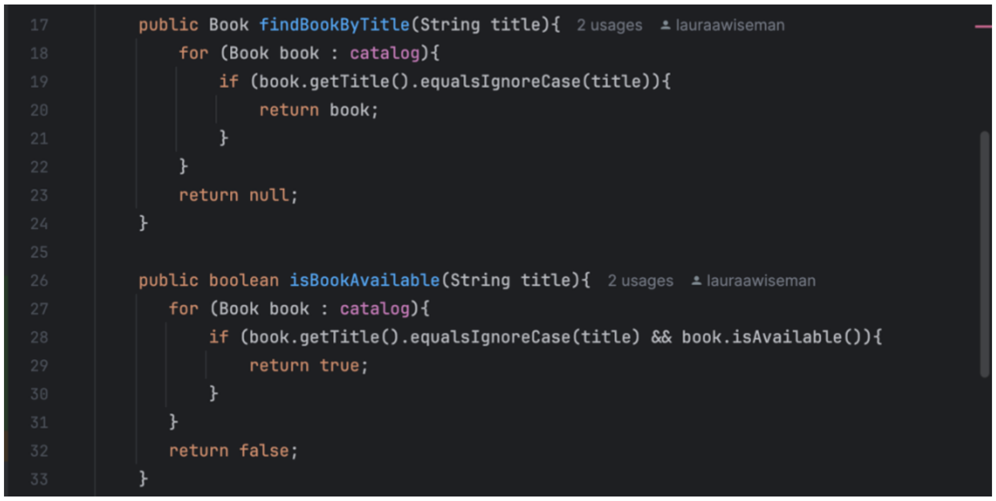
SDAT AND DEV OPS COMBINED QAP 1

SD 12

Author: Laura Wiseman
Date: June 1st, 2025

1. Clean Code Explanations

Library.java

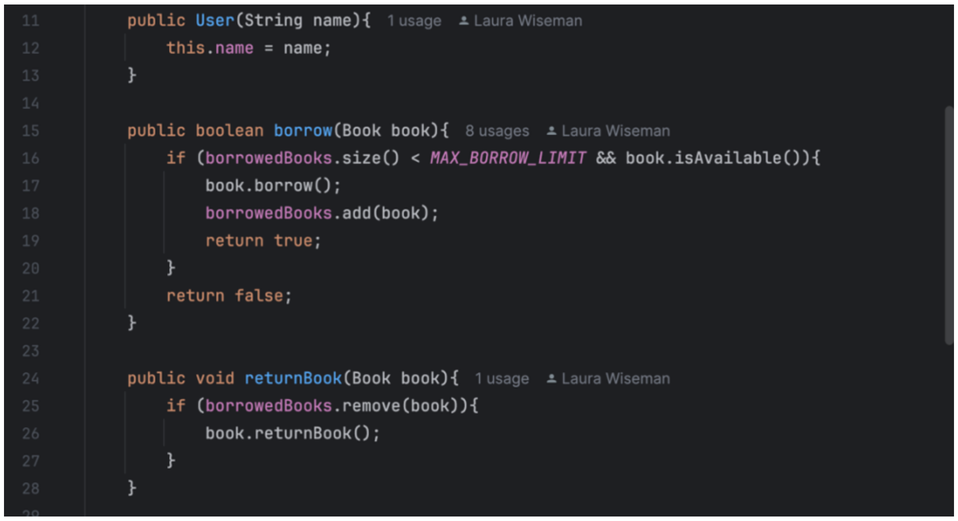


```java
public Book findBookByTitle(String title){  2 usages  lauraawiseman
    for (Book book : catalog){
        if (book.getTitle().equalsIgnoreCase(title)){
            return book;
        }
    }
    return null;
}

public boolean isBookAvailable(String title){  2 usages  lauraawiseman
   for (Book book : catalog){
       if (book.getTitle().equalsIgnoreCase(title) && book.isAvailable()){
           return true;
       }
   }
   return false;
}
```

Library.java: The findBookByTitle and isBookAvailable methods follow clean code principles through clear naming, consistent layout and direct logic. Both methods performs one task clearly, avoids extra variables, and uses early returns for simplicity, making the code easy to read and maintain.

User.java



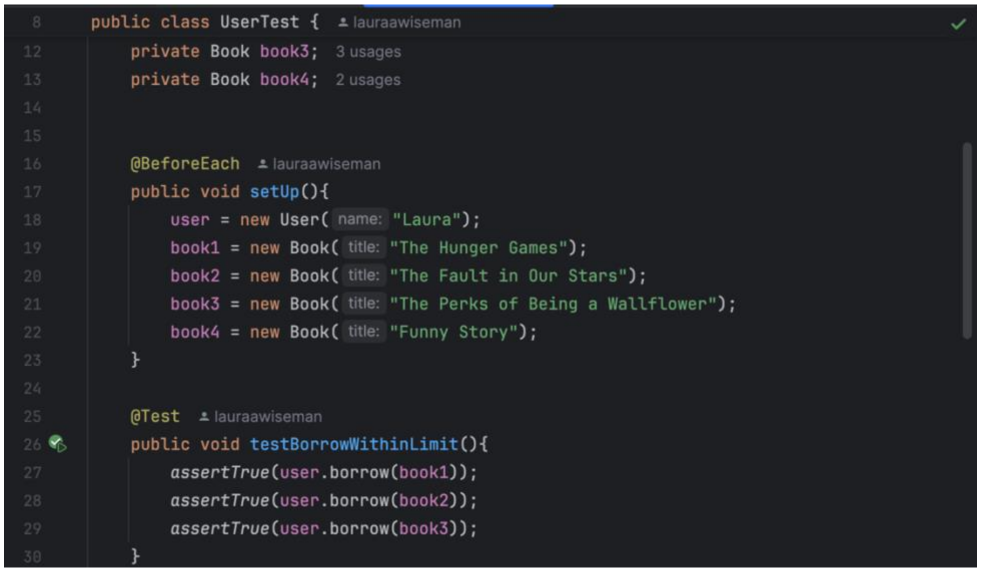
```java
    public User(String name){  1 usage  Laura Wiseman
        this.name = name;
    }

    public boolean borrow(Book book){  8 usages  Laura Wiseman
        if (borrowedBooks.size() < MAX_BORROW_LIMIT && book.isAvailable()){
            book.borrow();
            borrowedBooks.add(book);
            return true;
        }
        return false;
    }

    public void returnBook(Book book){  1 usage  Laura Wiseman
        if (borrowedBooks.remove(book)){
            book.returnBook();
        }
    }
```

User.java: In the borrow(Book book) method, the condition checks both the user's borrow limit and the book's availability in one clear line. If both are valid, it borrows the book and adds it to the user's list, then returns true. Otherwise, it returns false. This keeps the logic short and easy to follow.
In returnBook(Book book), the method checks if the book is in the borrowed list and if so, returns it. Using borrowedBooks.remove(book) directly in the condition avoids extra code and keeps things clean. Both methods use descriptive names and avoid unnecessary variables, making them east to read and maintain.

UserTest.java



```java
public class UserTest {  lauraawiseman
    private Book book3;  3 usages
    private Book book4;  2 usages


    @BeforeEach  lauraawiseman
    public void setUp(){
        user = new User( name: "Laura");
        book1 = new Book( title: "The Hunger Games");
        book2 = new Book( title: "The Fault in Our Stars");
        book3 = new Book( title: "The Perks of Being a Wallflower");
        book4 = new Book( title: "Funny Story");
    }

    @Test  lauraawiseman
    public void testBorrowWithinLimit(){
        assertTrue(user.borrow(book1));
        assertTrue(user.borrow(book2));
        assertTrue(user.borrow(book3));
    }
```

UserTest.java: These test checks that a user can borrow books up to the allowed limit. In the setup() method, a user and four book objects are created. Then in the test, the user successfully borrows three books, which is the maximum.

2. **<u>Project:</u>** Library Management System built in Java. It simulates basic library functionality, such as managing users, borrowing and returning books, and searching the catalog. There are 3 classes, Book, User and Library, and all the interaction is done through code(no user input), and the logic is kept simple and readable.

Book Class Test Cases:

1. **Test:** Book is available by default – Confirm when a new book is created, it is marked available.
2. **Test:** Borrowing a book changes availability – When user borrows a book, its availability should be set to false.
3. **Test:** Returning a book sets it back to available – After a book is returned, it should become available again.

User Class Test Cases:

1. **Test:** User can borrow within limit – A user should be able to borrow up to 3 books.
2. **Test:** Borrowing fails after limit is reached – After borrowing 3 books, further attempts should fail.
3. **Test:** Returning a book allows another borrow – After returning a book, the user should be allowed to borrow a new one.

Library Class Test Cases:

1. **Test:** Find book by title – Checks if the library can locate a book using its title.
2. **Test:** Check availability by title – Validate whether a specific book title is currently available.

3. **Dependencies:** Junit 5, copied from junit5.org and made changes for my current version (23).

4. I did take some extra time setting up and getting used to IntelliJ, but other than that not any big issues.