Mark Kardash

Professor Carl Eyler

SDEV 460 7380: Software Security Testing

Homework 1: Unit Testing

25 March 2024

University of Maryland Global Campus

In this document, I will be presenting the test cases, and their results, for SDEV460 Assignment 1. Unfortunately, my code was full of errors, and I was not able to run a single unit test. However, I will still provide screenshots of the code, and explain how it was intended to work in the first place.

Before we can get to the unit tests, we must first ensure that the login program we created runs as intended. Most of the main code was created using Eclipse Windowbuilder. Below are screenshots of it.

Building the Application:



Figure 1: Application Code: Imports and Main Method

In the above screenshot, we import all the necessary functions for the program, and start the main method.

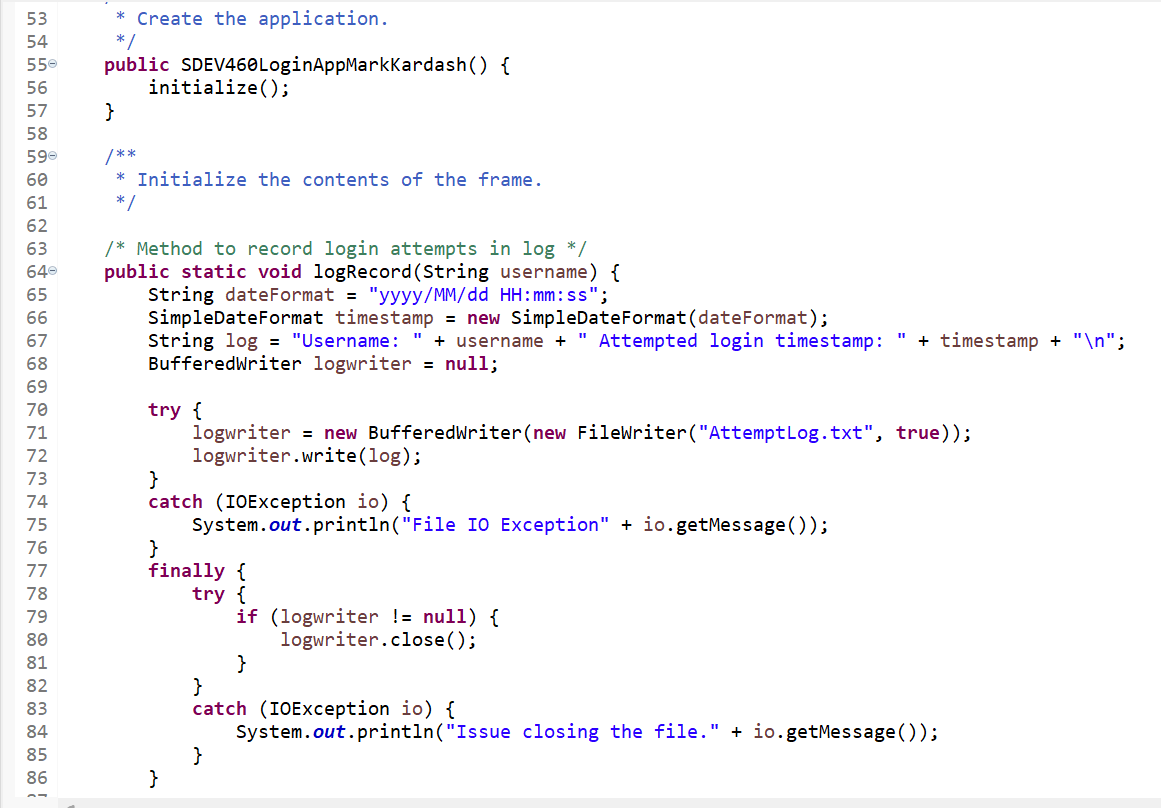


Figure 2: Application Code: Initializing app and writing logRecord method.

In the figure above, a method initializing the application can be seen, along with one for recording login credentials into a “.txt” file.

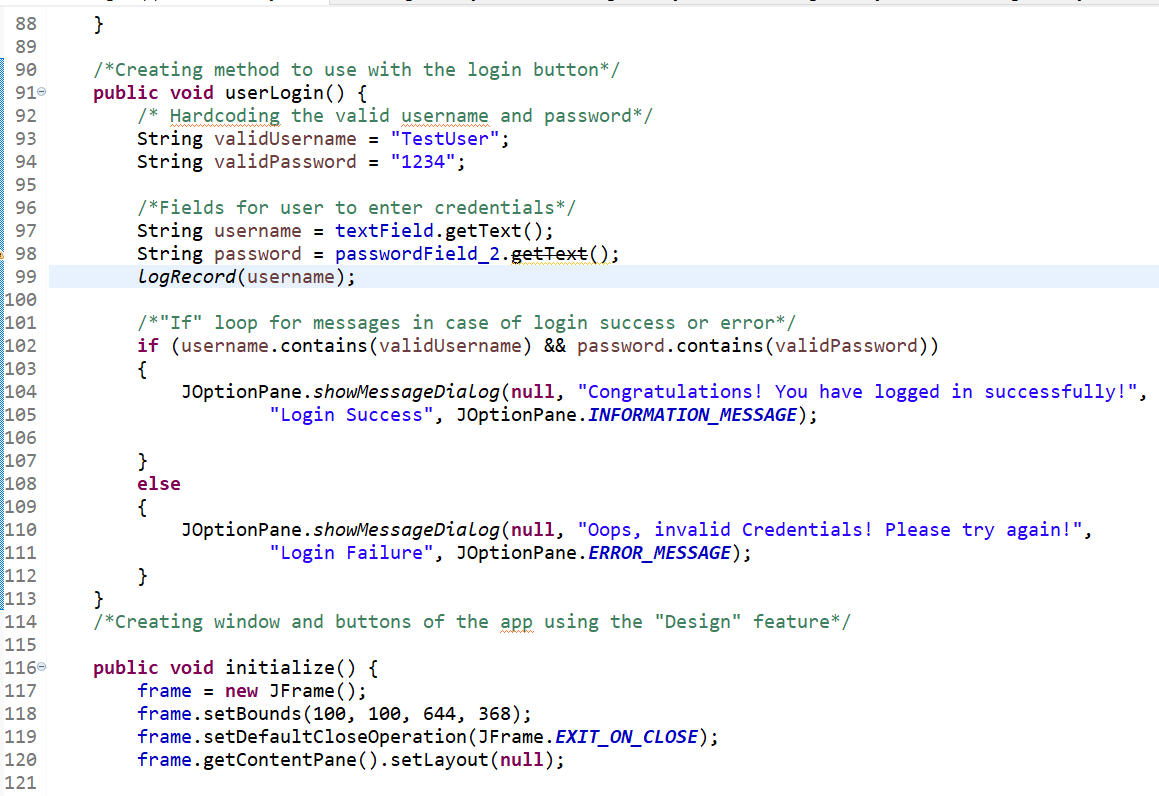


Figure 3: Application Code: User Login Method and initializing variables

Here, we create the method for the user login, and initialize all the variables we previously created.

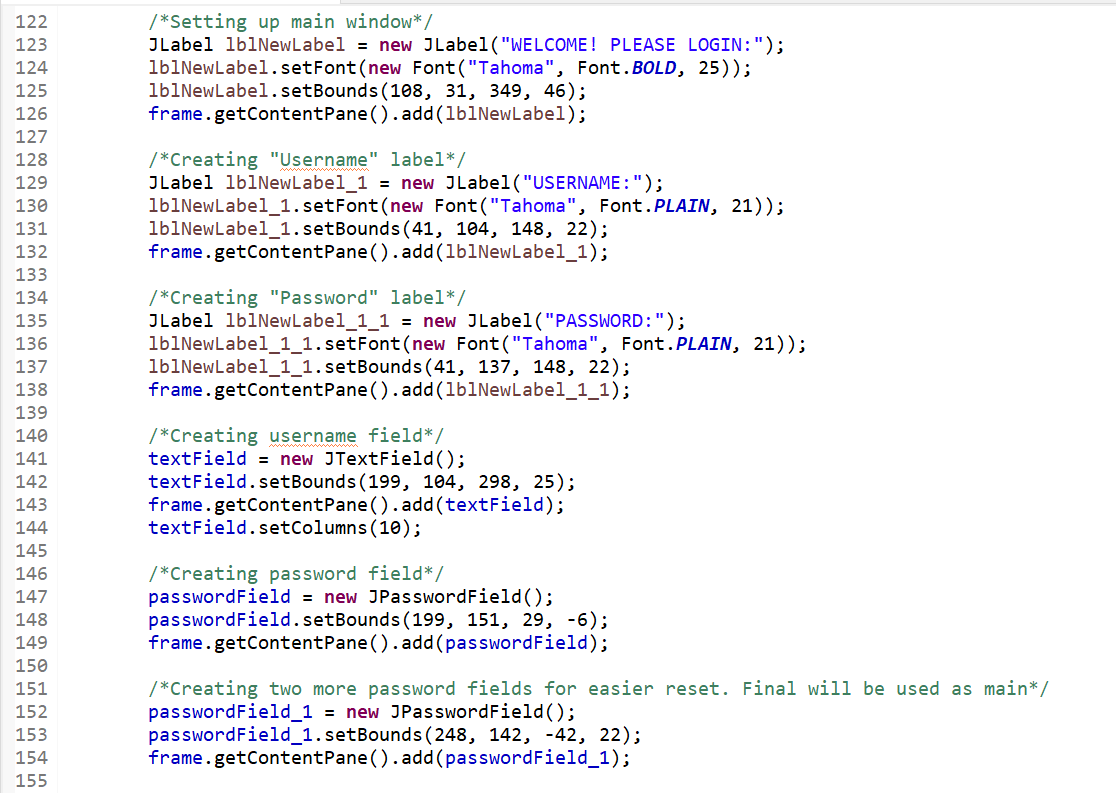


Figure 4: Creating user interface.

We now move to creating the actual user interface, by calling all the labels and buttons needed.

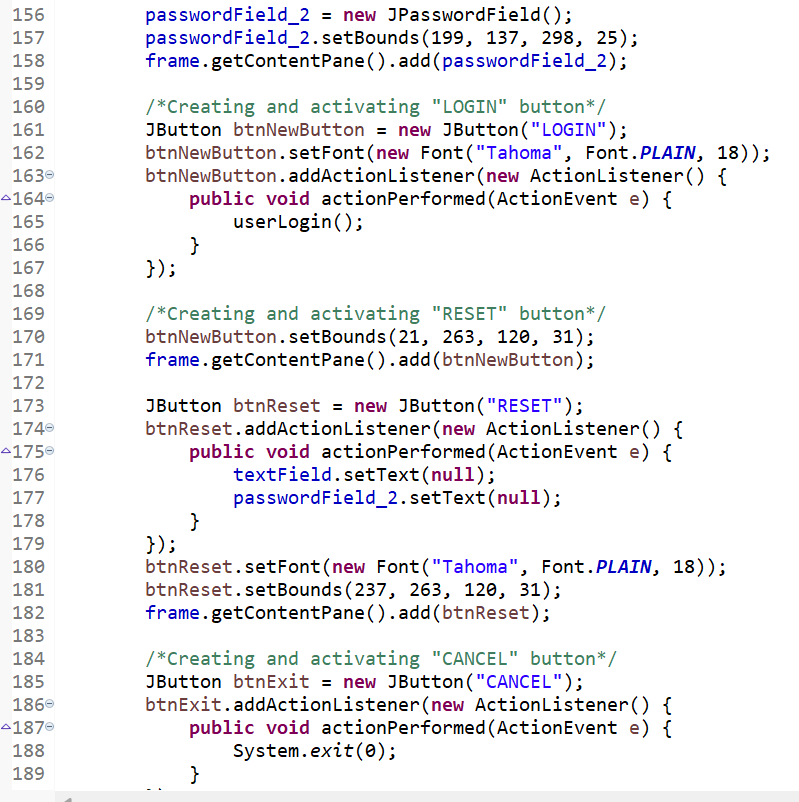


Figure 5: Creating User Interface Part 2

Yet more UI elements being created.

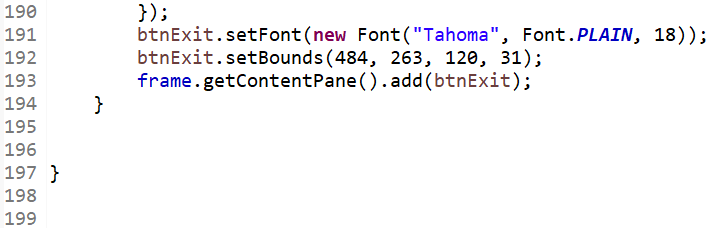


Figure 6: Creating User Interface Part 3

Finishing up with UI creation.

Running the Application:

To ensure proper functioning of the program, we run it:

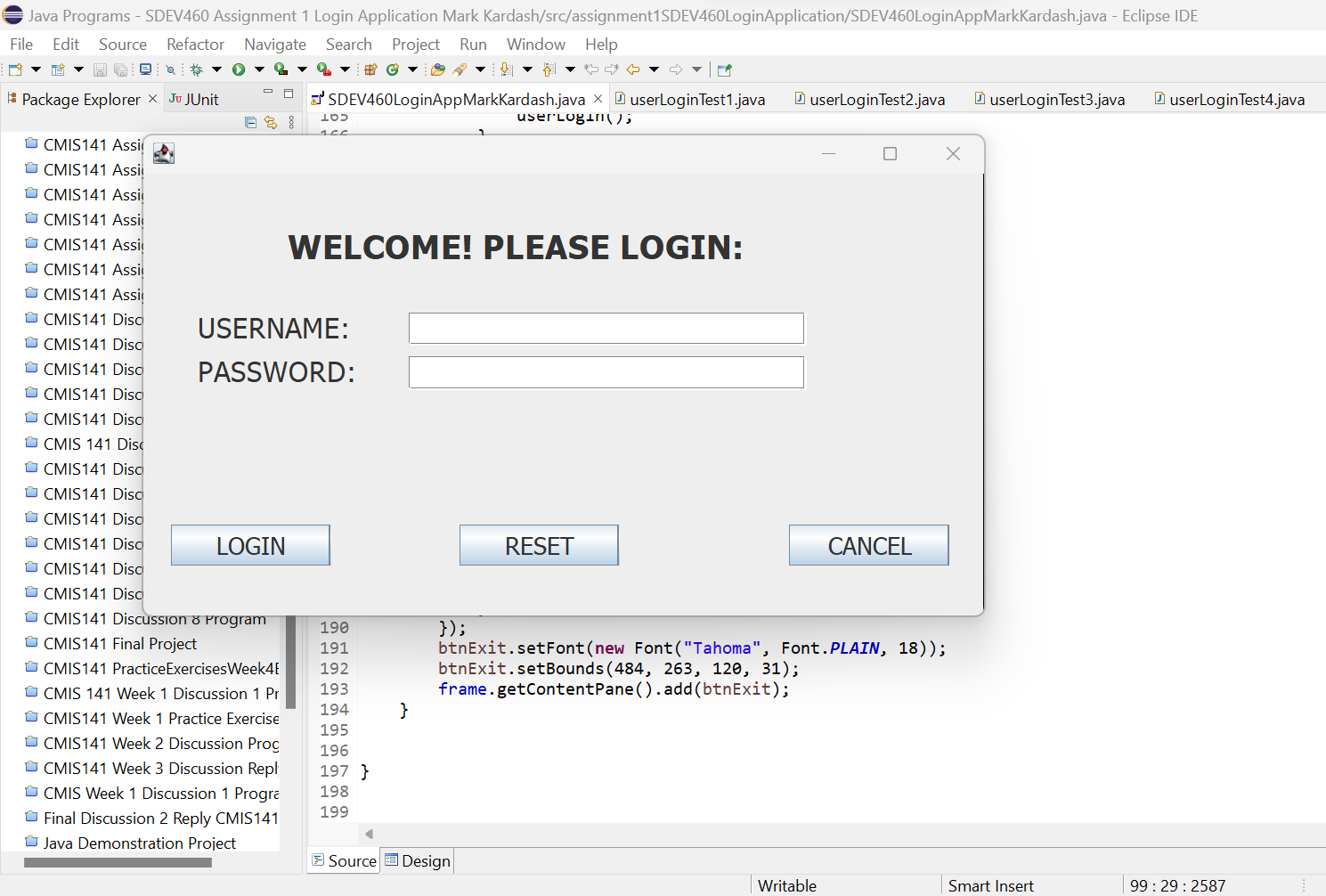


Figure 7: UI appears

As we can see, the user interface we created has appeared successfully.

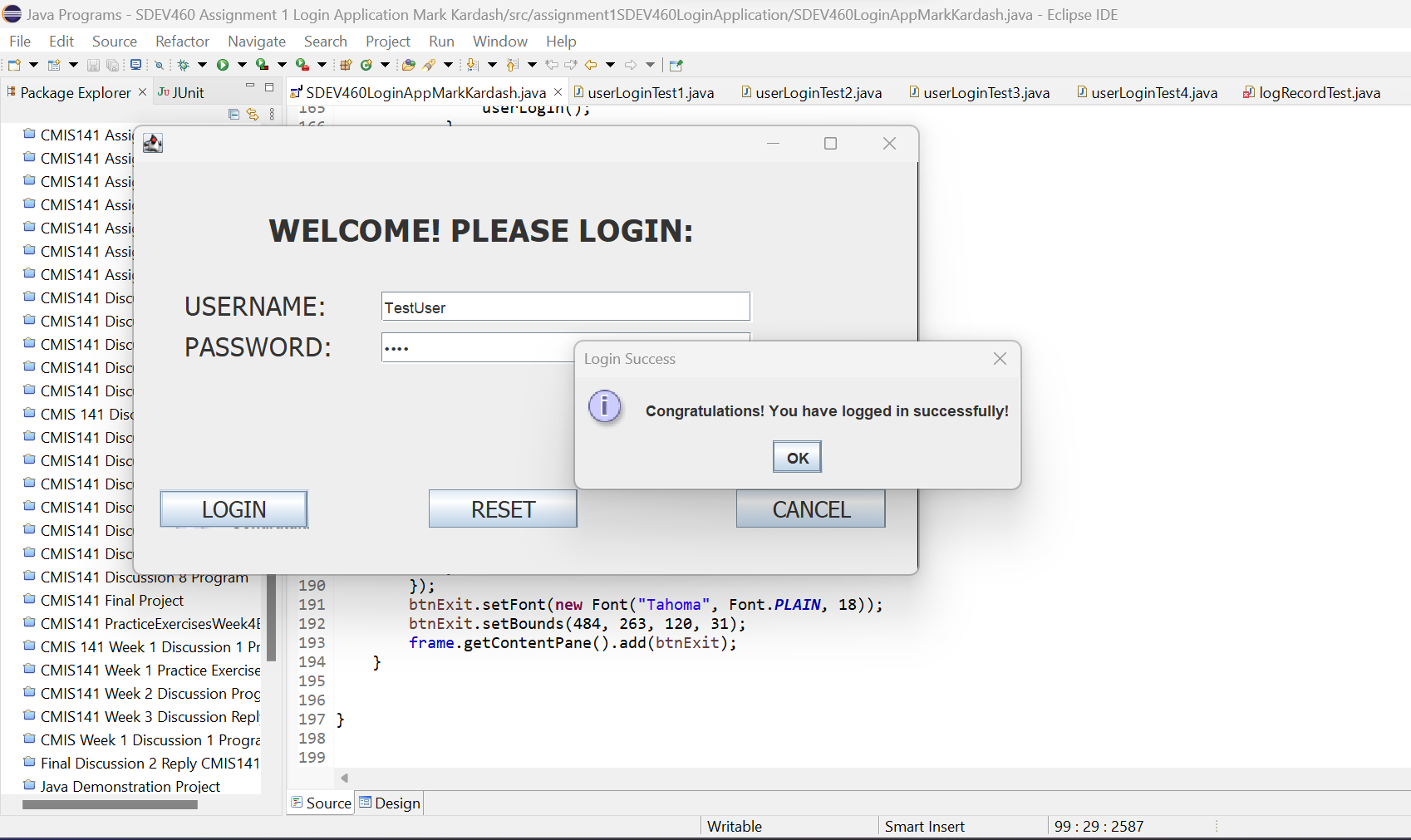


Figure 8: Correct Credentials Result

Entering the correct credentials will give the desired confirmation message.

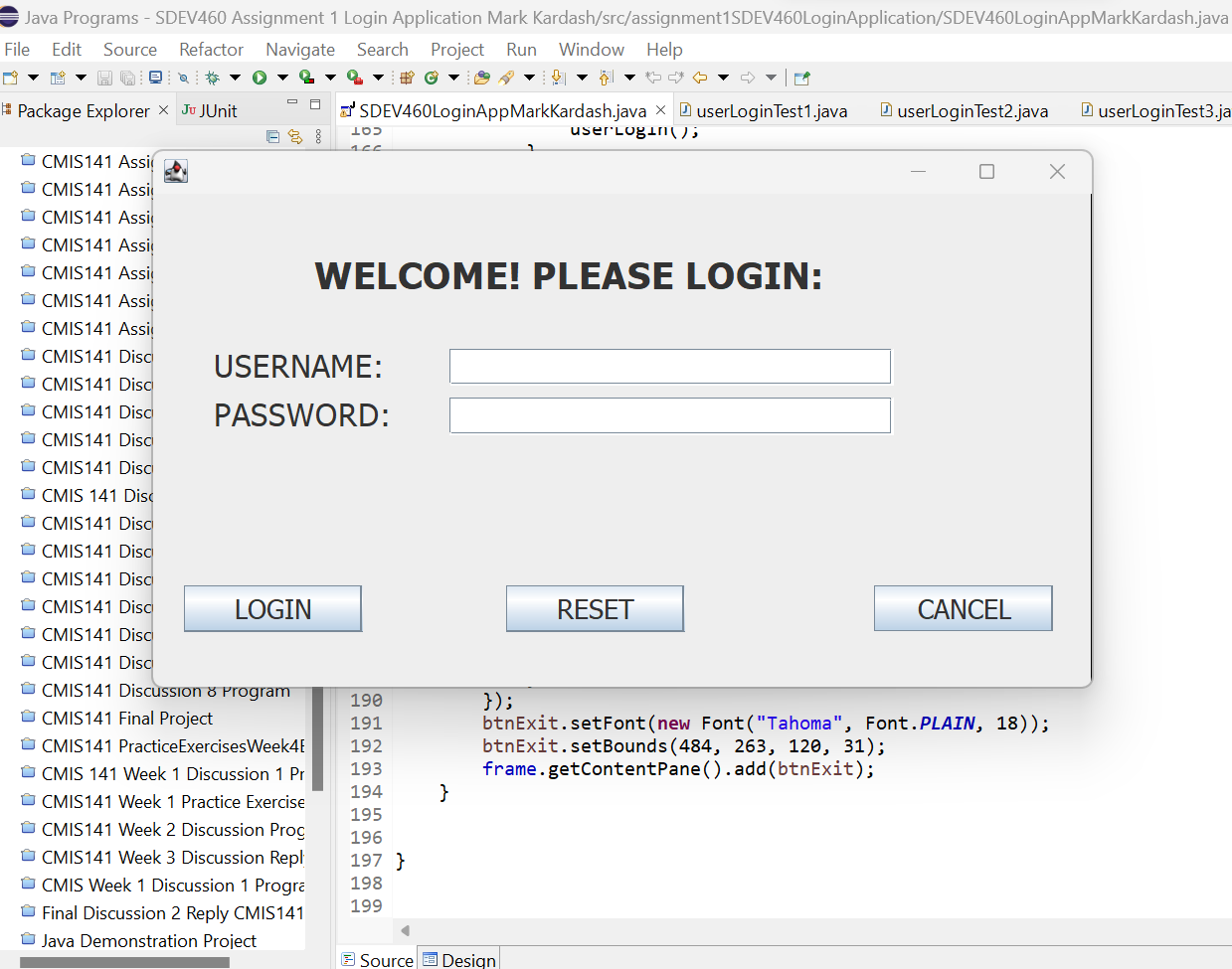


Figure 9: Resetting Credentials

Pressing the “Reset” button empties the credential fields.

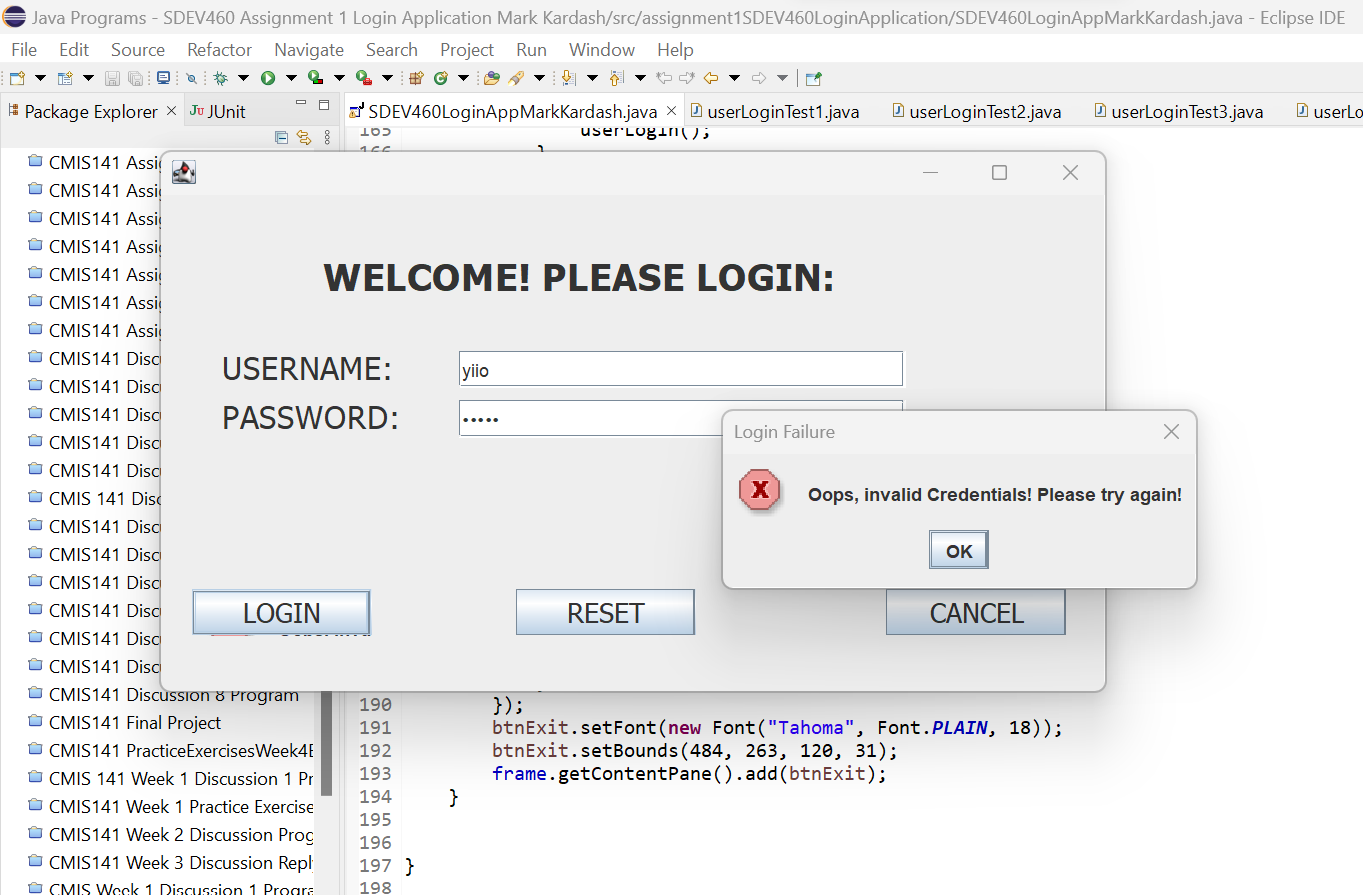


Figure 10: Invalid Credentials (Both)

Putting incorrect credentials into both fields throws the expected “Invalid Credentials” message.

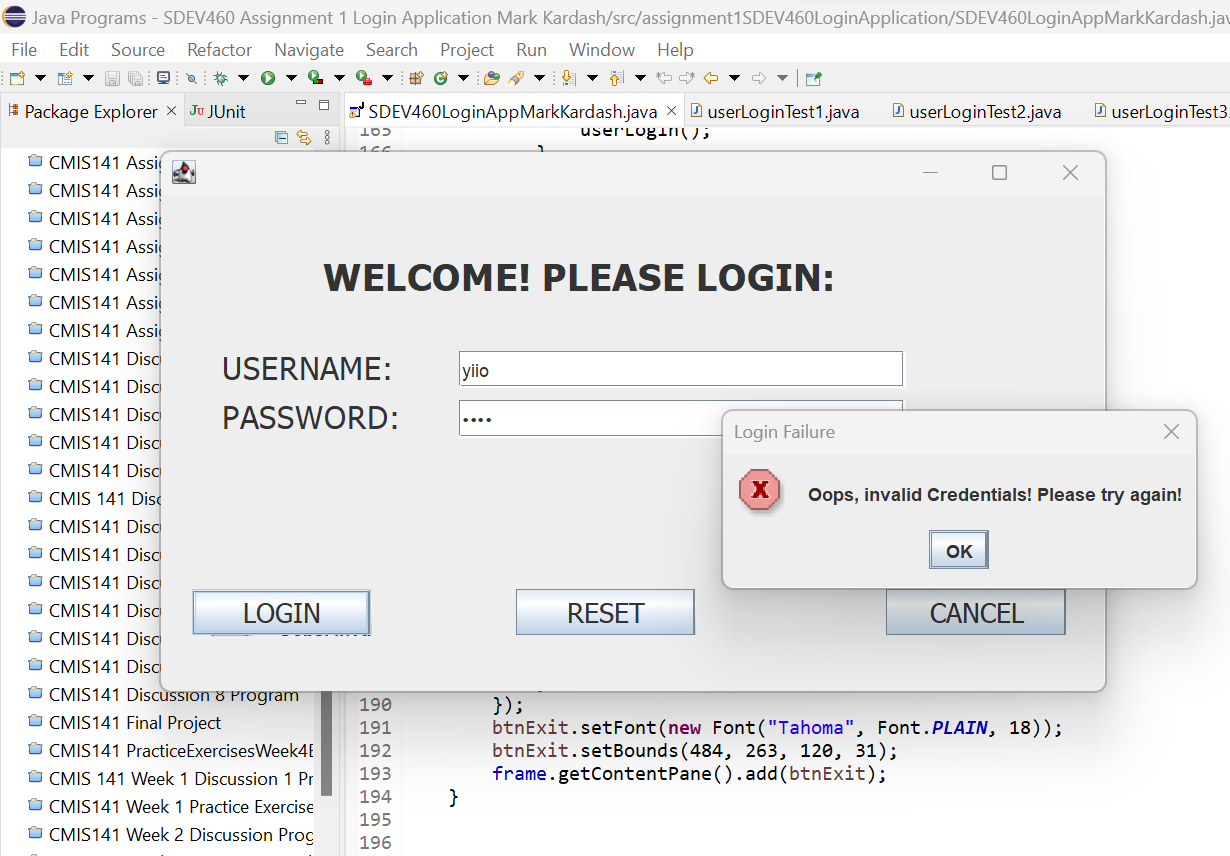


Figure 11: Invalid Credentials (Username)

So does putting only an invalid username.

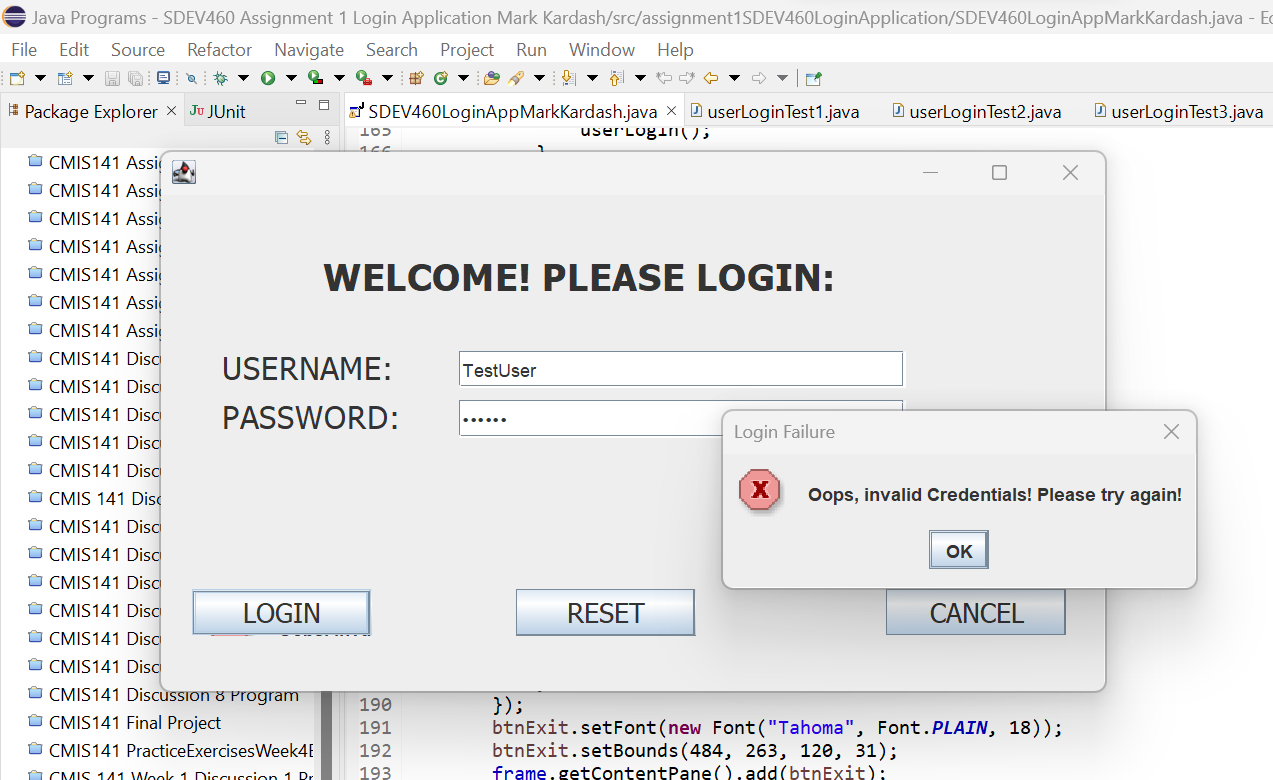


Figure 12: Invalid Credentials (Password)

And so does putting only an invalid password.

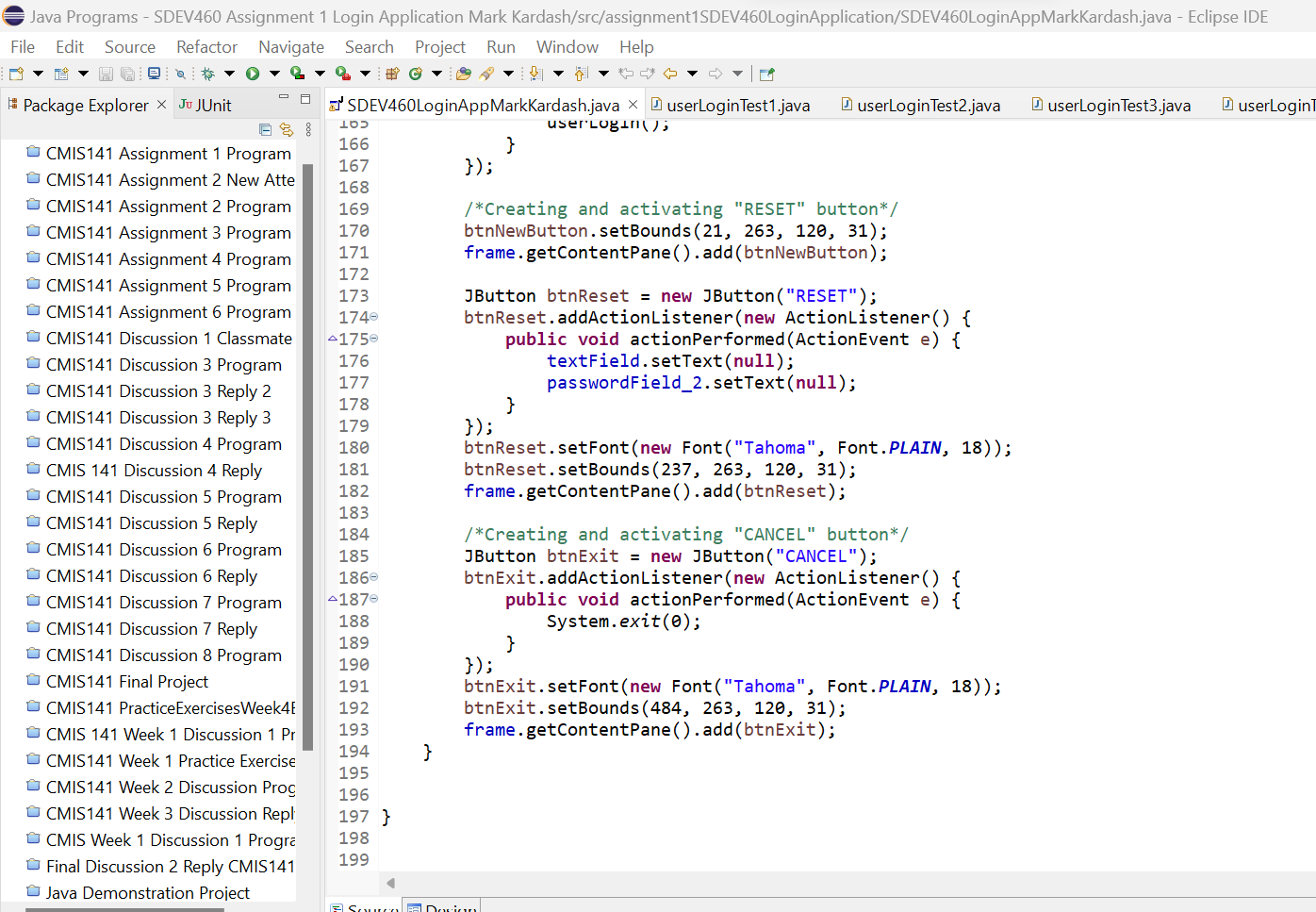


Figure 13: Exiting the program

Clicking “Cancel” causes the program to exit.

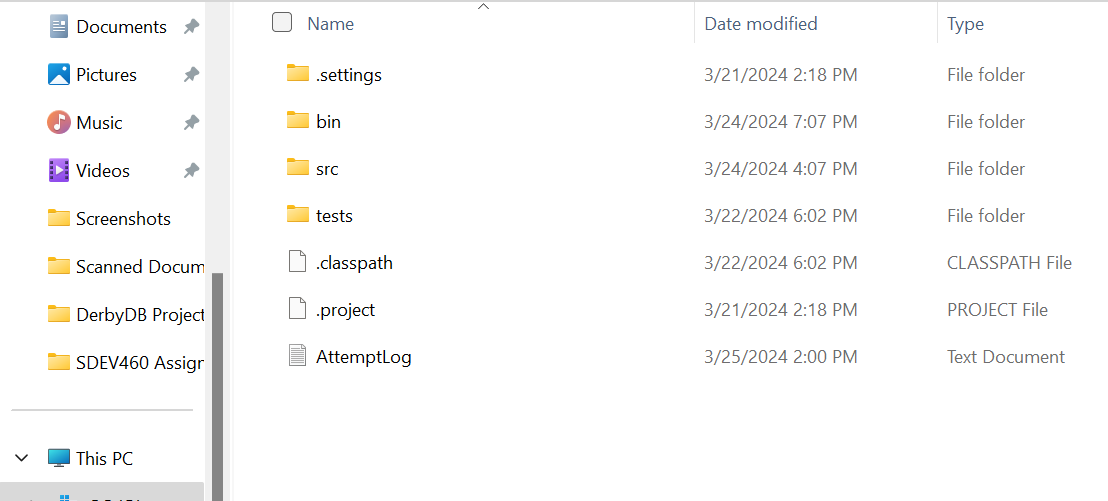


Figure 14: Creating log file.

The creation of the log file also appeared to be successful, as it was present in the file of the program. It also appears to be recording everything correctly, as the time and date of its last modification match those of the latest login.

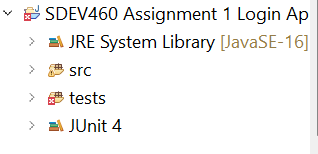


Figure 15: Log not present in Eclipse.

Strangely, however, “AttemptLog.txt” was not present within the Eclipse environment itself.

Unit Test Analysis:

As I explained in the beginning of this document, I did not manage to write proper code for any of the JUnit test cases. However, I will show the code that I did write for them, and explain how it was intended to work.



Figure 16: JUnit Test 1

In this test, we hardcode the correct credentials, and compare them with the credentials hardcoded within the “loginTest” method. This was done to make sure that the user is entering the correct credentials. For this first test, credentials would be the same, and there would be no error.

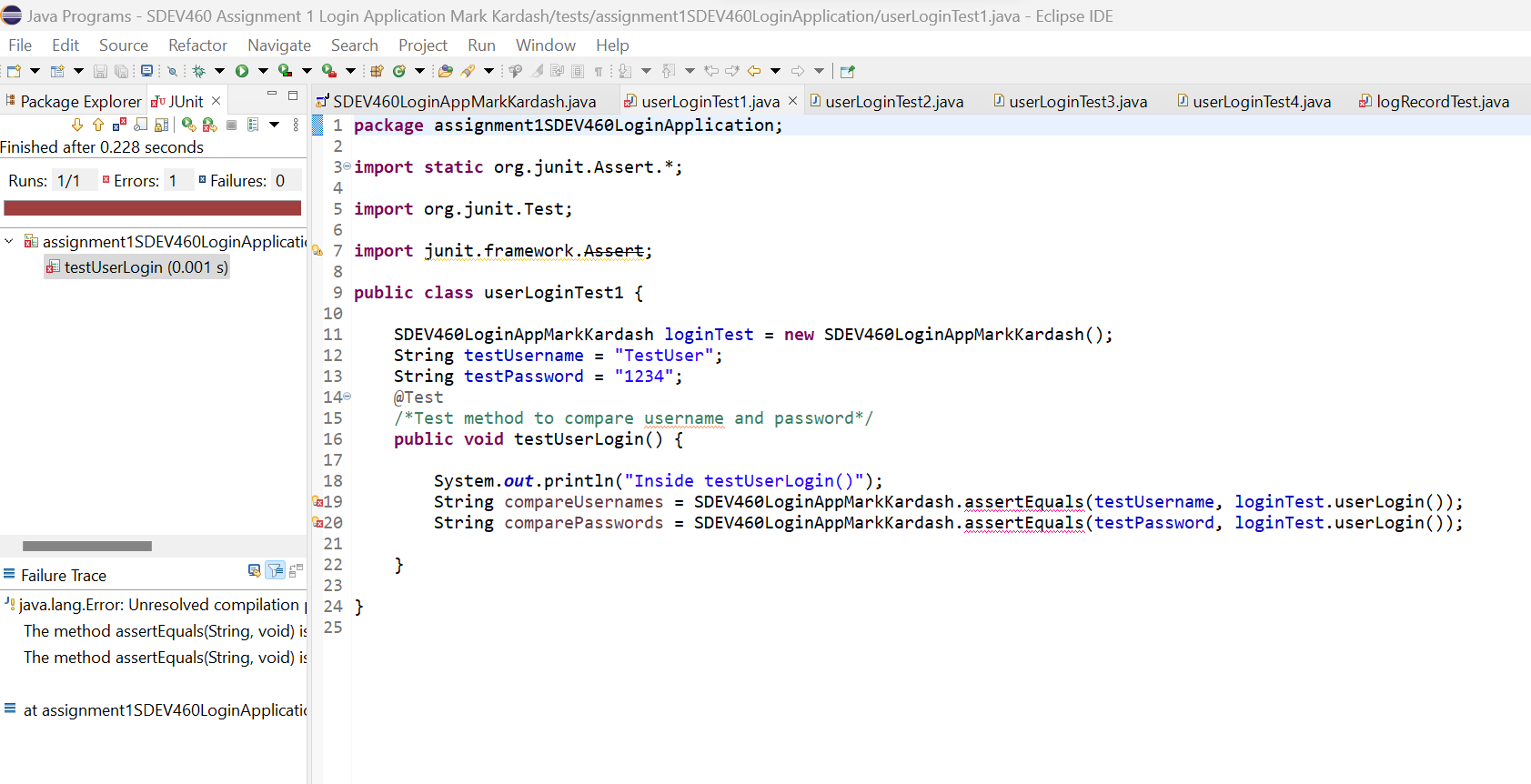


Figure 17: JUnit Test 1 Result.

However, because I was unable to properly link the test to the login method, it expectedly came back with an error.



Figure 18: JUnit Test 2

In the second test, both credentials entered were incorrect, so that, when compared to those expected in the “loginTest” method, they would not match, and the test would throw an error.

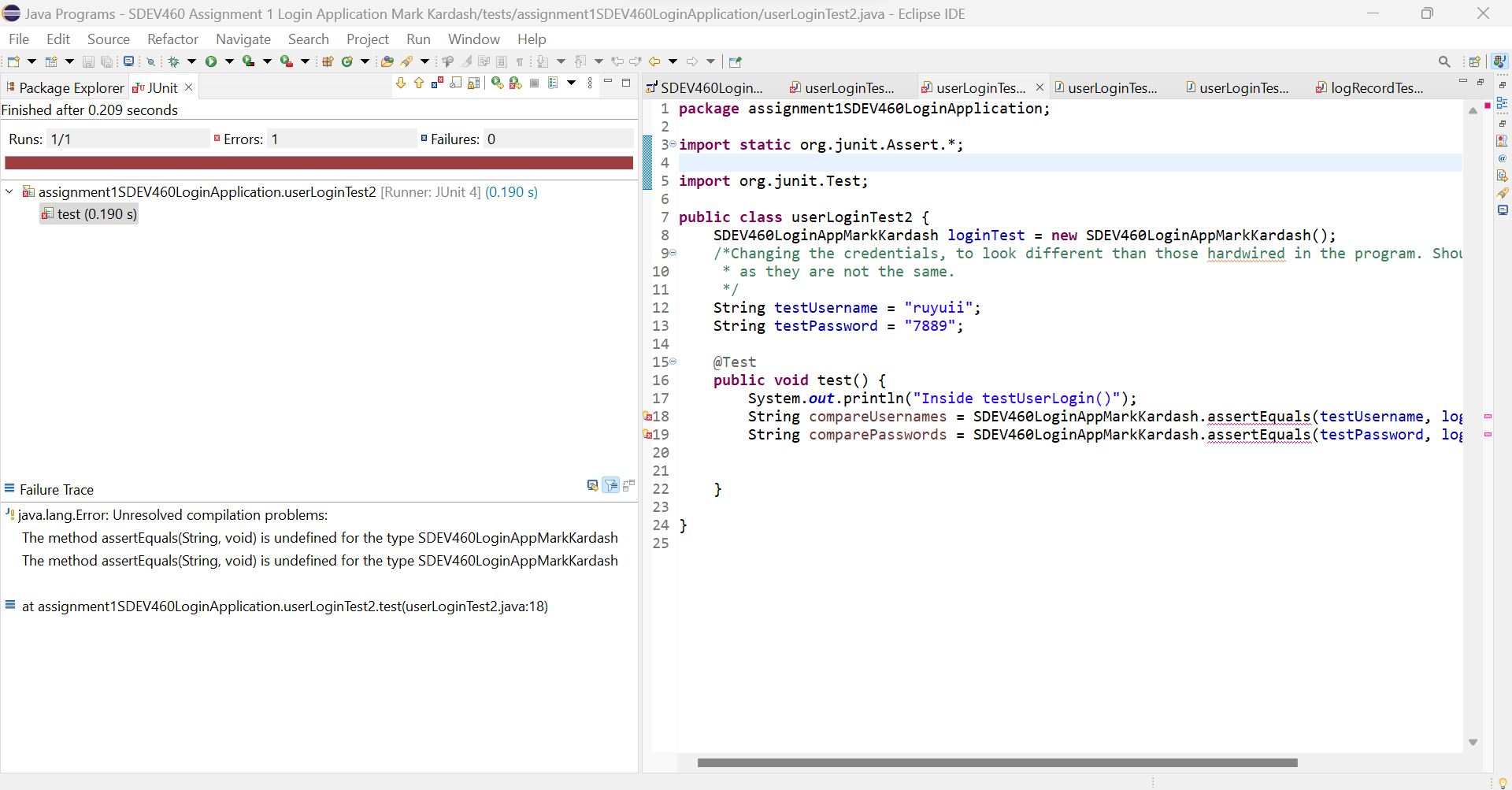


Figure 19: JUnit Test 2 Result

However, the error was thrown for an entirely different reason, as the two methods used were undefined.

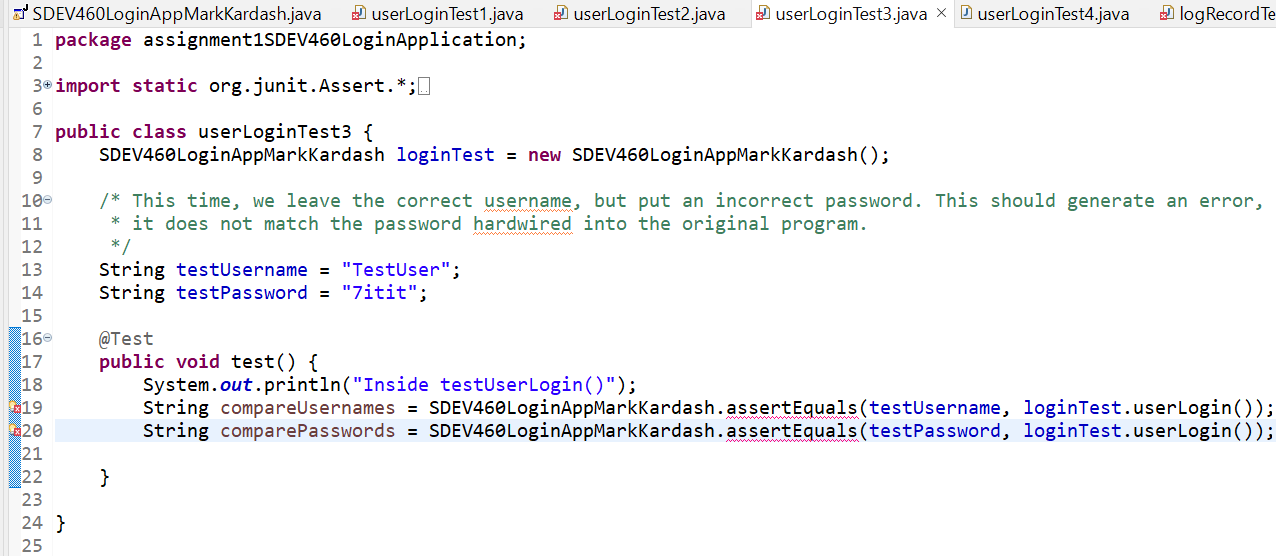


Figure 20: JUnit Test 3

In the third unit test, the password was changed to an incorrect one, to check how the program would function under this condition. Since one of the credentials is not incorrect, the test was intended to, again, show an error that would prove they do not match.

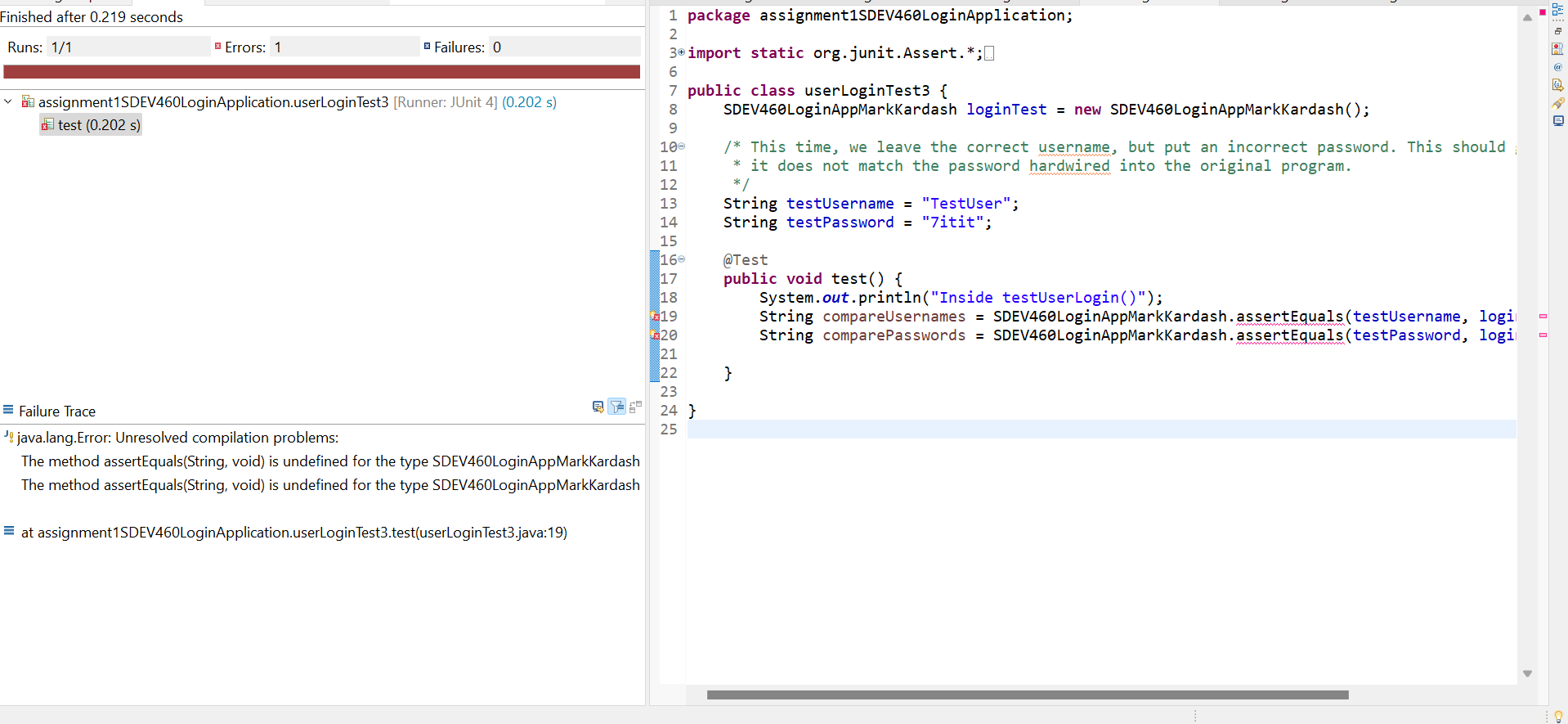


Figure 21: JUnit Test 3 Result

However, as before, the test only resulted in an error because the methods used were undefined.

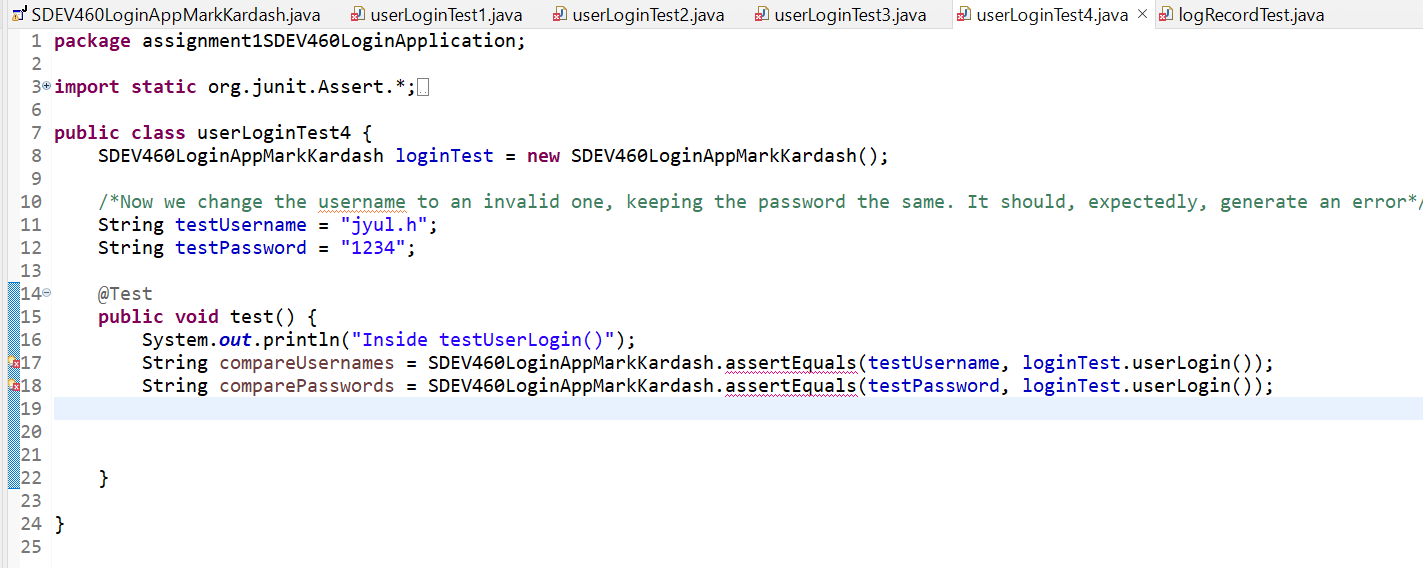


Figure 22: JUnit Test 4

The final login test had the correct password, but an incorrect username hardcoded. It was supposed to, of course, display an error due to them not matching the ones in the “loginTest” method.

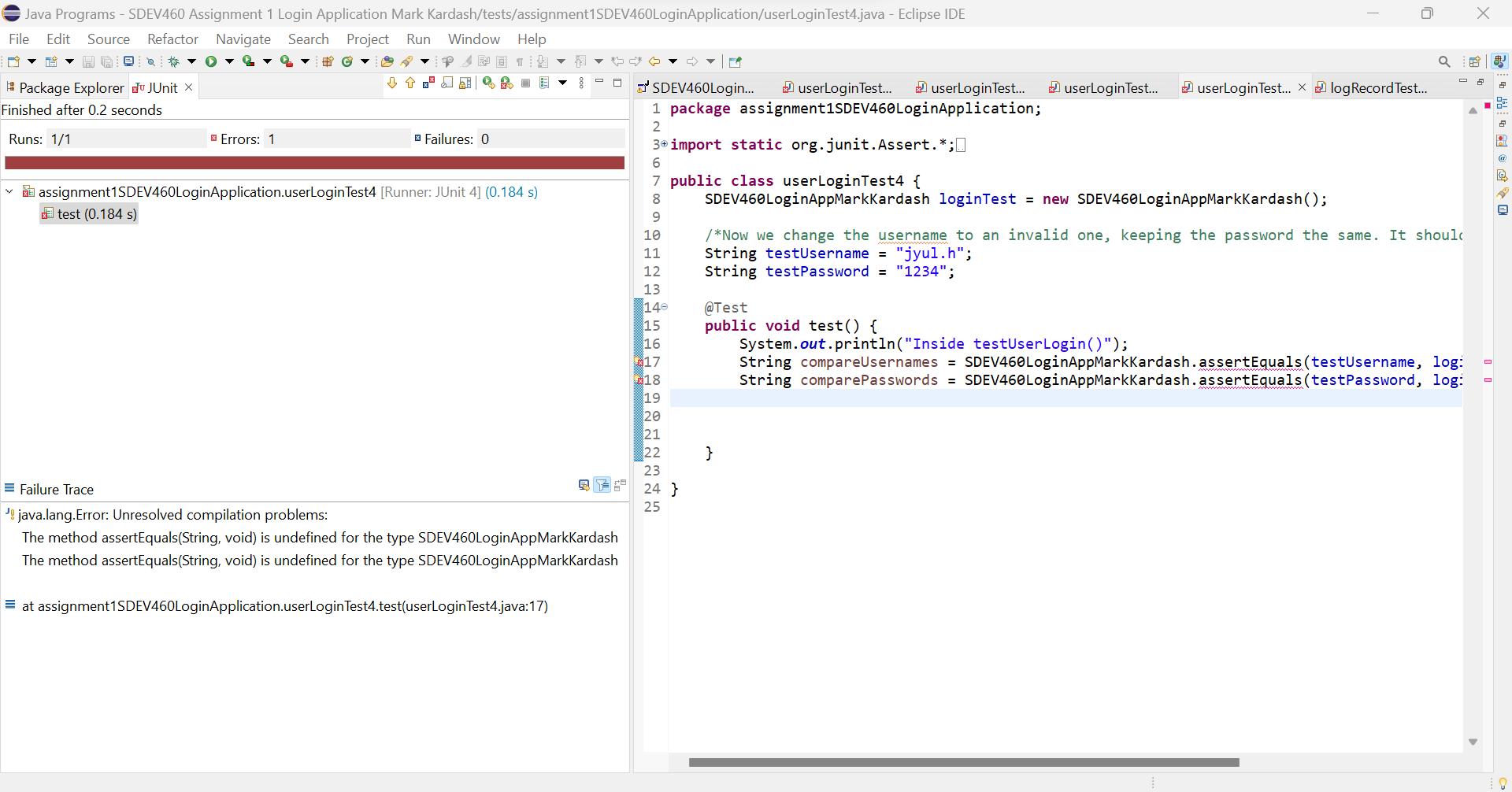


Figure 23: JUnit Test 4 Result

The true error shown was, as before, due to undefined methods, and my failure to correctly call the “loginTest” method.

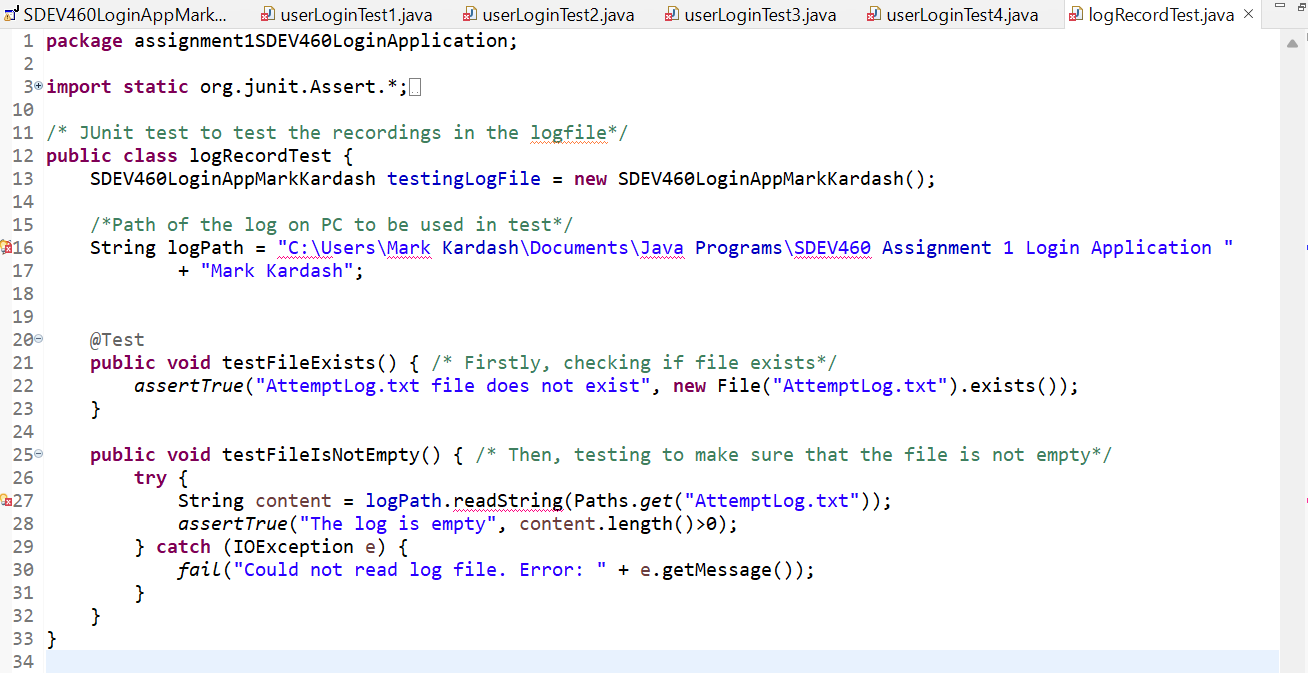


Figure 24: JUnit Test 5

The final test was aimed at the “.txt” log file. First, the test will check if it exists, and then check if it is empty or not. It was intended to do this by creating a string, “logPath”, which was the path to where the “.txt” log file was located on my computer. It then would use the “.exists()” method to check for its presence. Finally, it uses a “try” loop, as well as “readString()” and “content()” methods, to check the contents of the file. I seem to have correctly written the first part of the code, but failed once again on the second. The expected result would originally be for the test to pass completely.

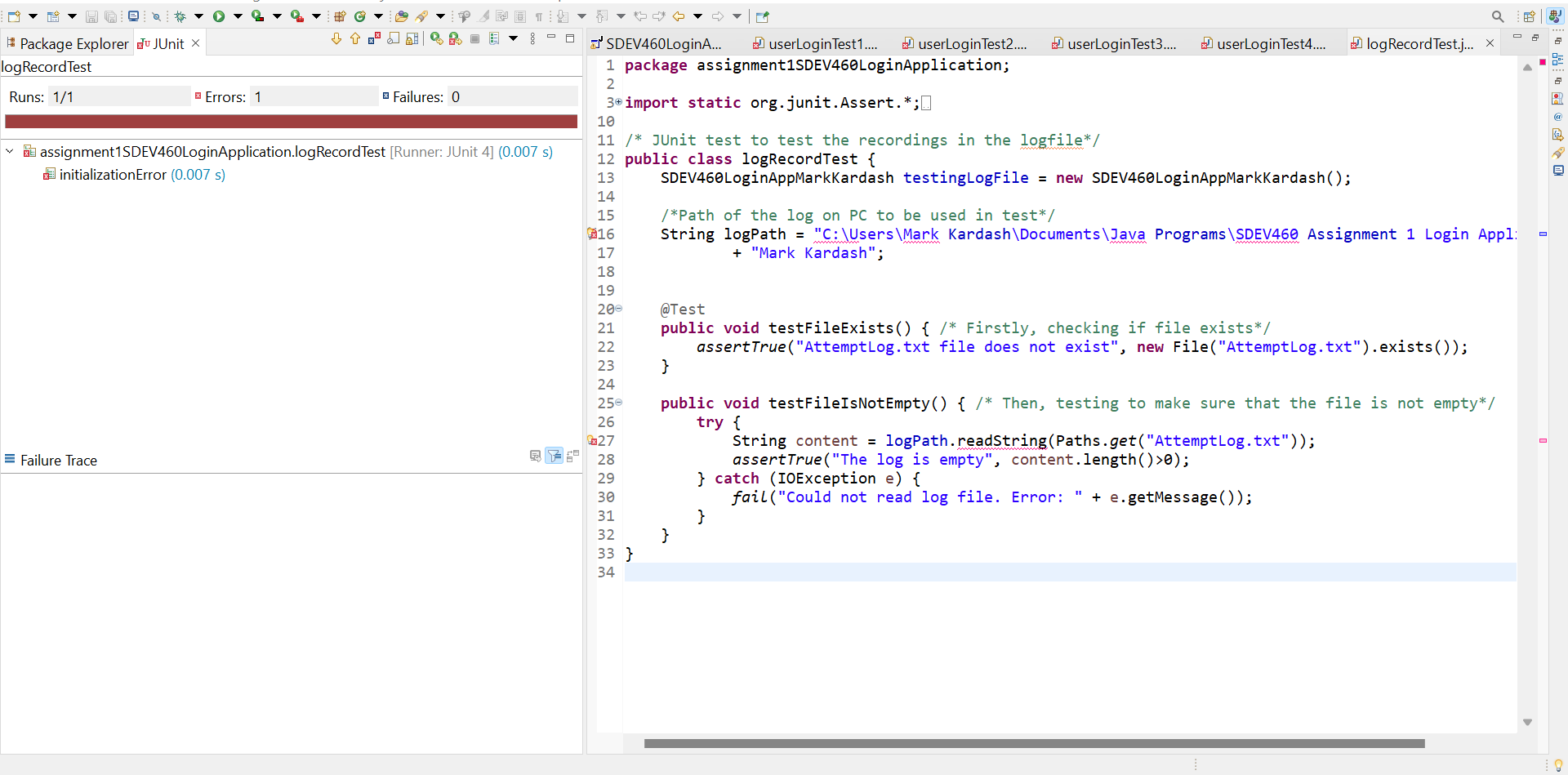


Figure 25: JUnit Test 5 Result

The test returned an error, although the “Failure Track” field was empty. However, it was clear that the test failed due to the inability to apply the “readString()” method, combined with the fact that it did not recognize the file path.

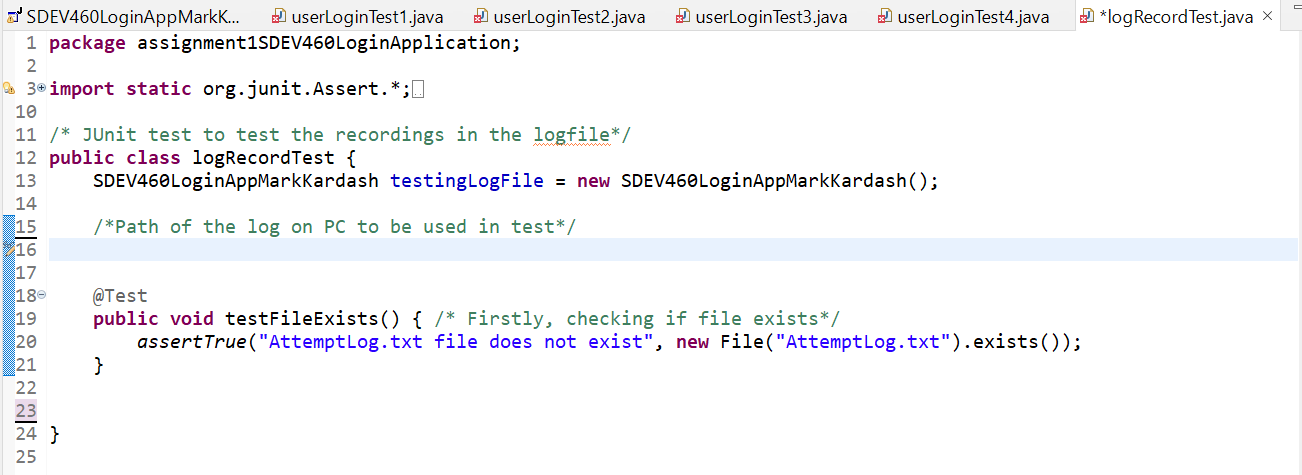


Figure 26: Running only first part of JUnit Test 5

To test my theory that I had gotten the first part of code (the one checking for the existence of the file) correctly, I decided to try and run it on its own.



Figure 27: JUnit Test “testFileExists()” method results.

My theory was proven correctly, making this the only test that passed.

Conclusion:

This seemed like a very easy and fast assignment, but it proved to be far from that. My guess is that I was not able to correctly implement the tests because of the way I designed the program. It simply did not have enough distinct methods for them to be tested individually, and testing the class as a whole did not give the desired results. I will use this experience as a lesson for the future, as to how I should design my programs.