Library Management System

Fletcher Baccus, Cheryl Twyman, Michael Lawyer

COSC 112

December 5, 2024

Project Description

- Our project is a Library Management System designed to manage book borrowing, member registration, and overdue notifications.
- Core Features Include:

1. Registering up to 10 members.
2. Searching catalog for books viewing their availability.
3. Borrowing and returning books with updates of availability.
4. Tracking overdue books and notifying registered members.

- Tools Used: Java, GitHub, and teamwork. Steps included planning, coding, and testing.

Key Challenges

- 1. Managing book availability: Used boolean arrays to track availability.
- 2. Handling user input: Implementing Input validation.
- 3. Integrating features: Combined what was learned over this course as well as some insight of coding before this class to create functionality.
- 4. Teamwork coordination: Meetings both in person and Online over short period of time throughout this week ☹ although we ensured effective collaboration.

Mid-Project Check-In

- Progress Report:

1. Implemented Features Such as:
   -Member registration, book borrowing, and search functionality completed.
2. Challenges Encountered:
   -Handling cases where a book is already borrowed.
   Solved(with “null” which means if the members index in “borrowedBooks” is “null” means that book hasn’t been borrowed yet allowing a member to only check out one of that book if not borrowed yet.
3. Next Step: Complete overdue notifications and final testing by the deadline.

Program Functionality

Register Member:

```java
static String[] members = new String[10]; // Member database. The max is 10 members.
static int memberCount = 0; // Track the number of registered members
```

```java
//Members are stored in an array, with up to 10 members allowed.
if (memberCount < members.length) {
    System.out.print("Enter member name: ");
    String memberName = scanner.nextLine();
    members[memberCount] = memberName;
    borrowedBooks[memberCount] = null;
    memberCount++;
    System.out.println("Member registered successfully!");
}
//Logic: Ensures no more than 10 members can register.
```

Search Books:

```java
//Searches for a book by title and checks availability.

for (int i = 0; i < books.length; i++) {
    if (books[i].equalsIgnoreCase(bookTitle)) {
        System.out.println("Book found: " + books[i] +
            " - " + (isBookAvailable[i] ? "Available" : "Not Available"));
        break;
    }
}
//Logic: Loops through the books array and displays the book's availability.
```

Borrow Books:

```
  // Allows members to borrow books, provided they have none already borrowed.

if (borrowedBooks[memberIndex] == null) {
    System.out.print("Enter the title of the book to borrow: ");
    String bookTitle = scanner.nextLine();
    int bookIndex = findBookIndex(bookTitle);

    if (bookIndex != -1 && isBookAvailable[bookIndex]) {
        isBookAvailable[bookIndex] = false;
        borrowedBooks[memberIndex] = books[bookIndex];
        System.out.println("Book borrowed successfully!");
    } else {
        System.out.println("Book is either not available or already borrowed.");
    }
}
//Logic: Checks if the member already has a book borrowed and ensures the requested book is available.
```

```java
  // Allows members to borrow books, provided they have none already borrowed.

if (borrowedBooks[memberIndex] == null) {
    System.out.print("Enter the title of the book to borrow: ");
    String bookTitle = scanner.nextLine();
    int bookIndex = findBookIndex(bookTitle);

    if (bookIndex != -1 && isBookAvailable[bookIndex]) {
        isBookAvailable[bookIndex] = false;
        borrowedBooks[memberIndex] = books[bookIndex];
        System.out.println("Book borrowed successfully!");
    } else {
        System.out.println("Book is either not available or already borrowed.");
    }
}
//Logic: Checks if the member already has a book borrowed and ensures the requested book is available.
```

Return Books:

```
//Loops: Used to repeat through books and members.

//Example: Searching the book catalog.

for (int i = 0; i < books.length; i++) {
    if (books[i].equalsIgnoreCase(bookTitle)) {
        System.out.println("Book found.");
        break;
    }
}

//Arrays: Store data for books, members, and borrowing information.

java
Copy code
static String[] books = {"Bleach", "One Piece", "Naruto", "Dragon Ball"};
static boolean[] isBookAvailable = {true, true, true, true};

//Conditional Statements: Check availability and handle edge cases.


if (borrowedBooks[memberIndex] == null) {
    // Proceed with borrowing
} else {
    System.out.println("You already have a borrowed book.");
}
```

Code Breakdown

User Interface

```
    //Example of Main Menu:


System.out.println("\n=== Library Management System ===");
System.out.println("1. Register Member");
System.out.println("2. Search for a Book");
System.out.println("3. Borrow a Book");
System.out.println("4. Return a Book");
System.out.println("5. Check Overdue Notifications");
System.out.println("6. Exit");
System.out.print("Enter your choice: ");
```

- Definition: The way people interact with system information

Allows users to:

- Register as a library member.
- Search for books.
- Borrow and return books.
- Check Overdue Notifications
- And Exit

```java
// 1. For Loop (Example: Overdue Notifications):
//- Used to repeat over the list of members.
// - Checks each member to see if they have overdue books.

// Code Example:
for (int i = 0; i < memberCount; i++) {
    if (borrowedBooks[i] != null) {
        System.out.println("Member: " + members[i] + ", Overdue Book: " + borrowedBooks[i]);
    }
}

// - Ensures that all records are checked efficiently.

2. Do-While Loop (Example: Main Menu):
//- Executes the menu options repeatedly until the user chooses to exit.

//Code Example:
do {
    System.out.println("Enter your choice: ");
    choice = scanner.nextInt();
} while (choice != 6);

// Guarantees the menu displays at least once.
```

Adding the Loops:

```
   //Switch Case Overview:
  // - Manages the user's menu choices.
  // - Simplifies control flow by mapping each choice to a action.

//Code Example:
switch (choice) {
    case 1: registerMember(scanner); break;
    case 2: searchBook(scanner); break;
    case 3: borrowBook(scanner); break;
    case 4: returnBook(scanner); break;
    case 5: checkOverdueNotifications(); break;
    case 6: System.out.println("Goodbye!"); break;
    default: System.out.println("Invalid choice!");
}

//Purpose:
   //- Maps menu choices to specific methods.
   //- Handles invalid input.
```

Adding the Switch Case:

```
=== Library Management System ===
1. Register Member
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Check Overdue Notifications
6. Exit
Enter your choice: 3

=== Library Book Borrowing ===
Enter your member name: Fletch101
Member not found. Please register first.

=== Library Management System ===
1. Register Member
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Check Overdue Notifications
6. Exit
Enter your choice: 1

=== Library Member Registration ===
Enter member name: Fletch101
Member registered successfully!
```

```
Enter your choice: 2

=== Library Book Search ===
Enter book title to search: one piece
Book found: One Piece - Available

=== Library Management System ===
1. Register Member
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Check Overdue Notifications
6. Exit
Enter your choice: 3

=== Library Book Borrowing ===
Enter your member name: Fletch101
Enter the title of the book to borrow: One Piece
Book borrowed successfully!

=== Library Management System ===
1. Register Member
2. Search for a Book
3. Borrow a Book
4. Return a Book
5. Check Overdue Notifications
6. Exit
Enter your choice: 6
Thank you for using the Library Management System. Goodbye!


...Program finished with exit code 0
Press ENTER to exit console.
```

Output

It will output numbers 1- 6 with each number carrying out different task depending on the users chosen number between 1-6. In the first Screenshot Fletcher tried borrowing a book but it wouldn’t allow him since he hasn’t registered yet. So it kindly asks him to register before trying to borrow a book. He then registers which allows him to check out the book One Piece.

Lessons Learned

- 1. Technical Skills: Improved knowledge of Java arrays, loops, and methods.
- 2. Problem Solving: solved challenges like data management and input validation.
- 3. Teamwork: Strengthened collaboration and communication skills.

Future Improvements

Enhanced Features:

- Such as implementing user authentication and notifications within other projects.

Scalability:

- Expand the system to support larger libraries with more data.

Conclusion

- Our Library Management System simplifies library operations by handling member registrations, book borrowing, and overdue tracking.
- We learned valuable technical and teamwork skills while completing this project.

?QUESTIONS?