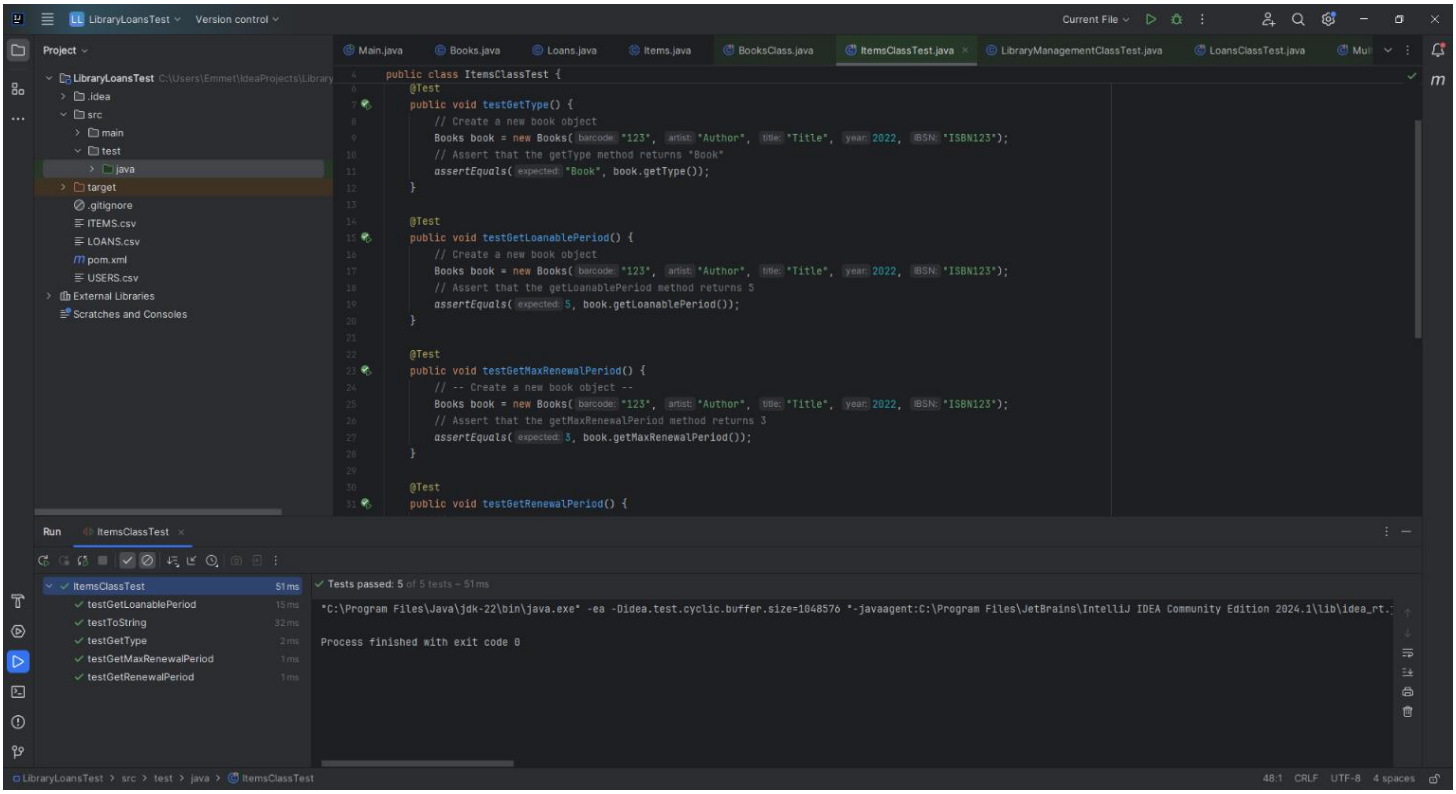
COM102 (Object Oriented Programming) - Practical Skills Assessment 2

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.

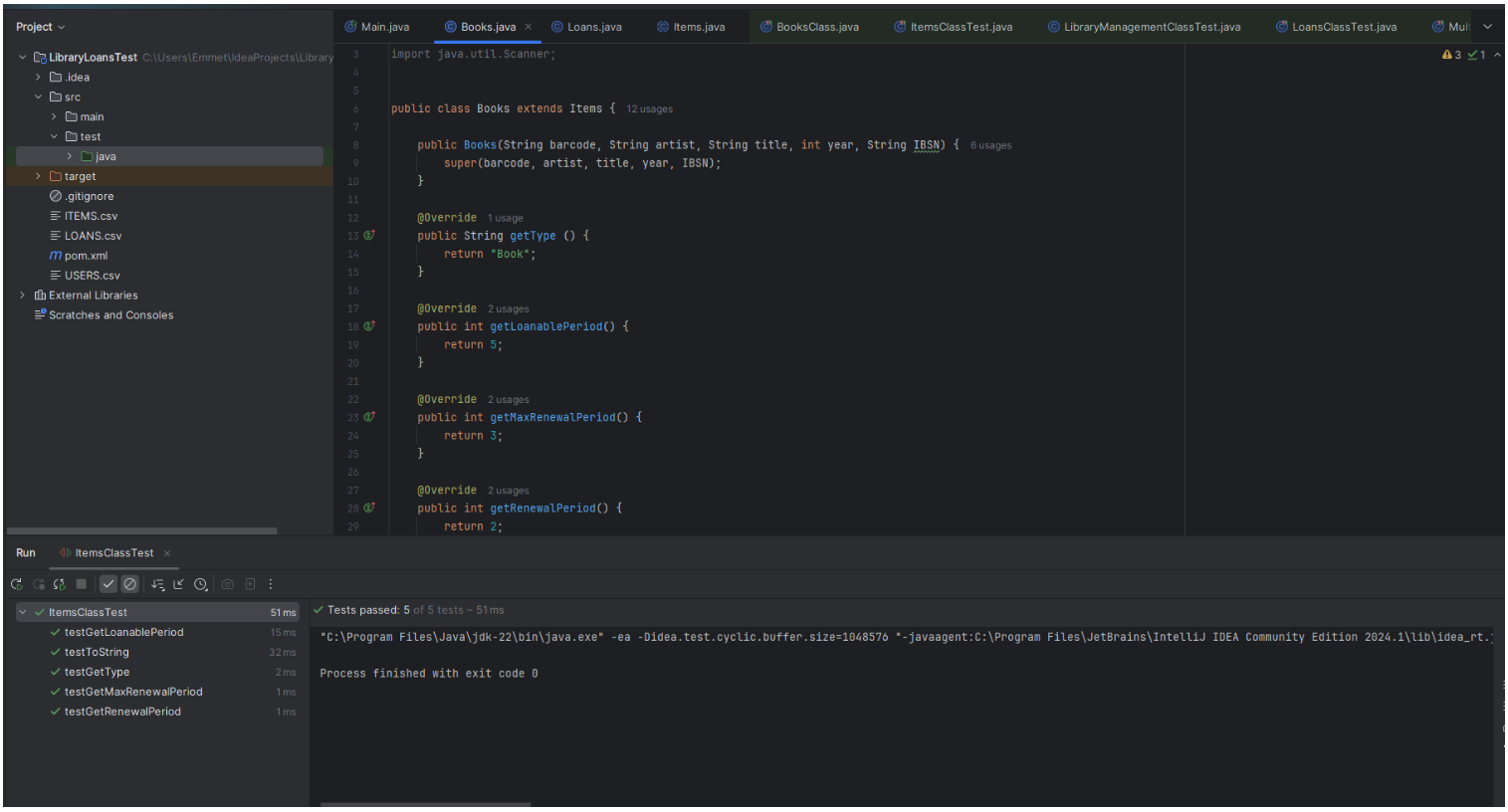
Item Class:

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.



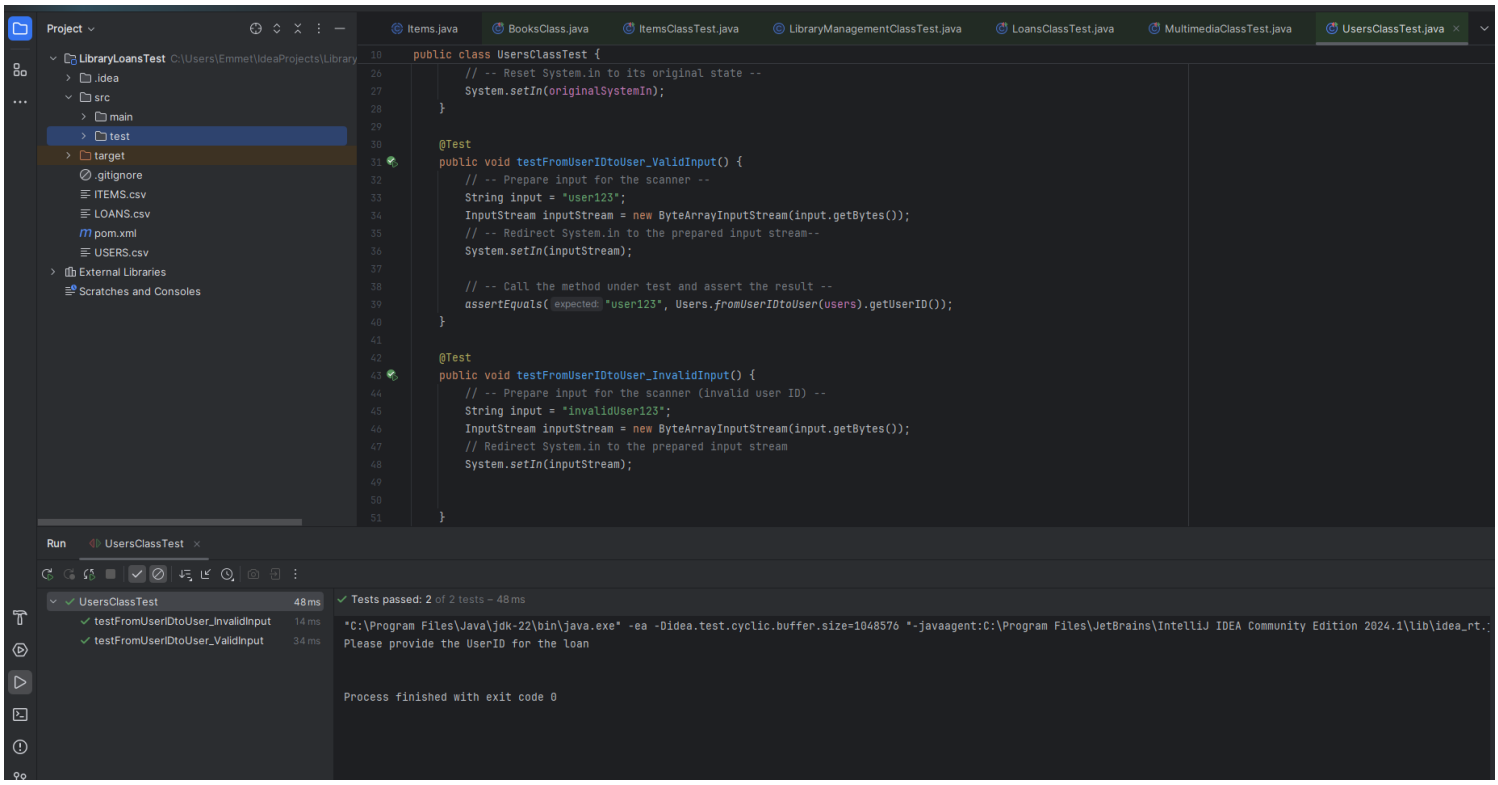
Books Class:

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 this 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.



User Class:

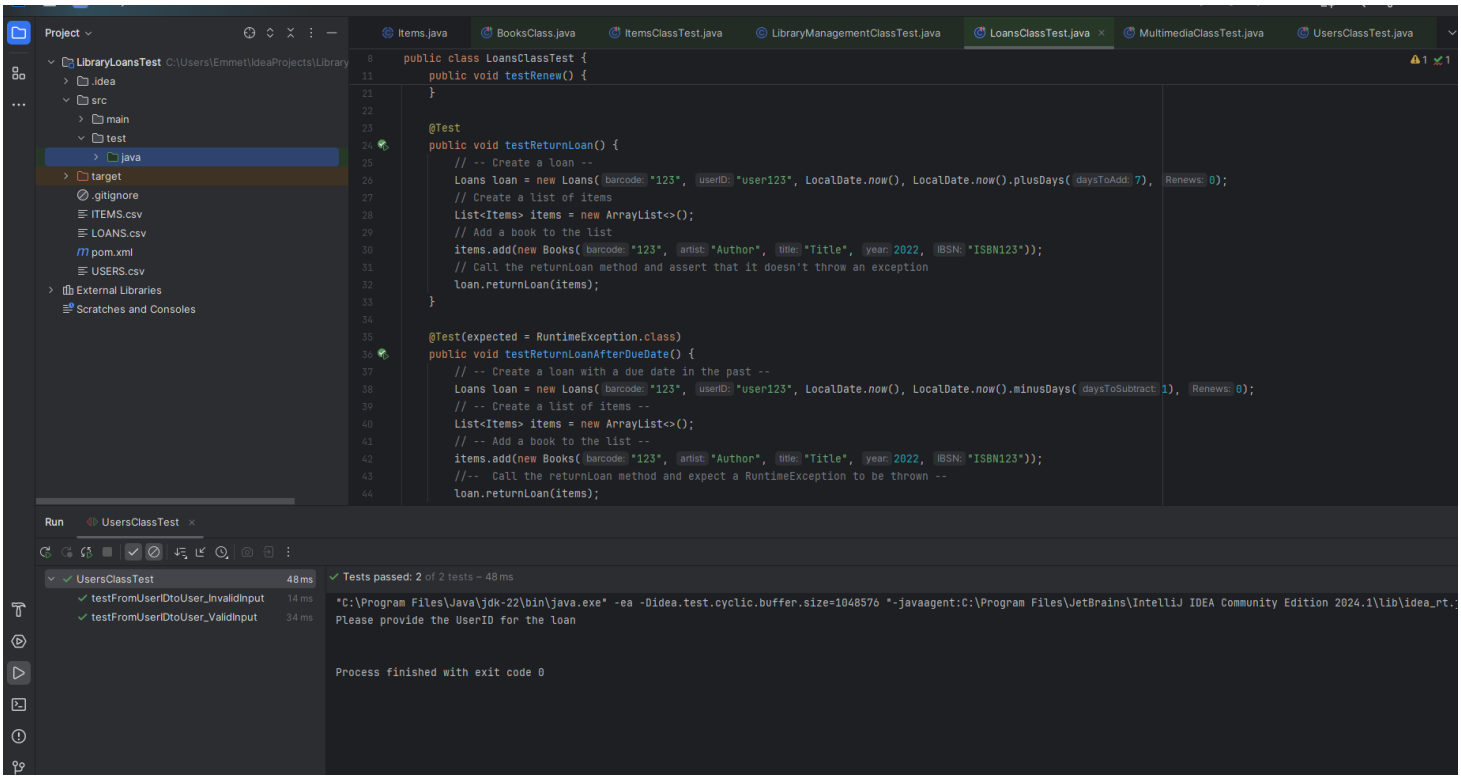
For this User class the goal that we had in mind was to thoroughly verify and examine the behaviour of the fromUserIDtoUser() method in the Users class, to ensure that it correctly identifies and returns the user object based on the provided user ID. Essentially, we wanted to make sure that it was reading 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 handing valid and invalid inputs how it is intended to.



Loan Class:

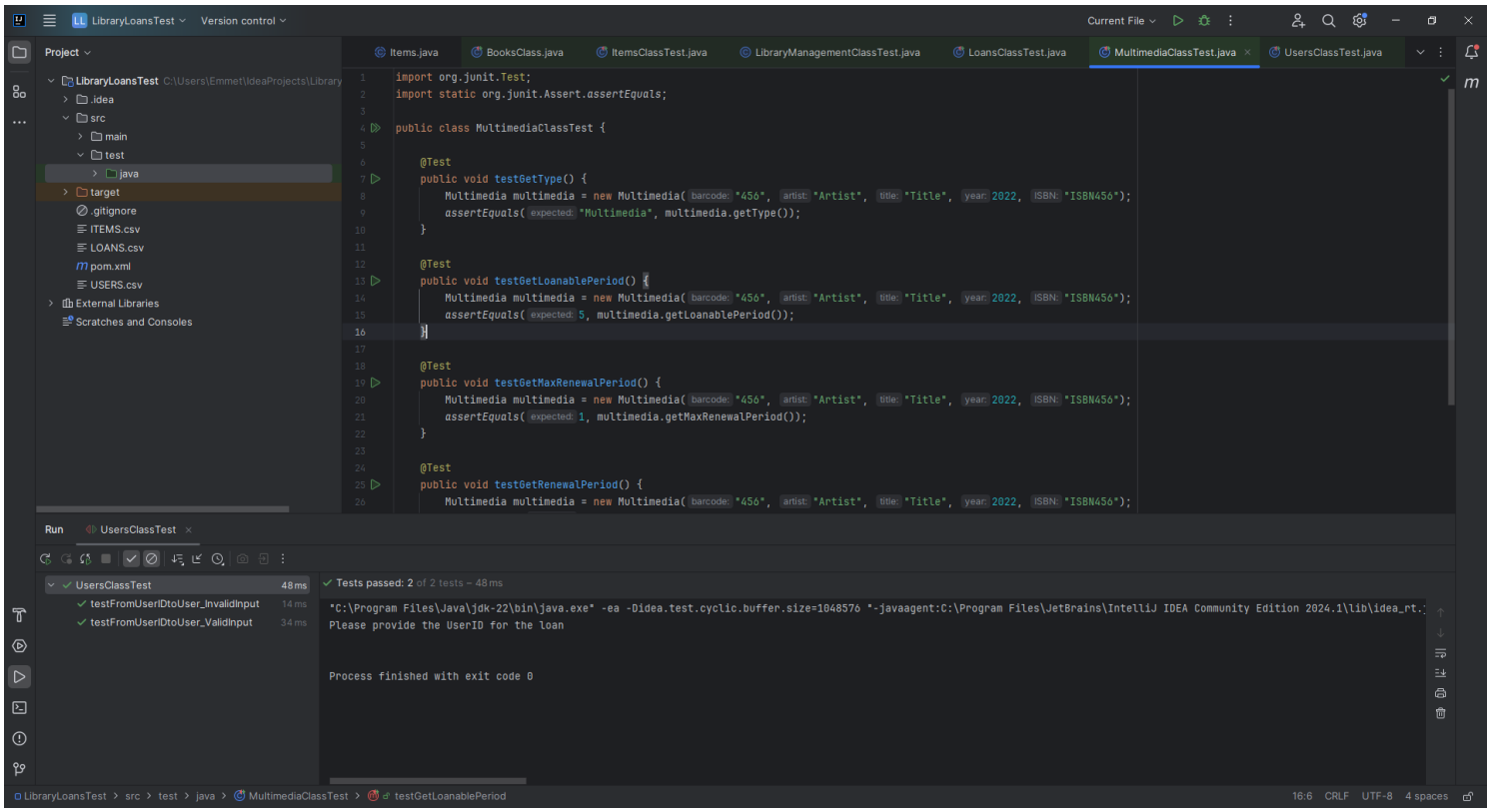
For this Loan class this included three main test cases that cover different aspects of the Loans class's functionality. The first one is testRenew() which checks the behaviour of the renew() method. It creates a loan object with a due date 7 days from the current date, adds a book to the loan, and then 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.



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.



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.