Lab: Library Management System

Overview

In this lab, you will build a simple Library Management System step by step. Each task will guide you through implementing and modifying the system while reinforcing Java concepts like iteration, conditions, classes, and objects.

Scenario

Imagine you are designing a system for a small library. The library has books, each with a title, an author, and an availability status (available or borrowed). Members of the library can borrow books, return books, and view the list of available books. The system should handle these operations interactively.

Task 1: Create the Book Class

Objective: Define the blueprint for a book.

1. Create a `Book` class with the following attributes:
    - `String title`
    - `String author`
    - `boolean isAvailable` (default value: `true`)
2. Add a constructor to initialize the title and author. Set `isAvailable` to `true` by default.
3. Add the following methods:
    - `getTitle()`: Returns the title of the book.
    - `getAuthor()`: Returns the author of the book.
    - `isAvailable()`: Returns the availability status.
    - `borrowBook()`: Sets `isAvailable` to `false` if the book is available; otherwise, displays an error message.
    - `returnBook()`: Sets `isAvailable` to `true`.

Task 2: Create the Library Class

Objective: Manage a collection of books.

1. Create a `Library` class with the following attributes:
   - `Book[] books` (fixed size, e.g., 10 books)
   - `int bookCount` (to track the number of books added)
2. Add a method `addBook(Book book)` to add a book to the library. Ensure you don't exceed the array's size.
3. Add a method `listAvailableBooks()` to print the titles and authors of all available books.
4. Add a method `findBook(String title)` to search for a book by its title. Return the `Book` object if found or `null` otherwise.

Task 3: Add User Interaction

Objective: Create a menu-driven system to interact with the library.

1. Create a `LibraryApp` class with a `main` method.
2. In the `main` method:
   - Instantiate a `Library` object.
   - Add some books to the library.
   - Use a `Scanner` to create a menu with the following options:
     1. List all available books.
     2. Borrow a book.
     3. Return a book.
     4. Exit.
3. Implement a loop that allows the user to choose an option and perform the corresponding action.

Example Menu:

```
1. List available books
2. Borrow a book
3. Return a book
4. Exit
Enter your choice:
```

Task 4: Add Borrowing and Returning Functionality

Objective: Enable members to borrow and return books.

1. Update the `Library` class with:
   - `borrowBook(String title)`: Borrow a book by its title. Use `findBook` to locate the book and call `borrowBook()`.
   - `returnBook(String title)`: Return a book by its title. Use `findBook` to locate the book and call `returnBook()`.

Task 5: Enhance the System (Optional)

Objective: Add more functionality to the system.

1. Add a `Member` class to represent library members. Include attributes like `String name` and `Book[] borrowedBooks` (fixed size, e.g., 5 books per member).
2. Modify the `Library` class to handle multiple members. Track which member has borrowed which book.
3. Add an option to register a new member and view a member's borrowed books in the menu.

Expected Outcomes

By the end of this lab, you should have:

1. A `Book` class to represent individual books.
2. A `Library` class to manage the collection of books using arrays.
3. A menu-driven program to allow users to interact with the library system.
4. (Optional) A more complex system with members and advanced features.

Code Use Examples:

Task 1 Example – Use of Book class:

```
Book book1 = new Book("1984", "George Orwell");
System.out.println(book1.getTitle() + " by " + book1.getAuthor());
System.out.println("Available: " + book1.isAvailable());
book1.borrowBook();
System.out.println("Available: " + book1.isAvailable());
```

Task 2 Example – Use of Library class:

```
Library library = new Library();
Book book1 = new Book("1984", "George Orwell");
Book book2 = new Book("To Kill a Mockingbird", "Harper Lee");
library.addBook(book1);
library.addBook(book2);
library.listAvailableBooks();
Book searchResult = library.findBook("1984");
if (searchResult != null) {
    System.out.println("Found: " + searchResult.getTitle());
}
```

Task 4 Example – Use of Library class – borrow and return:

```
library.borrowBook("1984");
library.listAvailableBooks();
library.returnBook("1984");
library.listAvailableBooks();
```