

static void addMember() {

System.out.print("Enter Member ID: ");

int id = sc.nextInt(); sc.nextLine();

System.out.print("Enter Member Name: ");

String name = sc.nextLine();

members.add(new Member(id, name));

System.out.println("👤 Member added successfully!");

}

static void displayMembers() {

if (members.isEmpty()) {

System.out.println("❌ No members available.");

return;

}

System.out.println("\n--- Member List ---");

for (Member m : members) m.displayMember();

}

// ---------- BORROW & RETURN ----------

static void borrowBook() {

System.out.print("Enter Member ID: ");

int mid = sc.nextInt();

Member mem = findMember(mid);

if (mem == null) {

System.out.println("❌ Member not found!");

return;

}

if (mem.borrowedBooks.size() >= 3) {

System.out.println("⚠️ Borrow limit reached (3 books max).");

return;

}

System.out.print("Enter Book ID to borrow: ");

int bid = sc.nextInt();

Book book = findBook(bid);

if (book == null) {

System.out.println("❌ Book not found!");

return;

}

if (!book.isAvailable) {

System.out.println("❌ Book already borrowed!");

return;

}

book.isAvailable = false;

book.borrowDate = LocalDate.now();

book.borrowedBy = mid;

mem.borrowedBooks.add(bid);

System.out.println("✅ " + mem.name + " borrowed: " + book.title);

}

static void returnBook() {

System.out.print("Enter Member ID: ");

int mid = sc.nextInt();

Member mem = findMember(mid);

if (mem == null) {

System.out.println("❌ Member not found!");

return;

}

System.out.print("Enter Book ID to return: ");

int bid = sc.nextInt();

Book book = findBook(bid);

if (book == null || book.isAvailable) {

System.out.println("❌ Invalid book ID or book not borrowed.");

return;

}

long days = ChronoUnit.DAYS.between(book.borrowDate, LocalDate.now());

double fine = (days > 7) ? (days - 7) \* 5 : 0;

if (fine > 0)

System.out.println("⚠️ Late return! Fine: ₹" + fine);

book.isAvailable = true;

book.borrowedBy = -1;

mem.borrowedBooks.remove(Integer.valueOf(bid));

System.out.println("✅ Book returned successfully!");

}

// ---------- STATISTICS ----------

static void showStatistics() {

long total = library.size();

long borrowed = library.stream().filter(b -> !b.isAvailable).count();

long available = total - borrowed;

System.out.println("\n--- Library Statistics ---");

System.out.println("Total Books: " + total);

System.out.println("Available Books: " + available);

System.out.println("Borrowed Books: " + borrowed);

System.out.println("Total Members: " + members.size());

}

// ---------- HELPER FUNCTIONS ----------

static Book findBook(int id) {

for (Book b : library) if (b.id == id) return b;

return null;

}

static Member findMember(int id) {

for (Member m : members) if (m.id == id) return m;

return null;

}

// ---------- SAVE / LOAD ----------

static void saveData() {

try (ObjectOutputStream o1 = new ObjectOutputStream(new FileOutputStream(BOOK\_FILE));

ObjectOutputStream o2 = new ObjectOutputStream(new FileOutputStream(MEMBER\_FILE))) {

o1.writeObject(library);

o2.writeObject(members);

} catch (Exception e) { e.printStackTrace(); }

}

static void loadData() {

try (ObjectInputStream i1 = new ObjectInputStream(new FileInputStream(BOOK\_FILE));

ObjectInputStream i2 = new ObjectInputStream(new FileInputStream(MEMBER\_FILE))) {

library = (ArrayList<Book>) i1.readObject();

members = (ArrayList<Member>) i2.readObject();

System.out.println("✅ Data loaded successfully!");

} catch (Exception e) {

System.out.println("⚠️ No previous data found, starting fresh.");

}

}

}

**7.Sample Output:**

=== LIBRARY MANAGEMENT SYSTEM ===

1. Add Book

2. Display All Books

3. Search Book (by ID/Title/Author)

4. Add Member

5. Display Members

6. Borrow Book

7. Return Book

8. View Statistics

9. Save & Exit

Enter your choice: 4

Enter Member ID: 101

Enter Member Name: Sathvika Merugu

👤 Member added successfully!

Enter your choice: 1

Enter Book ID: 201

Enter Title: Java Programming

Enter Author: James Gosling

Enter Category: Programming

📚 Book added successfully!

Enter your choice: 6

Enter Member ID: 101

Enter Book ID to borrow: 201

✅ Sathvika Merugu borrowed: Java Programming

Enter your choice: 2

--- Book List ---

ID: 201 | Title: Java Programming | Author: James Gosling | Category: Programming | Available: No

Enter your choice: 7

Enter Member ID: 101

Enter Book ID to return: 201

✅ Book returned successfully!

Enter your choice: 8

--- Library Statistics ---

Total Books: 1

Available Books: 1

Borrowed Books: 0

Total Members: 1

**Conclusion**

The Library Management System project demonstrates the use of Java programming concepts to create a functional library application.  
It automates tasks like adding books, managing members, borrowing and returning books, and provides persistent data storage.  
This project enhances understanding of object-oriented programming, file handling, and data management, while providing a practical solution for library management efficiently.