Project 1 Write Up: Dolev Peleg

Approach, Design, and Algorithm

The first thing I did when I approached the project was to read the write up, and the sample write up provided by the professor. Then, I created a stub version for each core method in PasswordCheckerUtility according to the provided Javadoc. I assumed that the core methods of this program are all the methods that check an individual requirement of the password's validity. I believe these methods are: isValidLength, hasUpperAlph, hasLowerAlpha, hasDigit, hasSpecialChar, and noSameCharInSquence. That is because there is no sense in checking if a password is valid(which combines all these methods), if I am not sure that all these methods work individually.

The next thing I did was to start creating the exception, and the implementation of each class, testing them one by one in a driver class. I made sure to move on to the next method only after ensuring that the current method I was working on was working as expected.

Another thing I had to check was the order of the requirements that my program checks. That is, if a password is missing multiple requirements, which exception should be thrown. After going back to the write up that was provided to me, I learned that the expected order (from the first to last) is: sufficient length, has an uppercase letter, has a lower case letter, has a digit, has a special character, and has no two characters repeating in a row. Therefore, I made sure that the isValidPassword calls the core method in the same order: isValidLength, hasUpperAlpha, hasLowerAlpha, hasDigit, hasSpecialChar, noSameCharInSquence.

After testing and ensuring that my program can check for a password validity, I made sure that my hasBetweenSixandNineChars and isWeakPassword methods work as expected. First, I was not sure if I should check for the password's validity in this method. This is because the WeakPasswordException that is thrown by this method has a message saying "The password is OK but weak - it contains fewer than 10 characters." But, after going through the GUI provided by the instructor, I understood that this is not needed, because the validity of the password is checked only by the isValidPassword method.

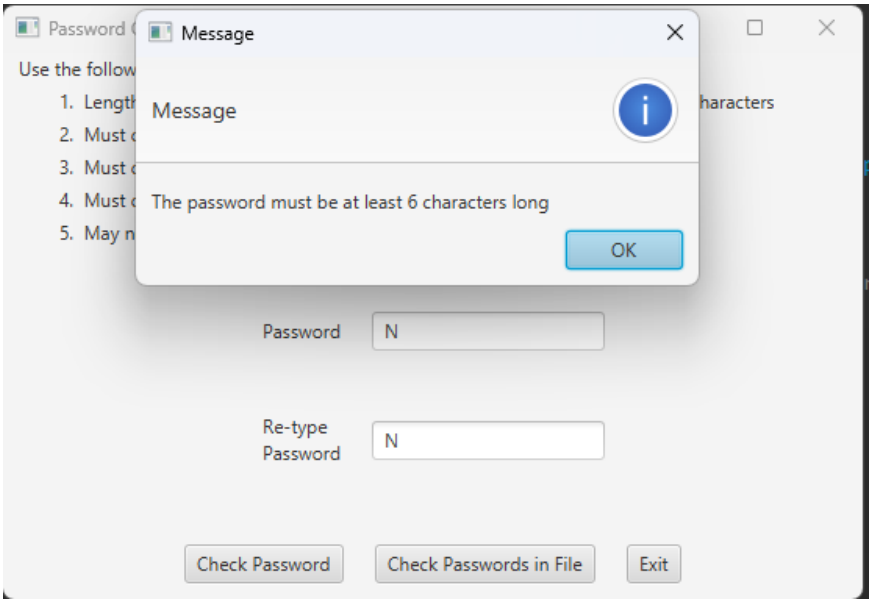
One thing that is important to note is that I followed the JDoc documents, therefore, hasSameCharInSequence returns true if the requirement is met, that is, if the password does not contain 2 characters in sequence.

I never worked with so many custom exception classes before, so first I struggled with knowing when and where they should be thrown. I managed to solve my issues when I ran enough tests, mostly the PasswordChecker_GFA_Test, PasswordCheckerTest_STUDENT, and PasswordCheckerTestPublic classes. Only when all these tests passed, I assumed that my program was ready.

Test Runs and Cases

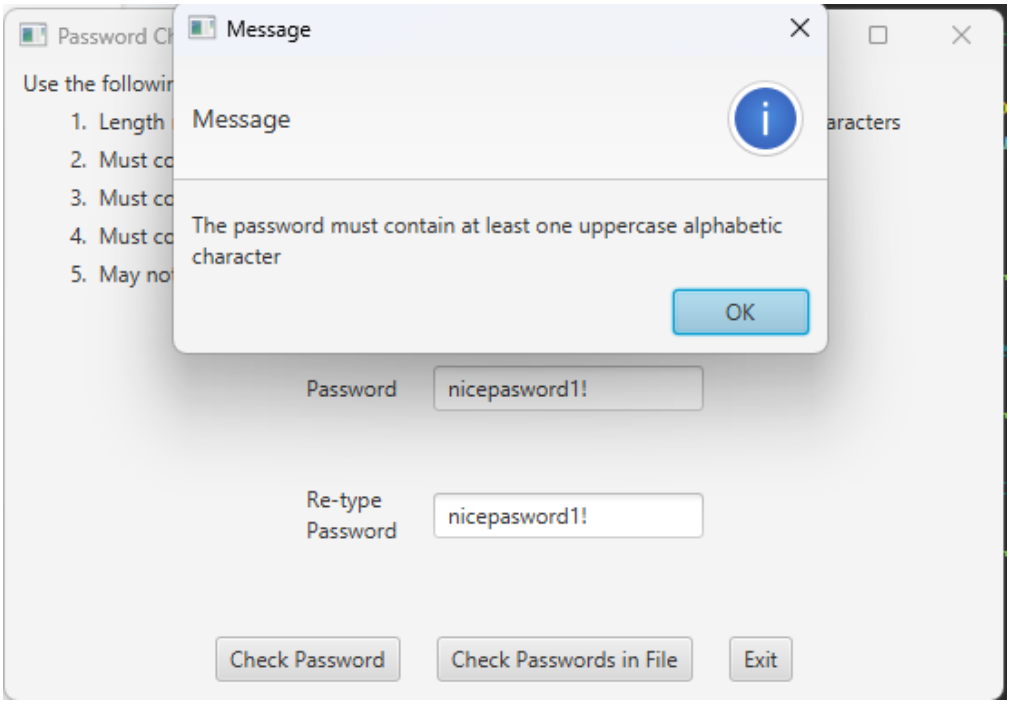
LengthException

Input	Expected Output	Actual Output
Password: N Re-Type: N	The password must be at least 6 characters long	The password must be at least 6 characters long



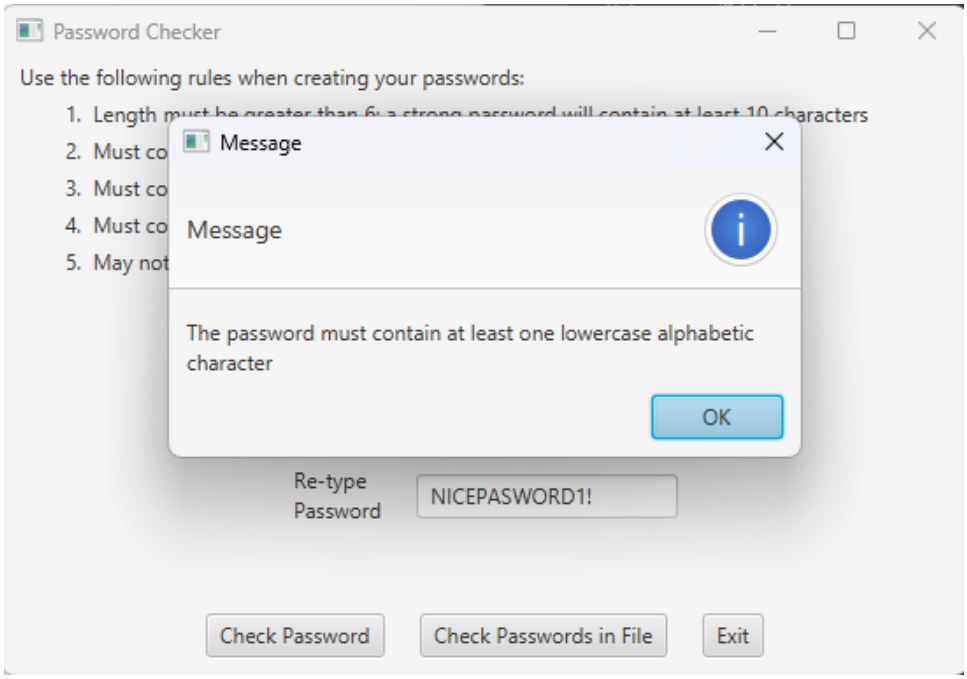
NoUpperAlphaExcpetion

Input	Expected Output	Actual Output
Password: nicepasword1! Re-Type: nicepasword1!	The password must contain at least one uppercase alphabetic character	The password must contain at least one uppercase alphabetic character



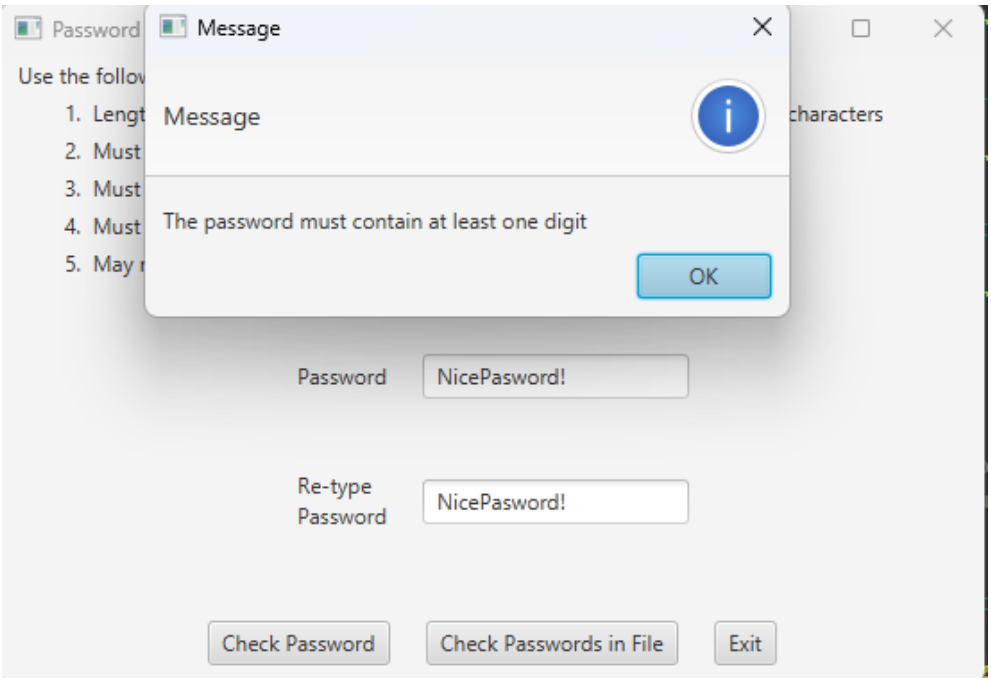
NoLowerAlphaExcpetion

Input	Expected Output	Actual Output
Password:NICEPASWORD1! Re-Type: NICEPASWORD1!	The password must contain at least one lowercase alphabetic character	The password must contain at least one lowercase alphabetic character



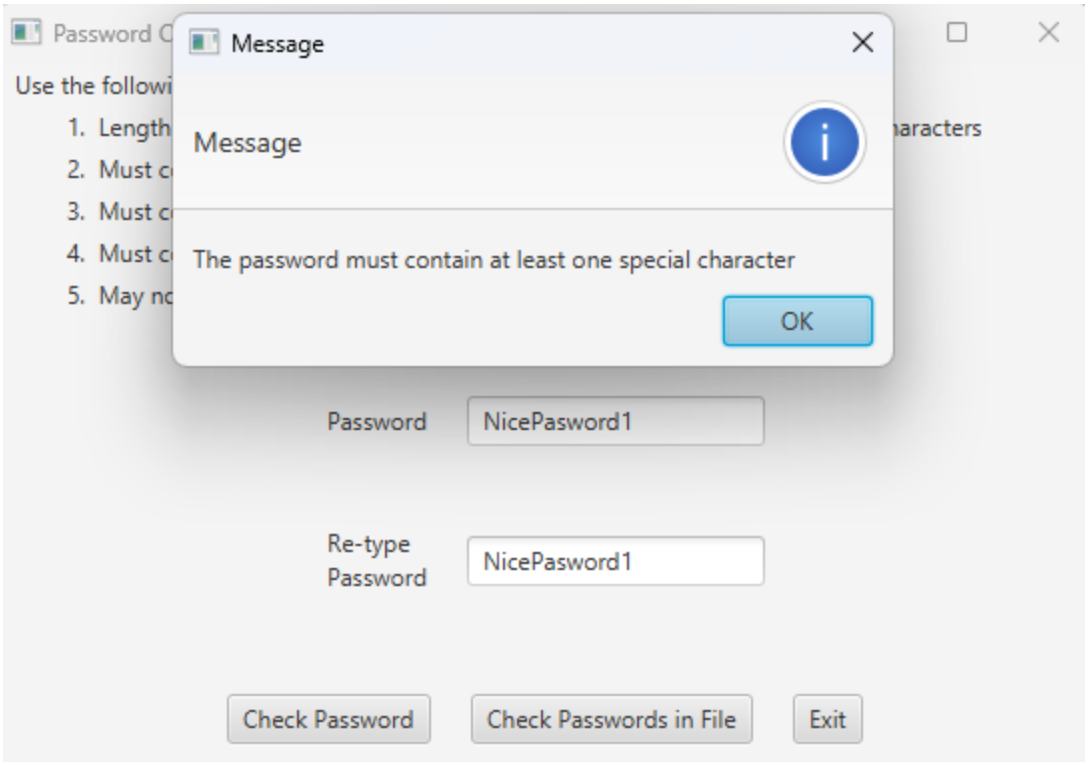
NoDigitExcpetion

Input	Expected Output	Actual Output
Password: NicePasword! Re-Type: NicePasword!	The password must contain at least one digit	The password must contain at least one digit



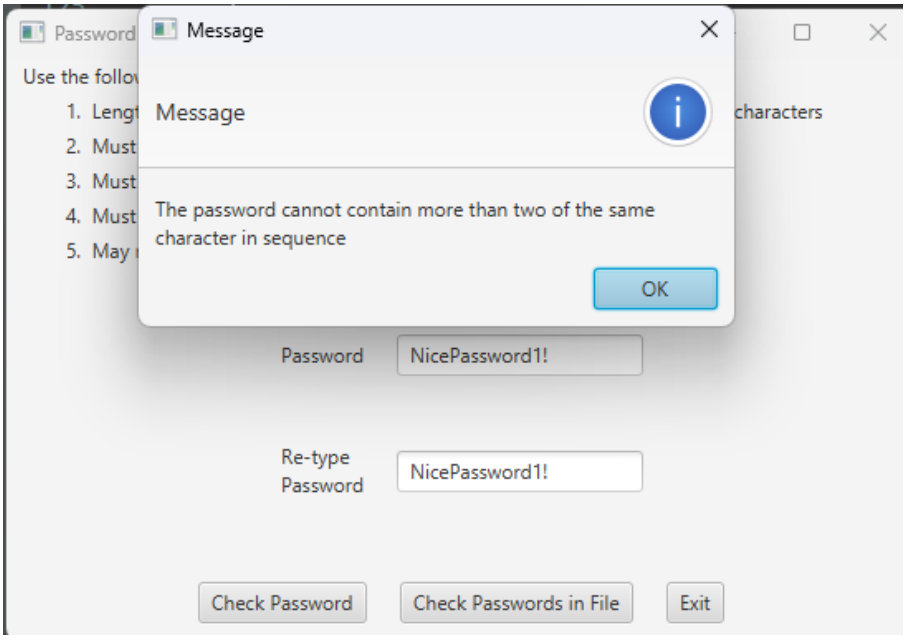
NoSpecialCharacterExcpetion

Input	Expected Output	Actual Output
Password: NicePasword1 Re-Type: NicePasword1	The password must contain at least one special character	The password must contain at least special character



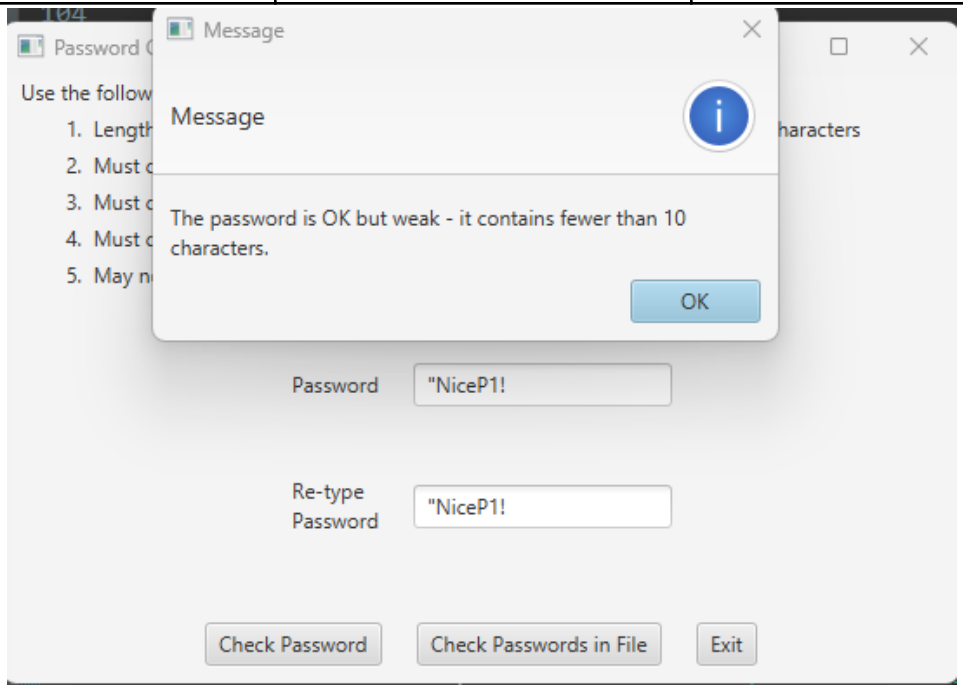
InvalidSequenceExcpetion

Input	Expected Output	Actual Output
Password: NicePassword1! Re-Type: NicePassword1!	The password cannot contain more than two of the same character in a sequence	The password cannot contain more than two of the same character in a sequence



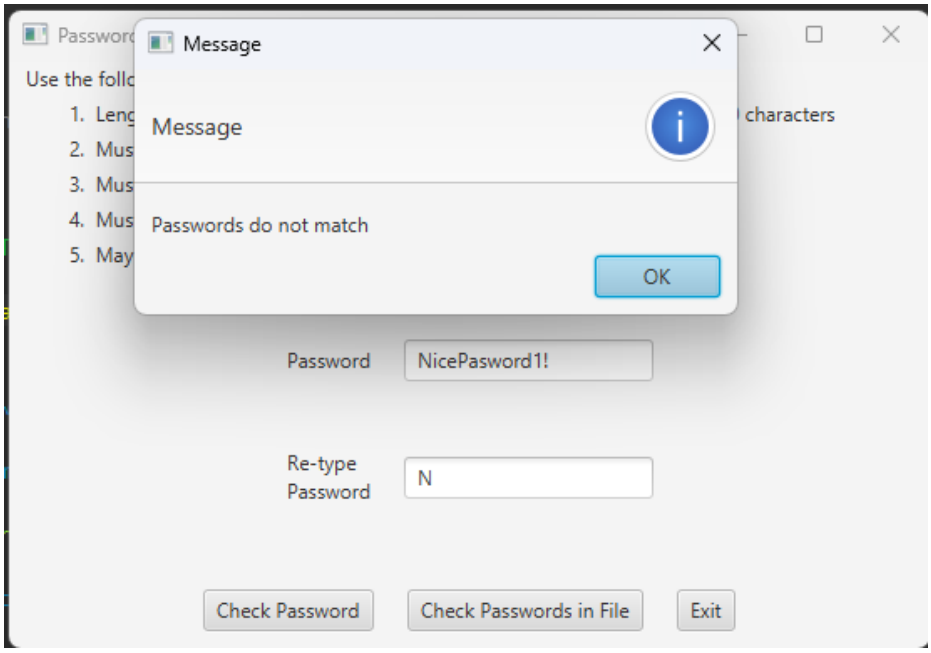
WeakPassowordExcpetion

Input	Expected Output	Actual Output
Password: NiceP1! Re-Type: "NiceP1!	The password is OK but weak - it contains fewer than 10 characters.	The password is OK but weak - it contains fewer than 10 characters.e



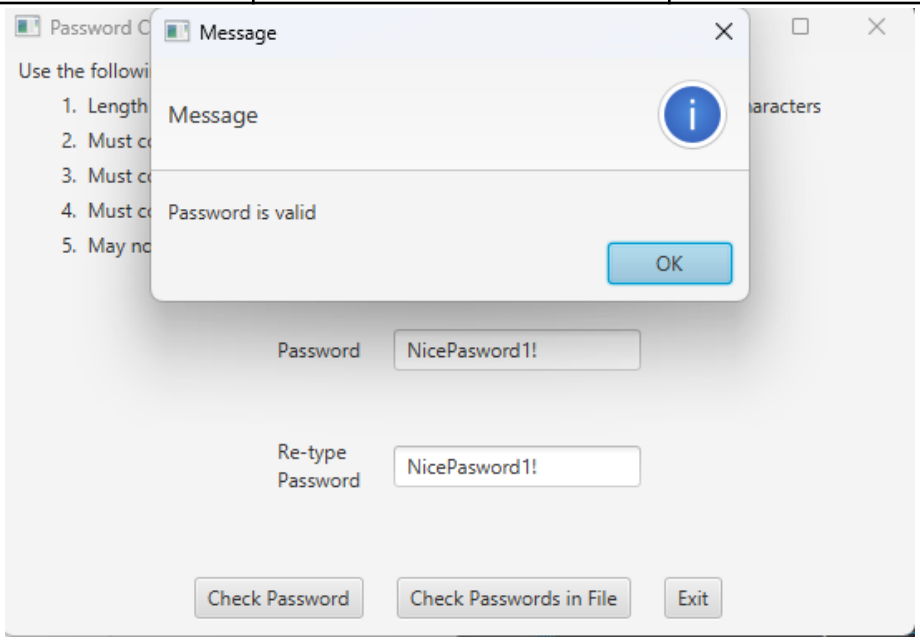
UnmatchedExcpetion

Input	Expected Output	Actual Output
Password: NicePassword1! Re-Type: N	Passwords do not match	Passwords do not match



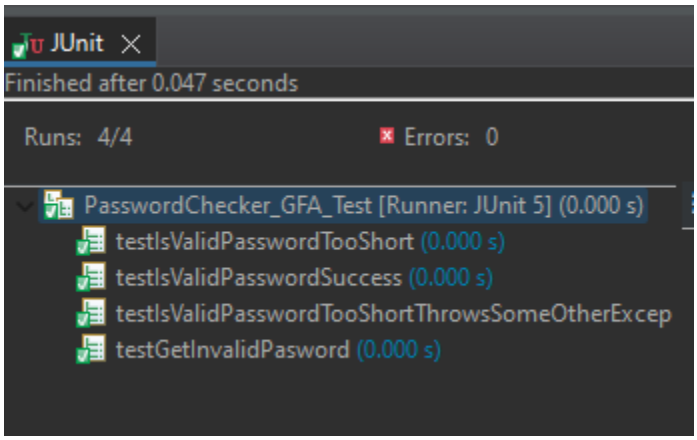
Strong Password

Input	Expected Output	Actual Output
Password: NicePasword1! Re-Type: NicePasword1!	Password is valid	Password is valid

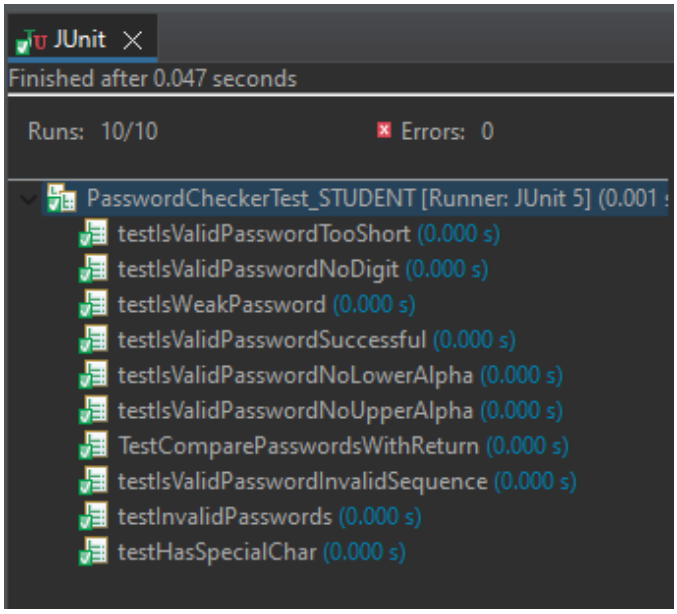


JUnit Testing

GFA (Public) Test Cases



Student Test Cases



Learning Experiences

One thing I did differently in this project was to make sure that I comment on my code while I write it. I used to have a bad habit of first finishing a method, and only then creating and adding the appropriate comments. As my codes get longer and more complex, I have learned that comments do not only help the reader of the code, but also guide me and help me debug and improve my code with more ease.

Working with a code that revolves around multiple exceptions was also a little confusing when I created my PasswordCheckerTest_STUDENT class. That is because I never had to create a test method that is supposed to throw an exception before. I was not sure how to create an assertTrue, or assertEquals that involve an exception. After examining

PasswordChecker_GFA_Test and PasswordCheckerTestPublic which were provided to me, I understood that I can call these methods with the exception's expected message. For Example, assertTrue("Threw another exception besides NoDigitException", false); from the testHasSpecialChar method.

Assumptions

1. The user will be using JUnit 5, Java, and JavaFX
2. Weak passwords are also valid
3. The InvalidFirstCharacter Exception is not in my code because it was not found in the JavaDoc.

Enhancements

- No enhancements were made.