Felix Yu

2/9/20

CMSC204 – Prof. Gary Thai

Project 1 – PasswordCheckerUtility

Lessons Learned

This project focuses on creating a utility class that function to validate created passwords based on various requirements such as length, character and numerical compositions. The utility class also functions to verify whether a created password is considered a weak password. Finally, the class will filter out and aggregate a list of invalid passwords for the user to review. Most importantly, this utility class was designed to respond to missing requirements by throwing different, customized exceptions. JUnit test class was provided to perform smoke testing to ensure exceptions are properly thrown in the case of an invalid password. The backend functionality of the utility class compliments a JavaFX-based GUI for password creation.

The most challenging part of this assignment for me is to figure out how to set up the invalidPasswords method in the utility class, which is designed to return a list of invalid passwords. I originally tried to use an if-else statement to add "invalid" passwords into a new array list that will be returned when the method is invoked. However, during testing, I found that the list of invalid passwords was not captured properly. Therefore, I investigated and realized that checking the condition if PasswordCheckerUtility.isValidPassword(password.get(i)) is true is not correct for this purpose because there is no "false" object returned when the password is invalid. Instead, it'll be more appropriate to try the statement and "catch" any relatable exceptions (LengthException etc.) that are being thrown. Therefore, I removed the if-else statement, and utilized a try-catch statement to supplement the for loop that goes through all passwords from an inputted list. The try-catch statement was able to successfully run the PasswordCheckerUtility.isValidPassword() method for each password and add any invalid password to the new array upon catching any specified exceptions.

From this project, I've learned to utilize a standalone class to verify a String input and produce a hierarchical requirement validation scheme. From these validation logics, I've learned to implement customized exception throwing and messages to handle data processing. Personally, I was the most excited to see that my JUnit test is functional and corroborate the modular function of the utility class code. This is my first time utilizing JUnit testing in completion and I find it very helpful in helping me work on my code in snippets.

From the GUI perspective, I was able to resolve any error in the code regarding the input of the GUI components by adding the javafx11 files into the build path. It was helpful to walkthrough the build path configuration. I became more familiarize with troubleshooting any similar issues and adding new JAR files and libraries to the path. However, I experienced difficulty launching the Java UI. Although I was able to run the PasswordDriveFX class (.java file) as a Java Application, I was unable to launch the UI for some unknown reason. I was only able to see the launching of a Java icon on my desktop without any UI windows opened. I have crossed checked my observation with the instructor and believe there may be

something wrong with the configuration within my Eclipse IDE. I will continue to look into this issue and ensure that future applications after this project can be accessed successfully.

Assumptions

- The requirements described in the documentation are the only rules that the password should strictly follow. There should be no restrictions on types of alphabet, white spaces or symbols.
- Input files will all be .txt format and contains at least one password
- "Weak" but otherwise valid passwords will not be recognized as a problem by the GUI since an exception will not be thrown.
- Since the function of the program is to check passwords for validity, passwords will be kept in plaintext and will not need character masking

Pseudocode

In class PasswordCheckerUtility:

- Method isValidPassword
    - Check If password's length is fewer than 6 characters, throw a customized exception when condition not satisfied
    - Check if password contains any upper case alphabets by using hasUpperCase boolean method, throw a customized exception when condition not satisfied
    - Check if password contains any lower case alphabets by using hasLowerCase boolean method, throw a customized exception when condition not satisfied
    - Check if password contains any numbers by using re-expressions the .match("(?=.*[0-9]).*") method, throw a customized exception when condition not satisfied
    - Check if password contains any repeated characters by using for loop to compare adjacent characters in the password and determining if they are equivalent (same number, same alphabet, same case etc.), throw a customized exception when condition not satisfied
    - If no exceptions are thrown due to unsatisfied conditions, then return true to verify password's validity
- Method isWeakPassword
    - Examine all characters in the String
    - Return true if the password is not at least 10 characters in length
- Method invalidPasswords
    - Examine all passwords inputted from an array
    - Return an array of passwords that led to exceptions being thrown
- Inner exception class LengthException
    - Extend RuntimeException
    - To be thrown if a read password has fewer than 6 characters
    - Error message: "The password must be at least 6 characters long"
- Inner exception class NoUpperAlphaException
    - Extend RuntimeException
    - To be thrown if a read password has no uppercase characters
    - Error message: "The password must contain at least one uppercase alphabetic character"

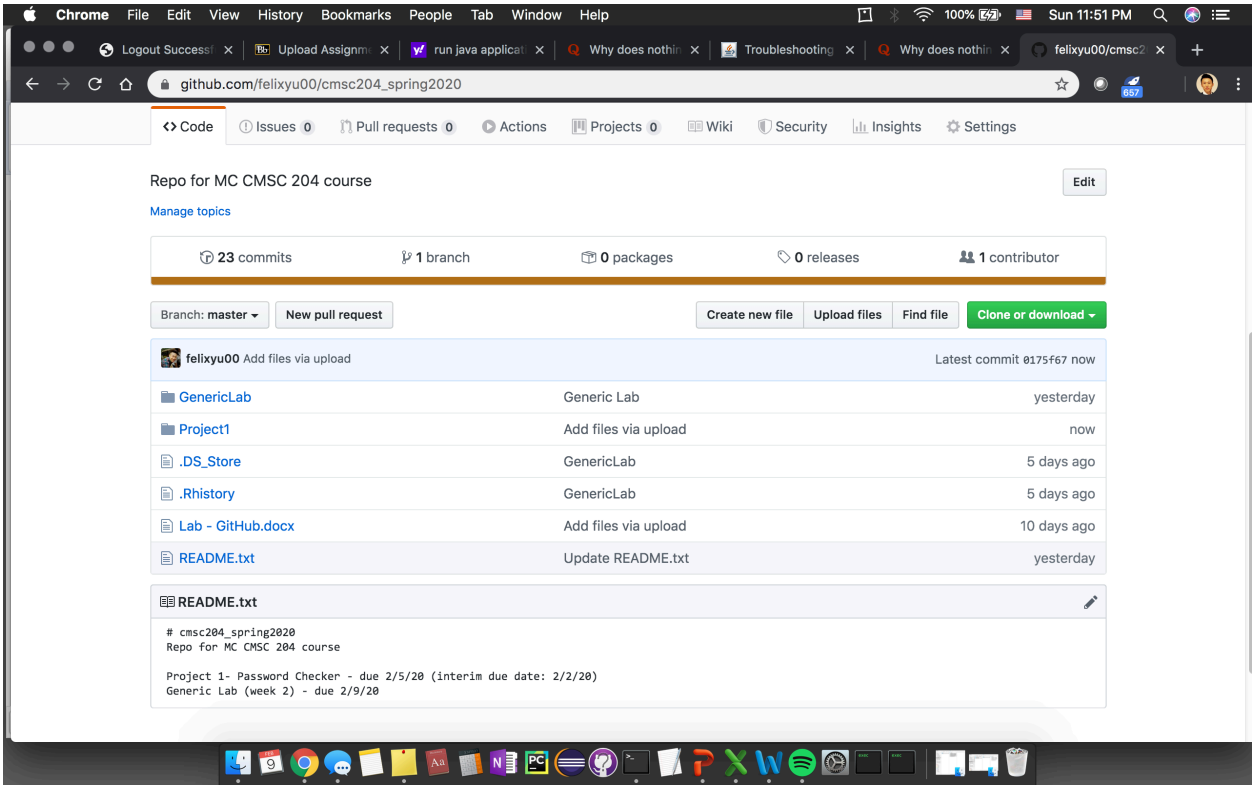
- Inner exception class NoLowerAlphaException
    - Extend RuntimeException
    - To be thrown if a read password has no lowercase characters
    - Error message: “The password must contain at least one lowercase alphabetic character”
- Inner exception class NoDigitException
    - Extend RuntimeException
    - To be thrown if a read password has no numerical digits
    - Error message: “The password must contain at least one digit”
- Inner exception class InvalidSequenceException
    - Extend RuntimeException
    - To be thrown if a read password has repeating characters in the String
    - Error message: “The password cannot contain more than two of the same character in sequence”

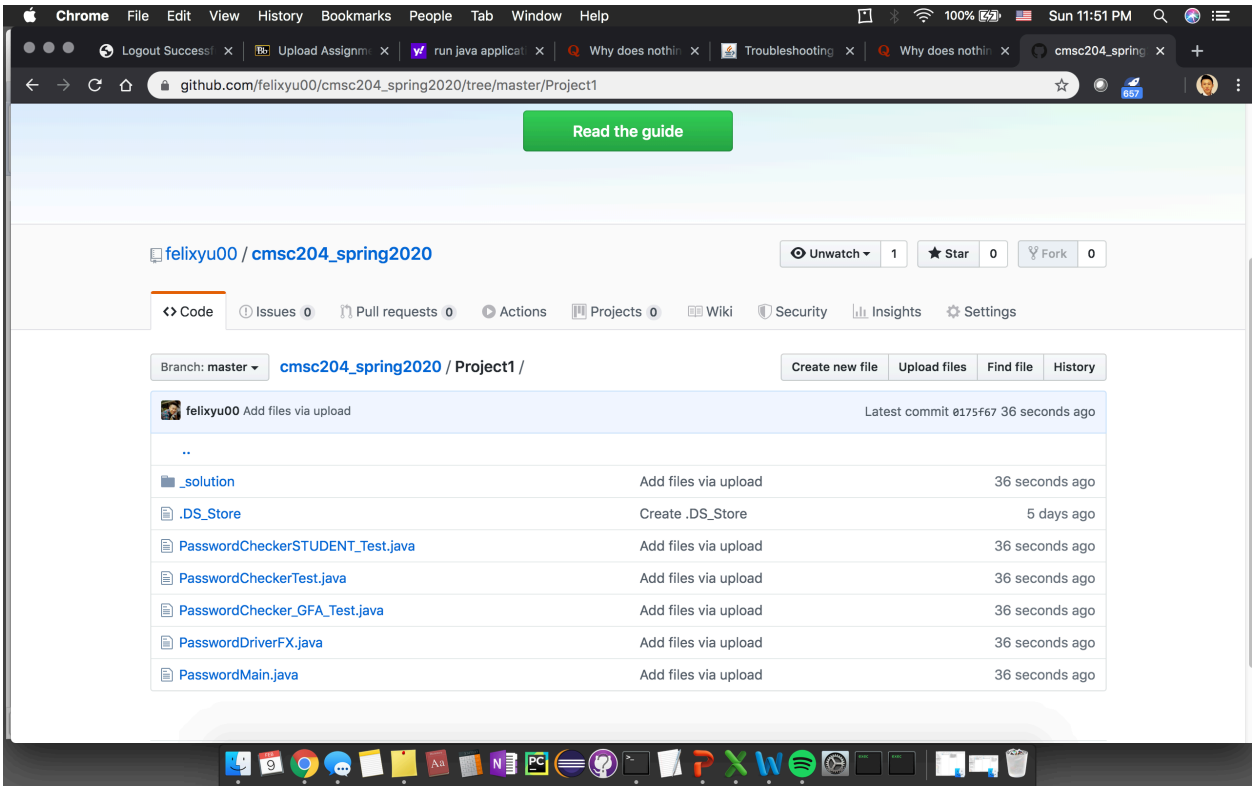
Test Plan

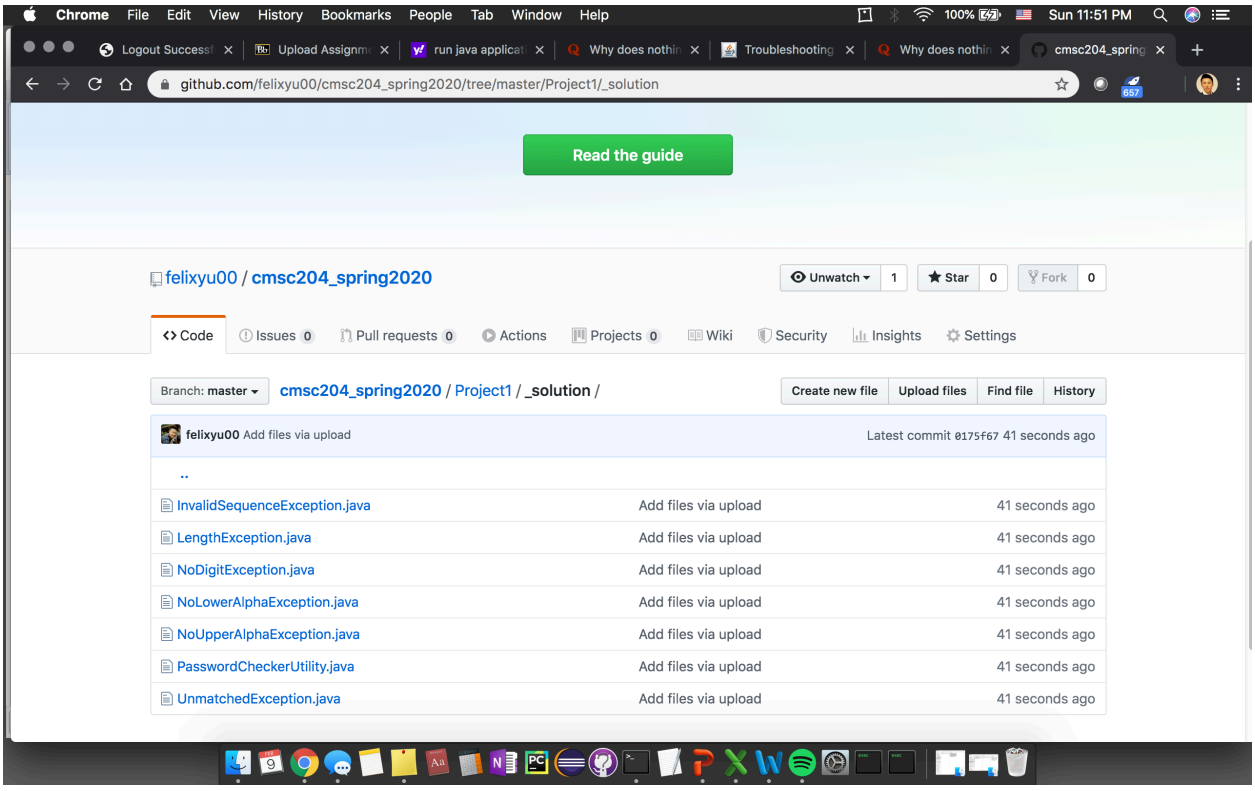
Test #	Input	Expected Output	Actual Output	Pass?
1	Entry: Felixyu0618 Re-entry: Felixyu0618	Valid password window	Valid password window	Y
2	Entry: Fel12 Re-entry: Fel12	Fail: not 6 characters	Fail: not 6 characters	Y
3	Entry: FelixYuChiuHang Re-entry: FelixYuChiuHang	Fail: no digit	Fail: no digit	Y
4	Entry: felixyuchiuhang123 Re-entry: felixyuchiuhang123	Fail: no uppercase letter	Fail: no uppercase letter	Y
5	Entry: HELLOWORLD123 Re-entry: HELLOWORLD123	Fail: no lowercase letter	Fail: no lowercase letter	Y
6	Entry: FelixxxYu123 Re-entry: FelixxxYu123	Fail: repeated characters	Fail: repeated characters	Y
7	Entry: FelixYu123 Re-entry: FelixYu124	Fail: passwords must match	Fail: passwords must match	Y

8	Entry: (blank) Re-entry: (blank)	Fail: must be at least 6 characters long	Fail: must be at least 6 characters long	Y
9	Read File: passwordTest.txt Note: file contains all valid passwords	Pass: No invalid passwords found	Pass: No invalid passwords found	Y
10	Read File: passwordTest2.txt Note: file contains four invalid passwords and one valid password	Pop up window with invalid passwords and their Exceptions. Valid password is not included	Pop up window with invalid passwords and their Exceptions. Valid password is not included (see screenshot)	Y
11	Entry: Felix123 Re-entry: Felix123	Valid password, but weak password	Valid password, but weak password	

GitHub Repository Screenshots



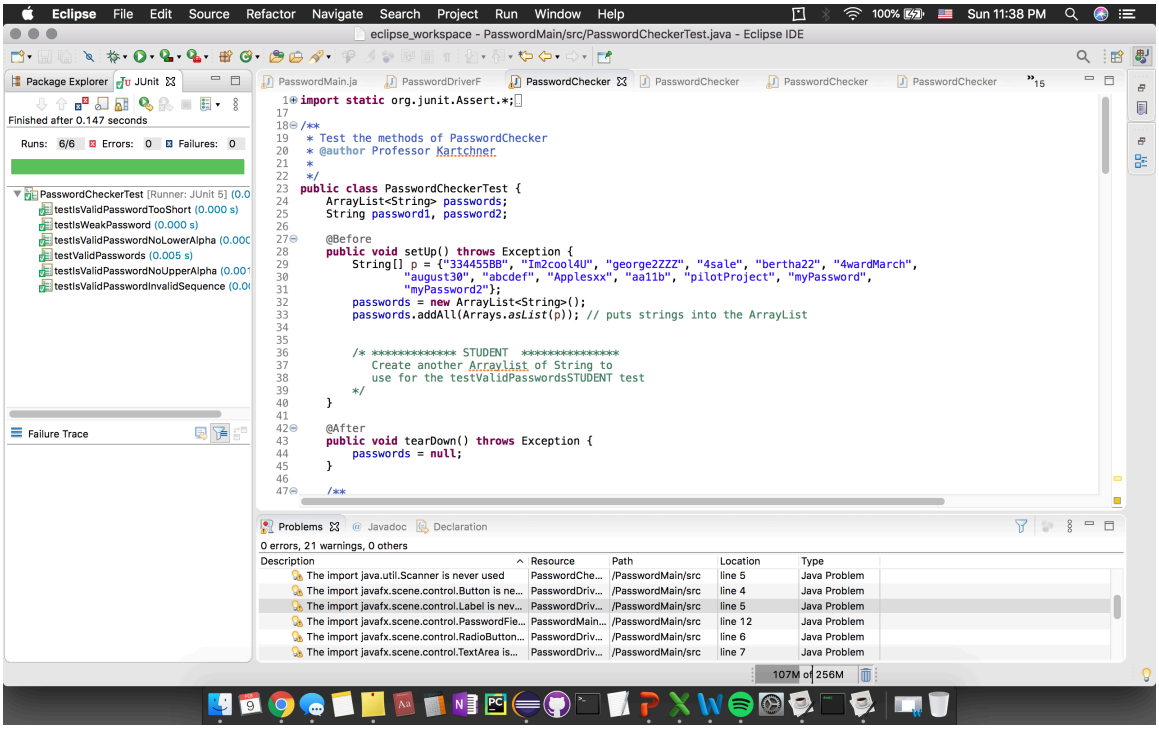




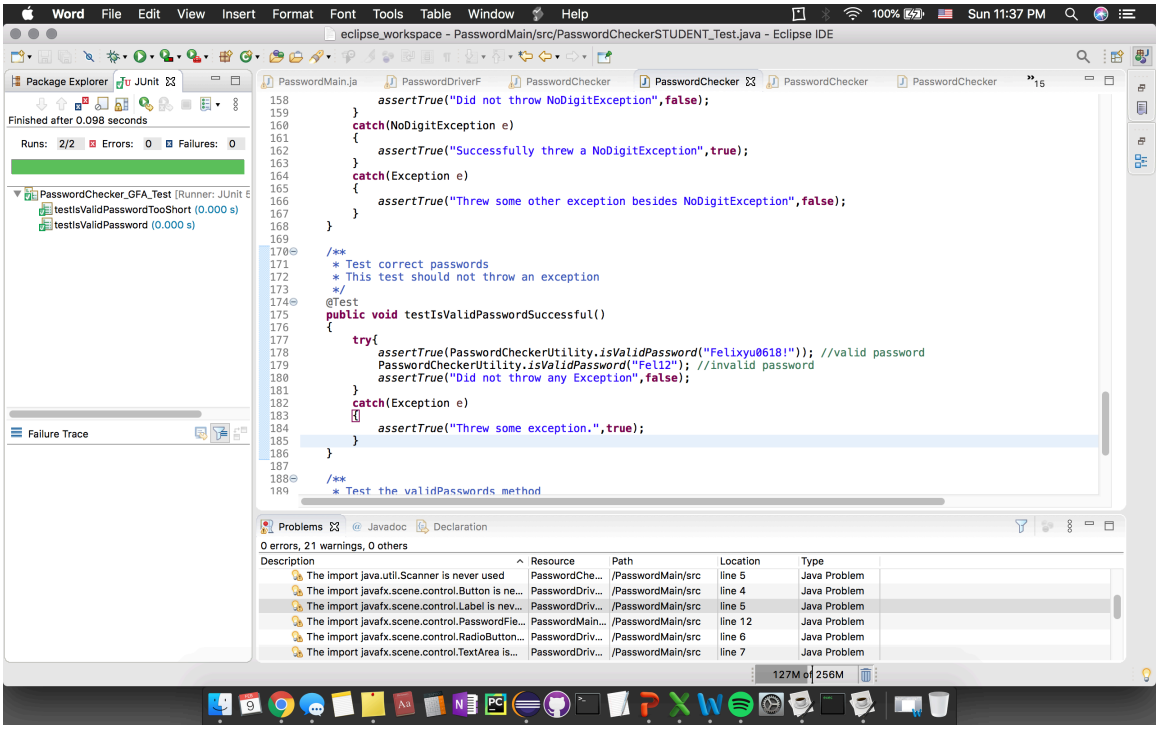
Test Screenshots

JUnit Testing

Test:



GFA_Test:



STUDENT_TEST: