Library Management System

QAP 1 DEV OPS AND SDAT COMBINED

Mimya Hafiz-SD12

1. A.

```java
/**
 * Test successful return of a previously borrowed book.
 * Ensures the book becomes available again.
 */
@Test
public void testReturnBook_Success() {
    libraryService.borrowBook( userId: "U001", isbn: "ISBN001");

    boolean returned = libraryService.returnBook( userId: "U001", isbn: "ISBN001");

    assertTrue(returned, message: "Book should be successfully returned.");
    assertTrue(libraryService.isBookAvailable( isbn: "ISBN001"), message: "Book should be available again after return.");
}
```

Instead of using vague names like b or r, I use clear and descriptive names such as borrowed and returned. This improves readability.

1. B.

```java
/**
 * Represents a book in the library system.
 * Each book has an ISBN, a title, and an availability status.
 */

public class Book {  31 usages
    private final String isbn;  3 usages
    private final String title;  3 usages
    private boolean available;  4 usages
```

Each class is focused on a single responsibility. Book only handles book-related data. Similarly, User manages borrowing logic, and LibraryService handles operations across entities.

1. C.

```java
/**
 * Test attempting to borrow a book using a non-existent user ID.
 */
@Test
public void testBorrowBook_UserNotFound() {
    boolean result = libraryService.borrowBook( userId: "U999",  isbn: "ISBN001");

    assertFalse(result,  message: "Borrow attempt should fail for non-existent user.");
}
```

Using assert messages in JUnit helps clarify what the test is verifying. It simplifies debugging if something fails.

2.

Project Name: Library Management System (Java CLI-based)

This project allows library staff to manage books and users using a console application. Key functionalities include:

- Adding new books and users
- Borrowing and returning books
- Searching books by title

How works:

- Model: Contains Book and User classes to represent entities.
- Service: LibraryService handles the core logic for managing book availability and user borrowing behavior.
- Testing: JUnit 5 test classes (BookTest, UserTest, LibraryServiceTest) verify all business logic and edge cases.

Test Class	Key Scenarios Covered
BookTest	Constructor test, availability flag, toString()
UserTest	Borrowing limit, return logic, ID checks

Test Class	Key Scenarios Covered
LibraryServiceTest	Borrow/return flow, search, edge cases

3. These are the dependencies I used in my project:

```
<dependencies>
  <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter -->
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.9.3</version>
    <scope>test</scope>
  </dependency>

  <!-- https://mvnrepository.com/artifact/org.mockito/mockito-core -->
  <dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>5.18.0</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>RELEASE</version>
    <scope>test</scope>
  </dependency>
</dependencies>
```

I got it from https://mvnrepository.com/ .

4. I faced some issues with GitHub adding new rules for branches, and also for working with branches, with the passing of the JUnit test section before merging. It took me a long time to set up the repository perfectly for the project. Still, I have some issues to pass the test for PR in history, but the last ones are good to go.