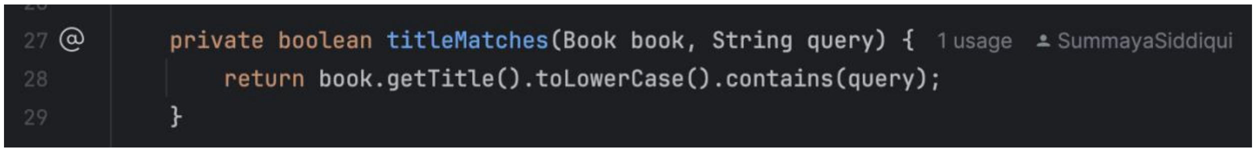
Project Documentation: Library Management System

1. Code Quality and Clean Code Practices

Example 1: Meaningful Naming Conventions

Code Snippet:

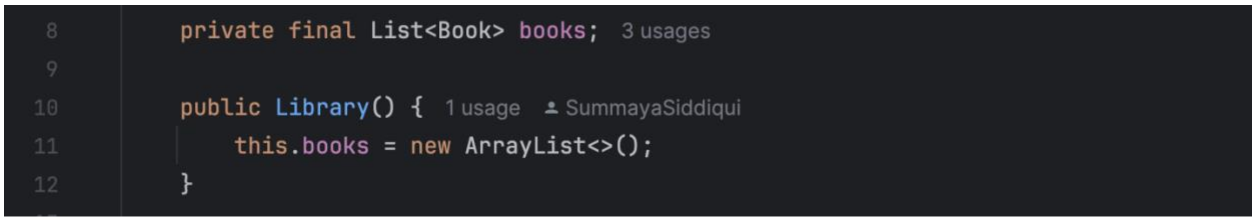


```
private boolean titleMatches(Book book, String query) {   1 usage   SummayaSiddiqui
    return book.getTitle().toLowerCase().contains(query);
}
```

Explanation: The method name `titleMatches` clearly describes its function. The parameters are also well-named to indicate their purpose. This makes the code easy to read and maintain.

Example 2: Encapsulation and Data Hiding

Code Snippet:



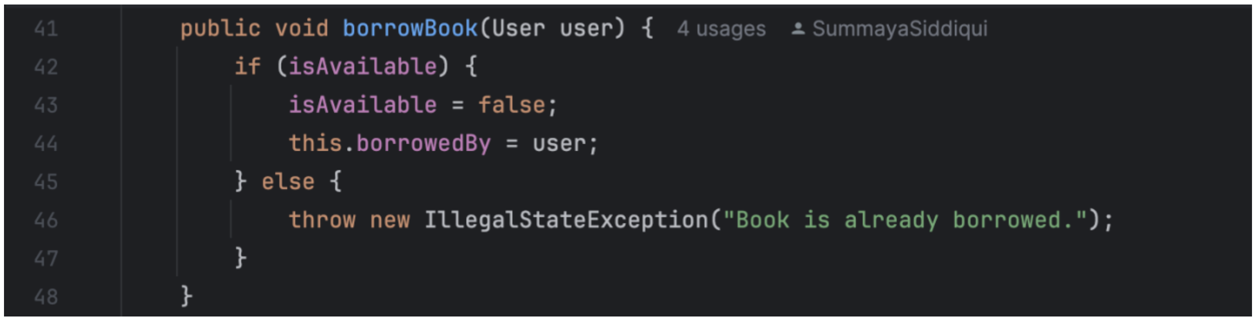
```
private final List<Book> books;   3 usages

public Library() {   1 usage   SummayaSiddiqui
    this.books = new ArrayList<>();
}
```

Explanation: The `books` list is declared `private final`, ensuring it cannot be accessed directly outside the `Library` class. This enforces encapsulation and prevents unintended modifications.

Example 3: Meaningful Exception Handling

Code Snippet:



```
public void borrowBook(User user) {   4 usages   SummayaSiddiqui
    if (isAvailable) {
        isAvailable = false;
        this.borrowedBy = user;
    } else {
        throw new IllegalStateException("Book is already borrowed.");
    }
}
```

Explanation: Instead of using generic errors, an `IllegalStateException` with a clear message is thrown. This makes debugging easier and improves maintainability.

2. Project Overview

What the Project Does

The Library Management System allows users to:

- Borrow and return books.
- Search for books by title, author, or ISBN.
- Enforce a borrowing limit of three books per user.

How It Works

- The `User` class manages user information and borrowed books.
- The `Book` class represents a book with title, author, and ISBN.
- The `Library` class stores a collection of books and provides search functionality.

Test Cases Used

Book Tests

- Testing the book availability.
- Ensuring that an already borrowed book is not borrowed by anyone.
- Preventing the return of a book which is never borrowed.

User Tests

- Ensuring a user can borrow up to 3 books.
- Checking that exceeding the borrowing limit throws an exception.
- Verifying that a user can return a borrowed book.
- Ensuring a user cannot borrow the same book twice.
- Preventing users from returning a book they never borrowed.

Library Tests

- Searching by title, author, and ISBN.
- Handling searches for non-existent books.

3. Dependencies

Required Dependencies

- **JUnit 5** (for testing): `org.junit.jupiter:junit-jupiter:5.11.4`
- **Java Development Kit (JDK)** 23

4. Challenges Faced During QAP

Problem 1: Borrowing the Same Book Twice

Issue: Initially, a user could borrow the same book multiple times. **Solution:** Implemented a check in `borrowBook()` to throw an exception if the book is already borrowed.

Problem 2: Case-Sensitive Search

Issue: Searching for books was case-sensitive, leading to inconsistent results. **Solution:** Converted search queries and book attributes to lowercase for uniform matching.

Problem 3: Incorrect Test Assertion

Issue: A test case checking for ISBN search was failing due to a typo in the assertion message. **Solution:** Fixed the assertion message and ensured it matched expected output.