

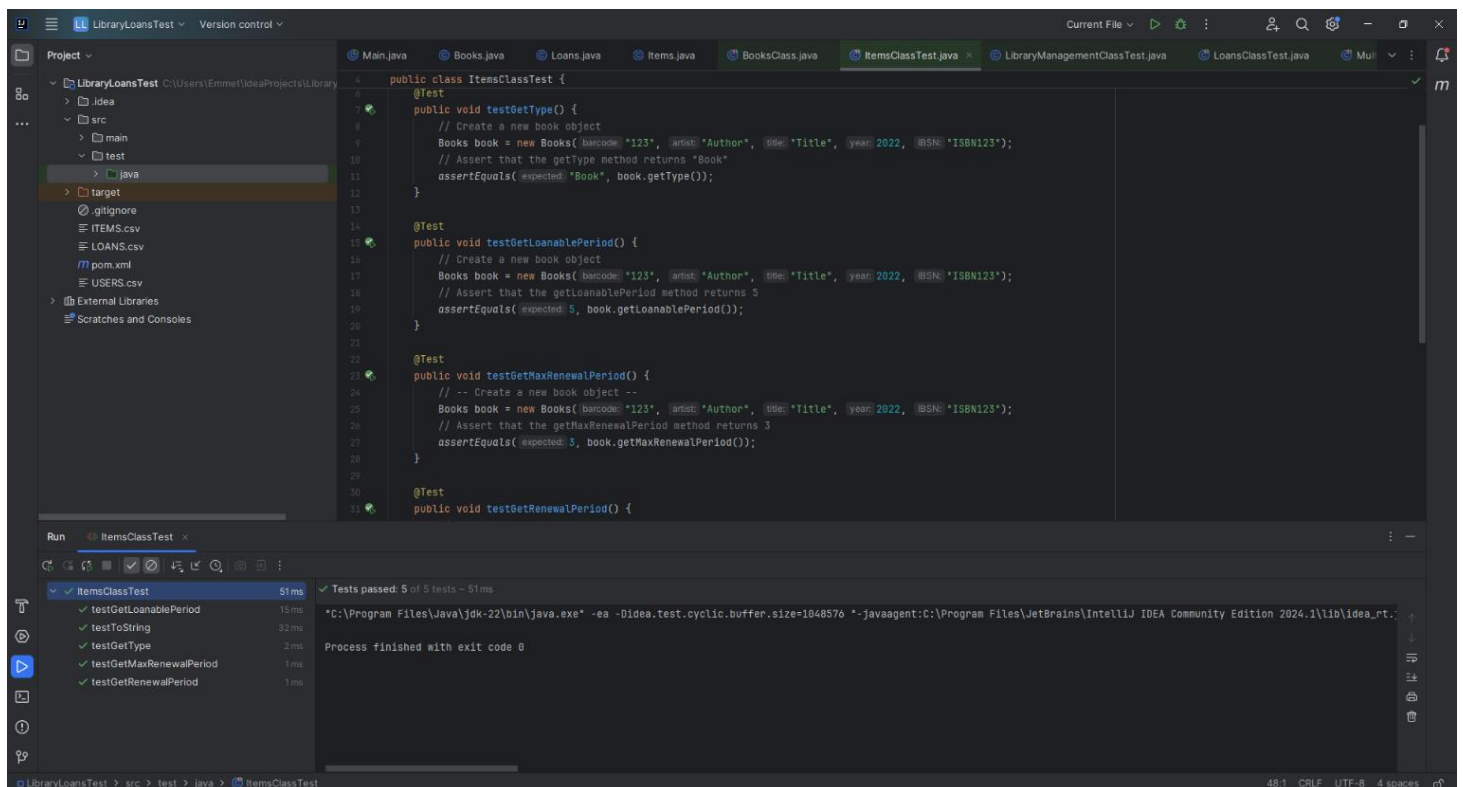
COM102 (Object Oriented Programming) - Practical Skills Assessment 2

Testing Document

This Testing Document provides an overview of the comprehensive testing that has been conducted on the library management system. For this system we wrote out Unit tests on each class within our system, The software that we used for our testing was Junit which is a very popular framework that many developers tend to use for their coding practices, as testing is a crucial element in the software development lifecycle, as it puts to the test the functionality, reliability, and performance of a system. For our system this was important to make sure that individual components of the system, such as the issue, renew, and return loan operations, was working how intended. This document will provide the testing approach and test cases to ensure the quality, integrity and ultimately showcase our library management system.

Item Class:

The Items Class serves as a comprehensive way of tests for the Books class within a library management system. It helps to examines the various aspects of the Books class's elements and functionality. This test that we have conducted focuses on a specific behaviour or attribute of the Books class, this would be the type, loanable period, renewal periods, and string representation, we wanted to evaluate methods regarding its compliance with expected goals. Some of these include getType, getLoanablePeriod, getMaxRenewalPeriod, getRenewalPeriod, and toString respectively to ensure that the Books class implementation is accurate and reliable making sure that no potential bugs are found.



The screenshot displays an IDE window with the following components:

- Project Explorer:** Shows the project structure with folders like 'src', 'test', and 'target'.
- Code Editor:** Contains the `ItemsClassTest` class with the following code:

```
public class ItemsClassTest {  
    @Test  
    public void testGetType() {  
        // Create a new book object  
        Books book = new Books(barcode: "123", author: "Author", title: "Title", year: 2022, isbn: "ISBN123");  
        // Assert that the getType method returns "Book"  
        assertEquals("expected: 'Book', book.getType());  
    }  
  
    @Test  
    public void testGetLoanablePeriod() {  
        // Create a new book object  
        Books book = new Books(barcode: "123", author: "Author", title: "Title", year: 2022, isbn: "ISBN123");  
        // Assert that the getLoanablePeriod method returns 5  
        assertEquals("expected: 5, book.getLoanablePeriod());  
    }  
  
    @Test  
    public void testGetMaxRenewalPeriod() {  
        // Create a new book object --  
        Books book = new Books(barcode: "123", author: "Author", title: "Title", year: 2022, isbn: "ISBN123");  
        // Assert that the getMaxRenewalPeriod method returns 3  
        assertEquals("expected: 3, book.getMaxRenewalPeriod());  
    }  
  
    @Test  
    public void testGetRenewalPeriod() {  
    }  
}
```
- Run Console:** Shows the execution results:
 - Tests passed: 5 of 5 tests - 51ms
 - Process finished with exit code 0
- Test Results:** A table showing the execution time for each test:

Test Method	Execution Time
testGetLoanablePeriod	15ms
testToString	32ms
testGetType	2ms
testGetMaxRenewalPeriod	1ms
testGetRenewalPeriod	1ms

Books Class:

This Books class represents a vital component of our library management system, encapsulating the properties and behaviour of books within the library collection. This class extends the Items superclass, inheriting its attributes such as barcode, artist, title, year, and ISBN. This class implements methods such as `getType`, `getLoanablePeriod`, `getMaxRenewalPeriod`, and `getRenewalPeriod`, all of which help to contribute to the overall functionality of our library system as it has characteristics and borrowing policies that are very important within our system as it is specific to books, enabling us to have an efficient management of loans, renewals, and returns, otherwise we wouldn't. Conducting testing we have found that this class seems to be consistent and has predictable behaviour when regarding the book related operations throughout our system.

The screenshot displays the IntelliJ IDEA IDE interface. The top pane shows the `Books.java` file, which extends the `Items` class. The code includes an import for `java.util.Scanner` and defines the `Books` class with a constructor and four overridden methods: `getType`, `getLoanablePeriod`, `getMaxRenewalPeriod`, and `getRenewalPeriod`.

```
3 import java.util.Scanner;
4
5
6 public class Books extends Items {
7
8     public Books(String barcode, String artist, String title, int year, String ISBN) {
9         super(barcode, artist, title, year, ISBN);
10    }
11
12    @Override
13    public String getType () {
14        return "Book";
15    }
16
17    @Override
18    public int getLoanablePeriod() {
19        return 5;
20    }
21
22    @Override
23    public int getMaxRenewalPeriod() {
24        return 3;
25    }
26
27    @Override
28    public int getRenewalPeriod() {
29        return 2;
30    }
31 }
```

The bottom pane shows the `Run` tab for `ItemsClassTest`. The test results indicate that all 5 tests passed successfully within 51 ms.

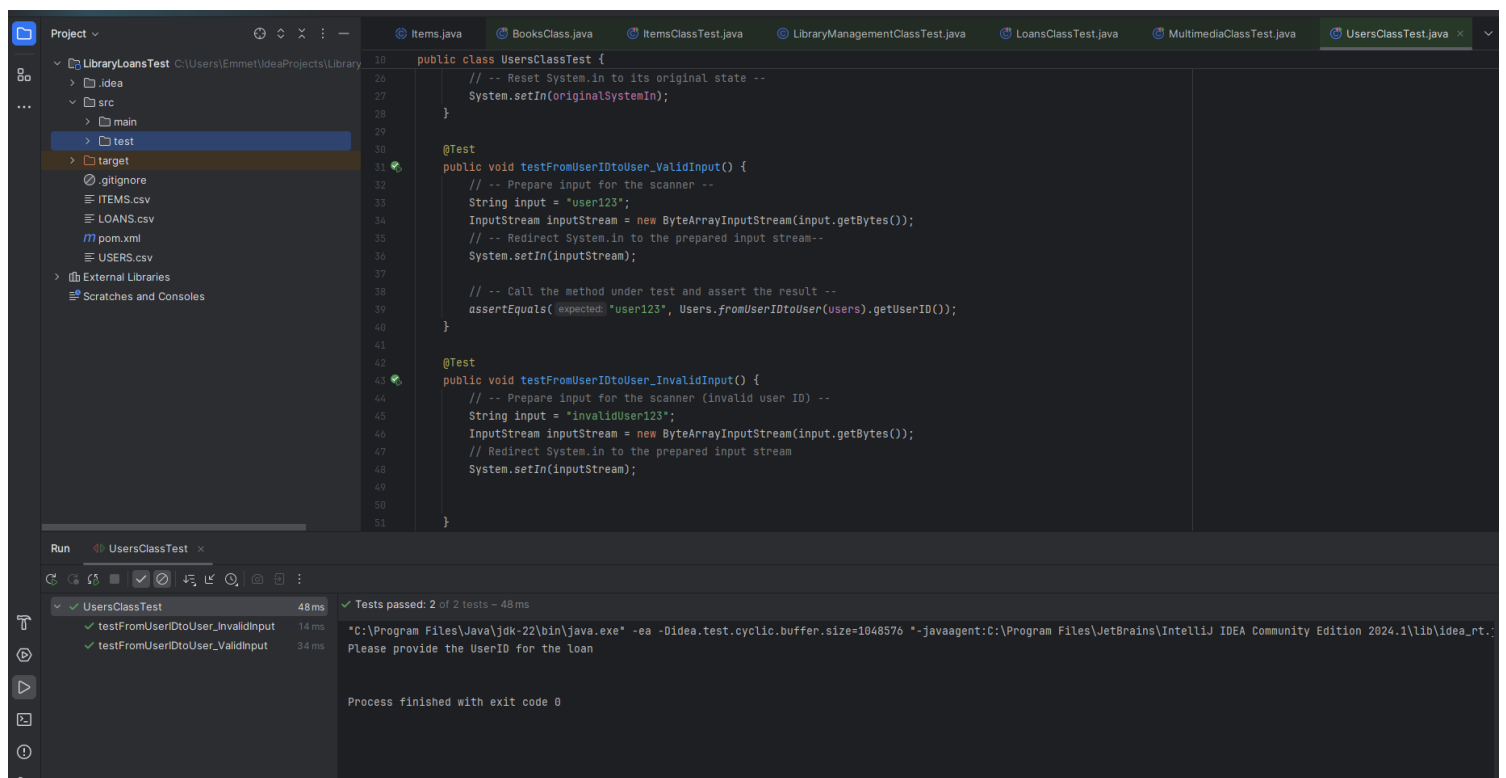
Test Method	Duration
<code>testGetLoanablePeriod</code>	15 ms
<code>testToString</code>	32 ms
<code>testGetType</code>	2 ms
<code>testGetMaxRenewalPeriod</code>	1 ms
<code>testGetRenewalPeriod</code>	1 ms

The console output shows the command used to run the tests and confirms that the process finished with exit code 0.

```
"C:\Program Files\Java\jdk-22\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt..."
Process finished with exit code 0
```

User Class:

For this User class the goal that we had in mind was to thoroughly verify and examine the behaviour of the `fromUserIDtoUser()` method in the `Users` class, to ensure that it correctly identifies and returns the user object based on the provided user ID. Essentially, we wanted to make sure that it was reading the valid and invalid inputs correctly. The test suite is divided into two main tests. First of all was the `testFromUserIDtoUser_ValidInput()` where we focused on setting up a list of users, which creates an input stream with the valid user ID, and then calls the `fromUserIDtoUser()` method, then we would check if the object that was created has the correct user input. Then the `testFromUserIDtoUser_InvalidInput()` ultimately checks what happens when an incorrect user ID is provided. Overall, this testing approach helps ensure that the `fromUserIDtoUser()` method works as expected, both handling valid and invalid inputs how it is intended to.



The screenshot displays the IntelliJ IDEA IDE with the `UsersClassTest.java` file open. The code defines two test methods: `testFromUserIDtoUser_ValidInput()` and `testFromUserIDtoUser_InvalidInput()`. Both tests use `System.setIn()` to redirect input from a scanner to a prepared `InputStream`. The first test uses the valid user ID "user123", and the second test uses the invalid user ID "invalidUser123". Both tests call the `Users.fromUserIDtoUser()` method and assert the result using `assertEquals()`.

```
18 public class UsersClassTest {
19     // -- Reset System.in to its original state --
20     System.setIn(originalSystemIn);
21 }
22
23 @Test
24 public void testFromUserIDtoUser_ValidInput() {
25     // -- Prepare input for the scanner --
26     String input = "user123";
27     InputStream inputStream = new ByteArrayInputStream(input.getBytes());
28     // -- Redirect System.in to the prepared input stream --
29     System.setIn(inputStream);
30
31     // -- Call the method under test and assert the result --
32     assertEquals("expected: 'user123'", Users.fromUserIDtoUser(users).getUserID());
33 }
34
35 @Test
36 public void testFromUserIDtoUser_InvalidInput() {
37     // -- Prepare input for the scanner (invalid user ID) --
38     String input = "invalidUser123";
39     InputStream inputStream = new ByteArrayInputStream(input.getBytes());
40     // Redirect System.in to the prepared input stream
41     System.setIn(inputStream);
42 }
43 }
```

The Run tab at the bottom shows the test results for `UsersClassTest`. Both tests passed successfully.

Test Name	Duration	Status
testFromUserIDtoUser_InvalidInput	14 ms	Passed
testFromUserIDtoUser_ValidInput	34 ms	Passed

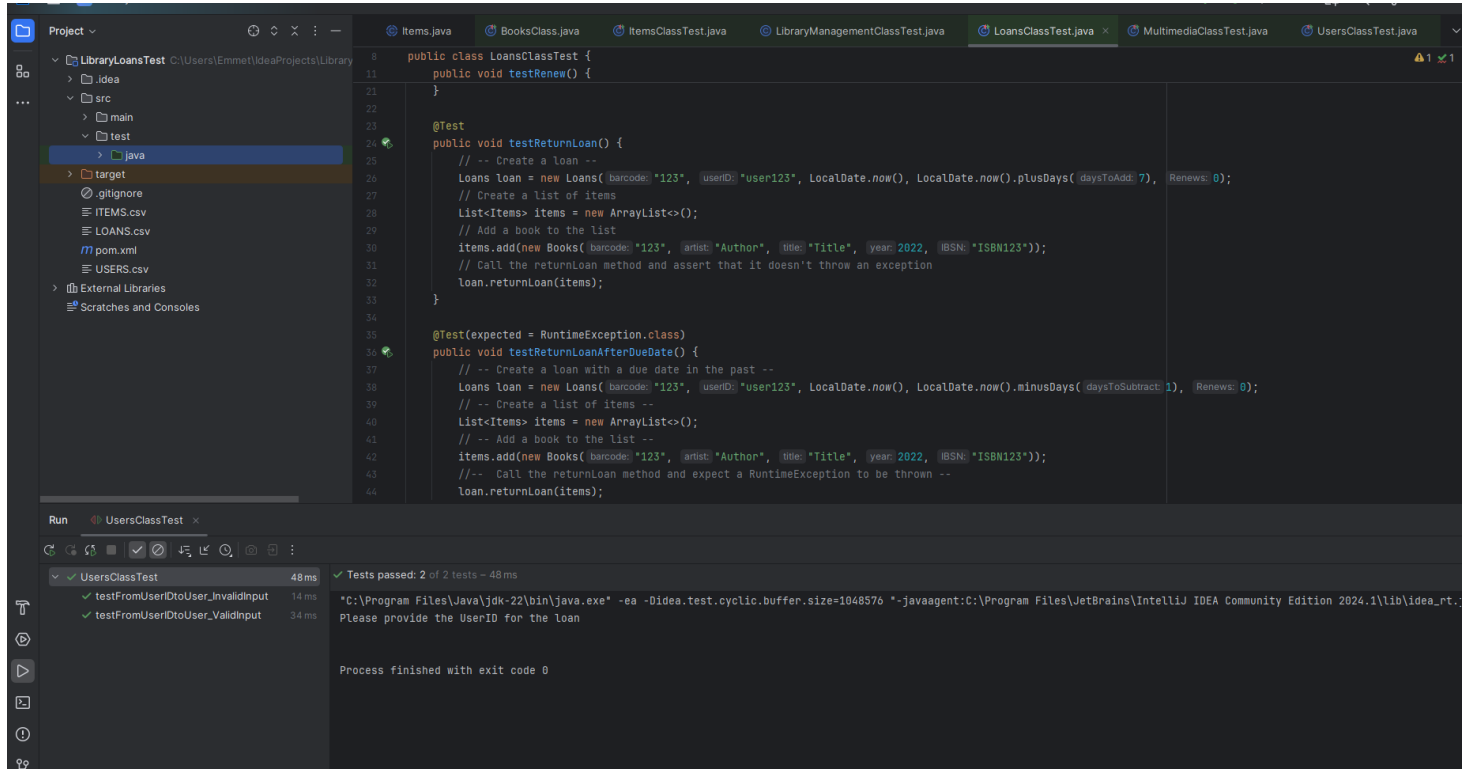
Tests passed: 2 of 2 tests - 48 ms

Process finished with exit code 0

Loan Class:

For this Loan class this included three main test cases that cover different aspects of the `Loans` class's functionality. The first one is `testRenew()` which checks the behaviour of the `renew()` method. It creates a loan object with a due date 7 days from the current date, adds a book to the loan, and then calls the `renew()` method. The test then asserts that the due date of the loan has been extended by 2 days, from 7 days to 9 days. Now onto the second this is `testReturnLoan()` which checks the behaviour of the `returnLoan()` method. It creates a loan object with a due date 7 days from the current date, adds a book to the loan, and then calls the `returnLoan()` method. This tells us that the method used does not throw any exceptions. And lastly this is `testReturnLoanAfterDueDate()`, which checks the behaviour of the `returnLoan()` method when

the loan is returned after the due date. It creates a loan object with a due date in the past, adds a book to the loan, and then calls the returnLoan() method after this method has been called, we suspect the RuntimeException to be thrown, as the loan is being returned after the due date. Overall, the loan class is working how expected as it can renew, return before and after successfully.



The screenshot displays the IntelliJ IDEA IDE with the `LoansClassTest.java` file open. The code defines two test methods: `testRenew()` and `testReturnLoan()`. The `testReturnLoan()` method is annotated with `@Test` and contains logic to create a loan, add items, and call `returnLoan()`. A second test method, `testReturnLoanAfterDueDate()`, is also present, annotated with `@Test(expected = RuntimeException.class)`, and it creates a loan with a past due date to verify an exception is thrown.

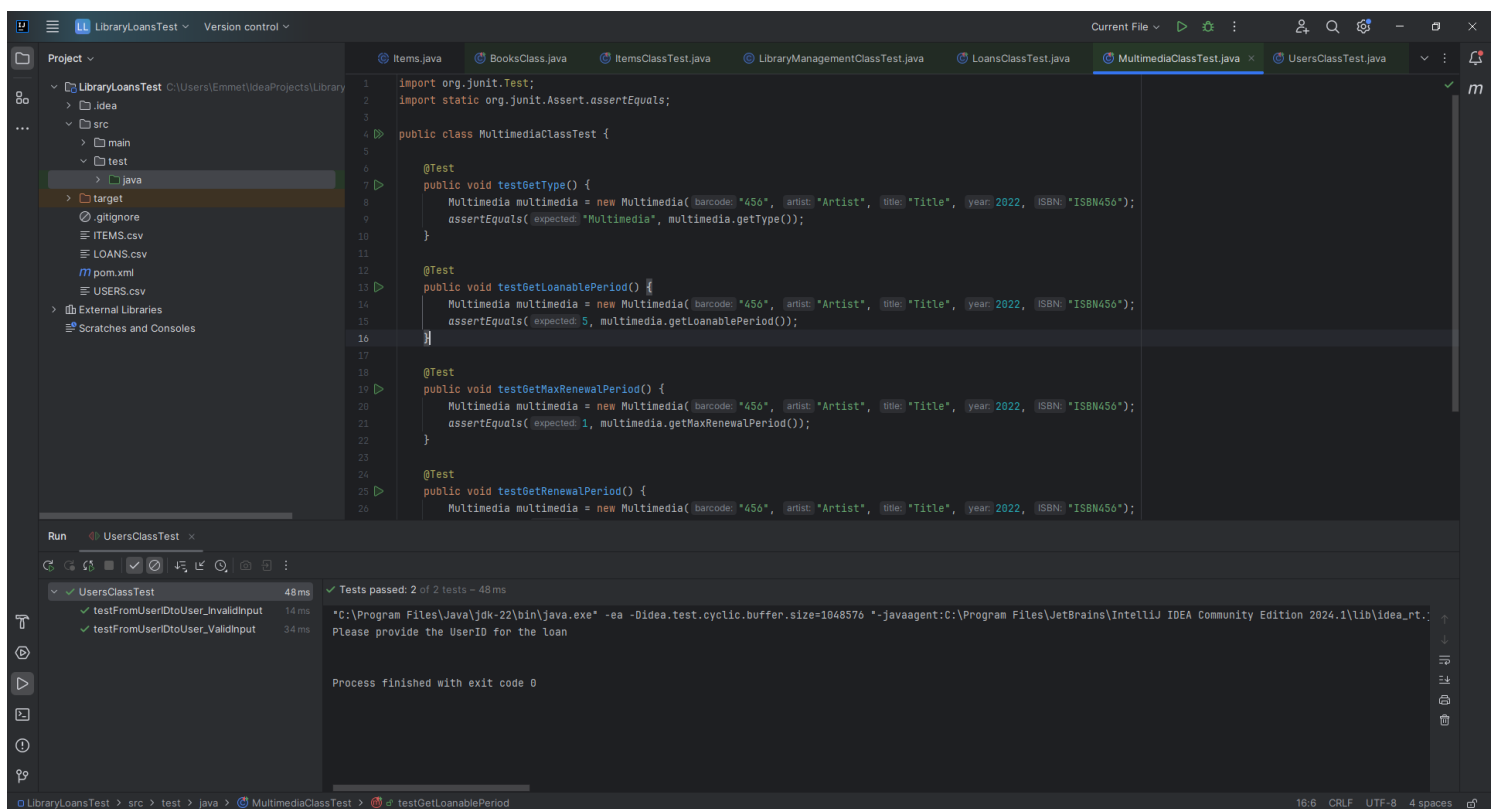
```
8 public class LoansClassTest {
11     public void testRenew() {
21     }
22
23     @Test
24     public void testReturnLoan() {
25         // -- Create a loan --
26         Loans loan = new Loans( barcode: "123", userID: "user123", LocalDate.now(), LocalDate.now().plusDays( daysToAdd: 7),
27                                // Create a list of items
28                                List<Items> items = new ArrayList<>());
29         // Add a book to the list
30         items.add(new Books( barcode: "123", artist: "Author", title: "Title", year: 2022, ISBN: "ISBN123"));
31         // Call the returnLoan method and assert that it doesn't throw an exception
32         loan.returnLoan(items);
33     }
34
35     @Test(expected = RuntimeException.class)
36     public void testReturnLoanAfterDueDate() {
37         // -- Create a loan with a due date in the past --
38         Loans loan = new Loans( barcode: "123", userID: "user123", LocalDate.now(), LocalDate.now().minusDays( daysToSubtract: 1),
39                                // Create a list of items --
40                                List<Items> items = new ArrayList<>());
41         // -- Add a book to the list --
42         items.add(new Books( barcode: "123", artist: "Author", title: "Title", year: 2022, ISBN: "ISBN123"));
43         //-- Call the returnLoan method and expect a RuntimeException to be thrown --
44         loan.returnLoan(items);
45     }
46 }
```

The Run window at the bottom shows the test results for `UsersClassTest`. Two tests passed: `testFromUserDtoUser_InvalidInput` (14 ms) and `testFromUserDtoUser_ValidInput` (34 ms). The total test time was 48 ms, and 2 out of 2 tests passed.

```
Run UsersClassTest
testFromUserDtoUser_InvalidInput 14 ms
testFromUserDtoUser_ValidInput 34 ms
Tests passed: 2 of 2 tests - 48 ms
"C:\Program Files\Java\jdk-22\bin\java.exe" -ea -Didea.test.cyclic.buffer.size=1048576 "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Community Edition 2024.1\lib\idea_rt.
Please provide the UserID for the loan
Process finished with exit code 0
```

Multimedia Class:

This Multimedia includes four stages that address the fundamental operations and workings of the Multimedia class, ensuring that its methods return the expected values. The testGetType() method verifies that the getType() method correctly identifies the type of the multimedia item as "Multimedia" and not "Book". The testGetLoanablePeriod() and testGetRenewalPeriod() methods check that if the getLoanablePeriod() and getRenewalPeriod() methods return the expected values of 5 days and 3 days, respectively. Then potentially the testGetMaxRenewalPeriod() method ensures that the getMaxRenewalPeriod() method returns the expected value of 1. Each test that we have conducted creates a Multimedia object and uses the JUnit's assertEquals() method to compare the expected values returned by the tested methods. This testing approach helps maintain the reliability and consistency of the Multimedia class.



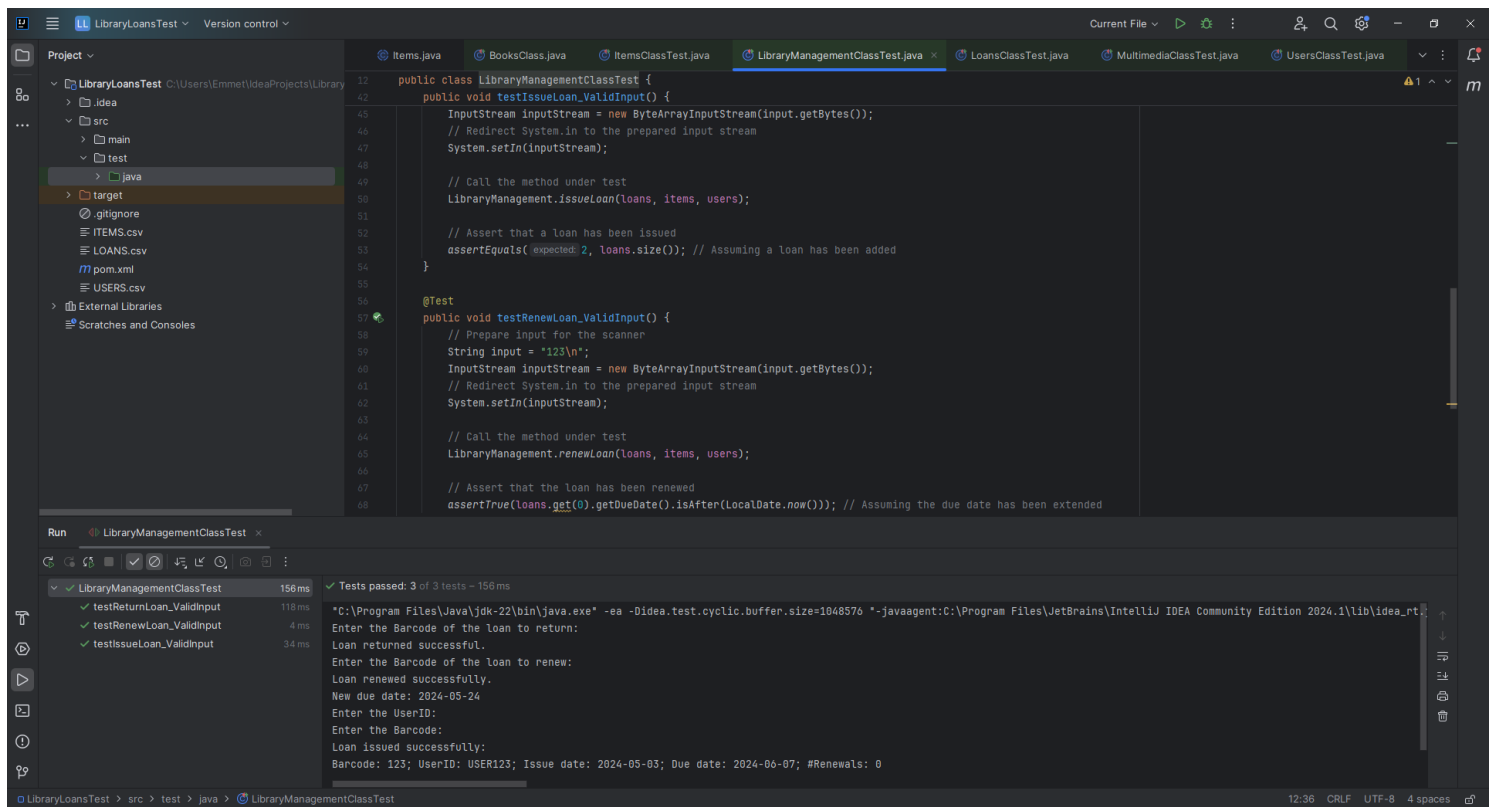
The screenshot displays an IDE window with the following components:

- Project View:** Shows the project structure for 'LibraryLoansTest' with folders for 'src', 'main', 'test', 'java', 'target', and files like '.gitignore', 'ITEMS.csv', 'LOANS.csv', 'pom.xml', and 'USERS.csv'.
- Code Editor:** Contains the 'MultimediaClassTest.java' file with the following code:

```
1 import org.junit.Test;
2 import static org.junit.Assert.assertEquals;
3
4 public class MultimediaClassTest {
5
6     @Test
7     public void testGetType() {
8         Multimedia multimedia = new Multimedia( barcode: "456", artist: "Artist", title: "Title", year: 2022, isbn: "ISBN456");
9         assertEquals( expected: "Multimedia", multimedia.getType());
10    }
11
12    @Test
13    public void testGetLoanablePeriod() {
14        Multimedia multimedia = new Multimedia( barcode: "456", artist: "Artist", title: "Title", year: 2022, isbn: "ISBN456");
15        assertEquals( expected: 5, multimedia.getLoanablePeriod());
16    }
17
18    @Test
19    public void testGetMaxRenewalPeriod() {
20        Multimedia multimedia = new Multimedia( barcode: "456", artist: "Artist", title: "Title", year: 2022, isbn: "ISBN456");
21        assertEquals( expected: 1, multimedia.getMaxRenewalPeriod());
22    }
23
24    @Test
25    public void testGetRenewalPeriod() {
26        Multimedia multimedia = new Multimedia( barcode: "456", artist: "Artist", title: "Title", year: 2022, isbn: "ISBN456");
```
- Run View:** Shows the execution of 'UsersClassTest' with a success message: 'Tests passed: 2 of 2 tests - 48 ms'. It also displays the command used to run the tests and the output: 'Please provide the UserID for the loan'.
- Bottom Bar:** Shows the current file path: 'LibraryLoansTest > src > test > java > MultimediaClassTest > testGetLoanablePeriod'.

Library Management System:

This class is most likely our most important within our system, as this class is responsible for making sure that all of the operations with this system is working. We needed to make sure that operations such as issuing loans, renewing loans, and returning loans work correctly with various input scenarios. For this test we had to check for three methods these are: `testIssueLoan_ValidInput()`, `testRenewLoan_ValidInput()` and `testReturnLoan_ValidInput()`. Let's discuss about the issue loan first this checks to see what happens when a valid user ID and item ID are given then it will be directed towards the Redirecting System.in to the `ByteArrayInputStream` so that the `issueLoan()` method can read the input that has been given then it Calls the `issueLoan()` method, passing in the loans, items, and users lists, and then adding a loan to the list of loans etc. The second one we will talk about is `testRenewLoan_ValidInput()` this follows a similar process to the first but instead asserts that the due date of the first loan has been extended and lastly is the `testReturnLoan_ValidInput()` which is the exact same format as the first two, but this calls that the loan has been removed from the loans list. Overall, we feel confident that the loan operations work as expected by covering the various input scenarios within this system.



The screenshot displays an IDE window with the following components:

- Project View:** Shows the project structure with folders like `src`, `main`, `test`, and `java`.
- Code Editor:** Contains the `LibraryManagementClassTest.java` file. The code includes three test methods:
 - `testIssueLoan_ValidInput()`: Sets up a `ByteArrayInputStream` and calls `LibraryManagement.issueLoan(loans, items, users)`. It asserts that the loan size is 2.
 - `testRenewLoan_ValidInput()`: Sets up a `ByteArrayInputStream` and calls `LibraryManagement.renewLoan(loans, items, users)`. It asserts that the due date of the first loan is after the current date.
 - `testReturnLoan_ValidInput()`: (Partially visible) Sets up a `ByteArrayInputStream` and calls `LibraryManagement.returnLoan(loans, items, users)`.
- Run View:** Shows the execution results of the tests. The output indicates that all three tests passed successfully, with a total of 3 tests passed in 156ms. The output also shows the input data used for the tests, including a barcode of 123, a user ID of USER123, and an issue date of 2024-05-03.