Approach, Design and Algorithm

First, I went over the write up about the assignment as well as looking at the Javadoc to see the methods and what each method consists of. I added all the java files over to my new project in Eclipse.

I figured the easiest way to start would be the exceptions. I looked at the layout of the LengthException and mirrored that for every exception. I copied each message from the write up to the correct exception.

Next, I began working on the the PasswordUtilityChecker class. I looked at the Javadoc to see the details of the methods along with the order. I ignored the comparePasswords method in the beginning since I felt it would be more work compared to the rest of the methods. I started with isValidLength which was pretty straight forward. I just checked if the password length was equal to or greater than 6. If not, throw the LengthException. For the hasUpperAlpha method, I started off with two variable's, a char and an int. Then I used a loop to run through each character of password. I used an if statement to see if the character was an uppercase. If so, increase the count by one. If there is 0 uppercases, it throws NoUpperAlphaException. I then copied this exact format into the hasLowerAlpha method. I changed the if statement to look for lowercase letters this time and to throw NoLowerAlphaException if 0 lowercases.

I moved onto hasDigit method. I copied the same format from hasUpperAlpha and changed the if statement to look for digits and to throw NoDigitException if 0 digits. For hasSpecialChar, I used the template that you gave us in the writeup. I skipped over to hasBetweenSixandNineChars method and used an if statement to check if the password length with greater than 5 and less than 10. For isWeakPassword method, I used an if statement to first check if the password if valid, and then checked if the password was between 6 and 9. If yes, then it throws then WeakPasswordException. I moved on to the getInvalidPassword method. I first created an ArrayList of strings called invalid list to hold the invalid passwords. I used a for loop to run through the given ArrayList of passwords. Using a try and catch, I assigned each password to the String word. Then I checked if the string was a valid password or not. If it's invalid, the catch will be thrown. This will add the password plus the exception message to the invalid ArrayList.

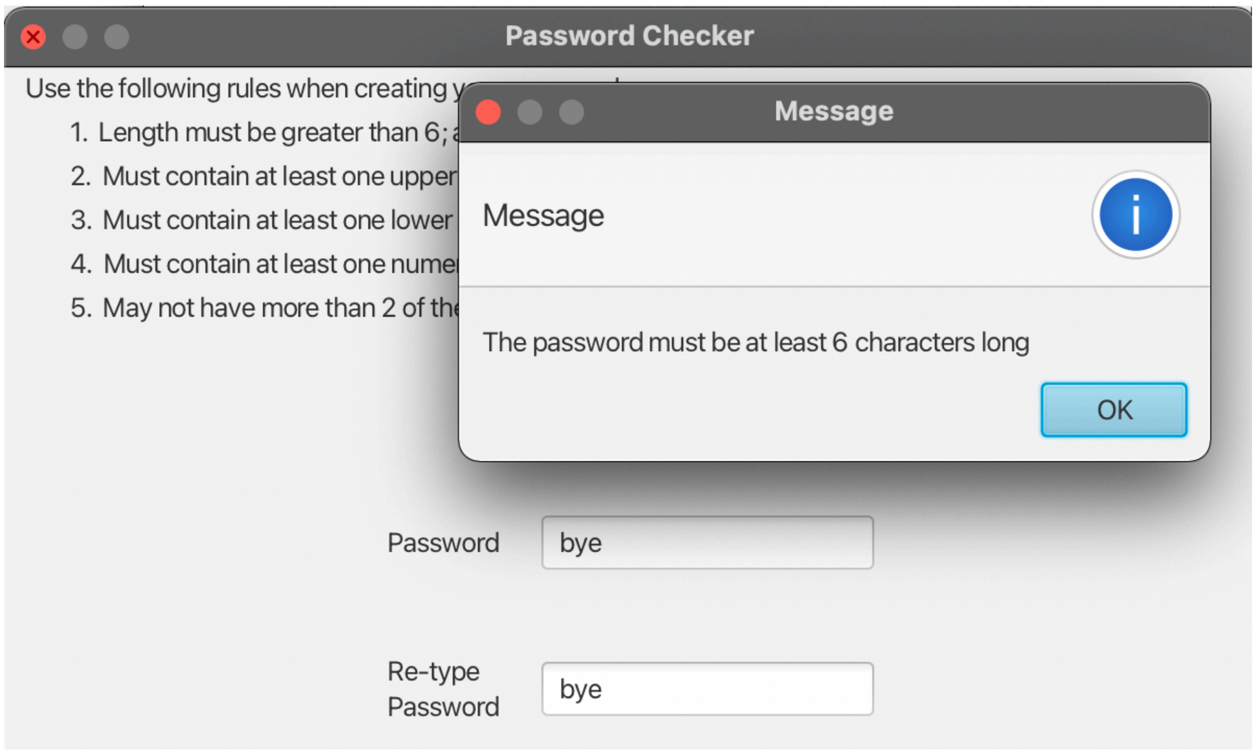
I then circled back to the isValidPassword method. I used an if statement to check is the password was valid length of 6 or more, has a uppercase letter, has a lowercase letter, has a digit, has a special character and no same character three times. I worked on the NoSameCharInSequence method next. I started with a for loop to run through each character of the password. I used an if statement to check the first character is equal to the character next to it. Then I added another if statement to check if the second character is equal to the third character. If it is, then the InvalidSequenceException is thrown. Lastly, I worked on the comparePasswords and comparePasswordsWithReturn methods. I used an if statement to compared if the two password lengths were the same. If they weren't, then UnmatchedException was thrown. Then I used a for loop to run through one of the passwords and to check if each character of both passwords were equal. If they were, then the count increases by one. I used a if statement to check if the count was equal to the length of the password. If it wasn't, then UnmatchedException was thrown. I copied this format into the returns method and tweaked it to have a return statement instead.

I started testing some of the invalid passwords in the write up and seemed to be going pretty smoothly. I was getting the correct exception message. All of this felt pretty straight forward so far. Then I tested a few correct passwords. I kept getting this message "index 8 out of bounds for length 8". I knew the passwords were correct so I wasn't sure what was going on. I kept looking at each method, over and over and over yet kept getting the same error message. I knew the error meant it had something to do with ArrayIndexOutOfBoundsException. But every example I saw online about it, talked about arrays or ArrayLists. So of course, my first genius idea was to stare at the invalidPassword method since it had an ArrayList in it. This didn't do a single thing for me. After an hour or so of struggle, I moved on to testing the Junits. I started with the public Junit. I passed all 5 Junit tests. I then checked the GFA and passed all 4 Junit tests. Then came the final Test with 6 Junits. I passed 5 out of 6 runs. I failed the InvalidSequence test. I looked over the method and couldn't find anything at first. Somewhere, something was going out of bounds. I looked closer at my for loops and everything finally clicked. After hours of struggling, I realized that I needed to take two off the password's length. We only care about if there is three of the same consecutive letters in a row and only need to search this until the third to last character. I tested the GUI again and entered in some valid passwords. I finally got the weak password messages along with the valid password messages.

Test Runs and Test Cases

LengthException

Input	Expected Output	Actual Output
Password - bye **Retype - bye**	The password must be at least 6 characters long.	The password must be at least 6 characters long.



Use the following rules when creating y

1. Length must be greater than 6; a
2. Must contain at least one upper
3. Must contain at least one lower
4. Must contain at least one nume
5. May not have more than 2 of the

Message

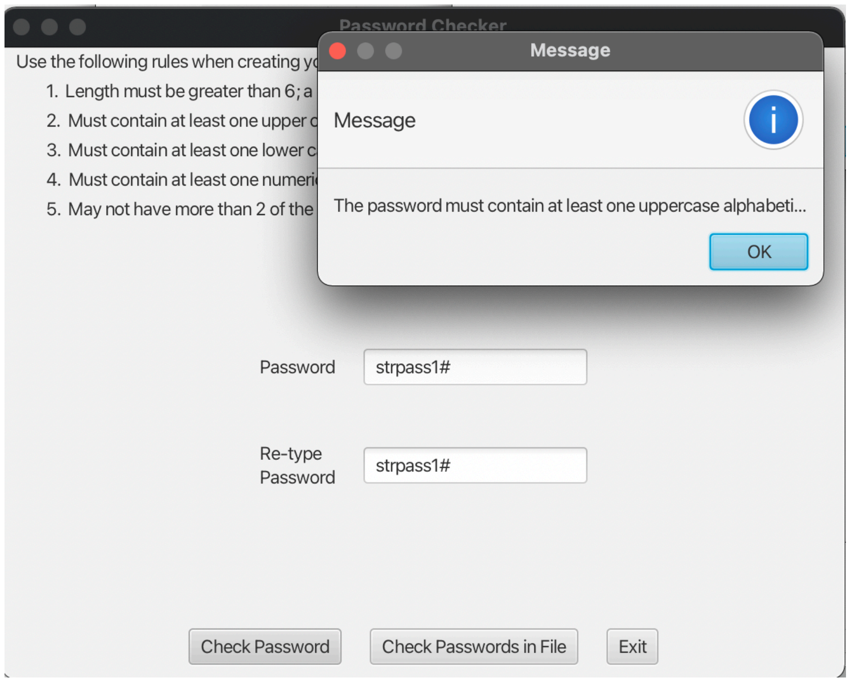
The password must be at least 6 characters long

Password: bye

Re-type Password: bye

NoUpperAlphaException

Input	Expected Output	Actual Output
Password - strpass1# **Retype - strpass1#**	The password must contain at least one uppercase character	The password must contain at least one uppercase character


Password Checker

Use the following rules when creating y

1. Length must be greater than 6; a
2. Must contain at least one upper c
3. Must contain at least one lower c
4. Must contain at least one numeri
5. May not have more than 2 of the

Message

Message

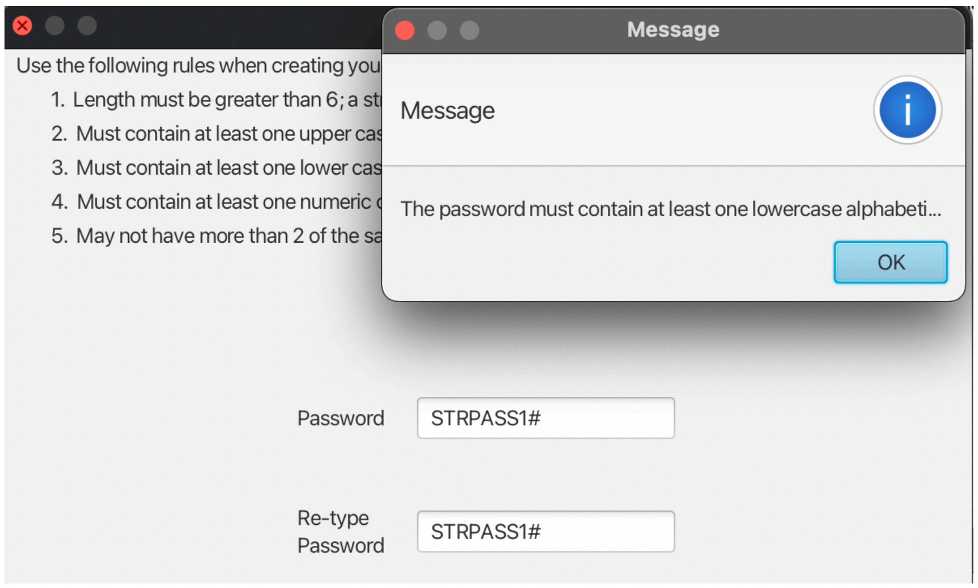
The password must contain at least one uppercase alphabeti...

Password: strpass1#

Re-type Password: strpass1#

NoLowerAlphaException

Input	**Expected Output**	**Actual Output**
Password - STRPASS1# **Retype - STRPASS1#**	The password must contain at least one lowercase character	The password must contain at least one lowercase character


Use the following rules when creating you

1. Length must be greater than 6; a st
2. Must contain at least one upper cas
3. Must contain at least one lower cas
4. Must contain at least one numeric
5. May not have more than 2 of the sa

Message

Message

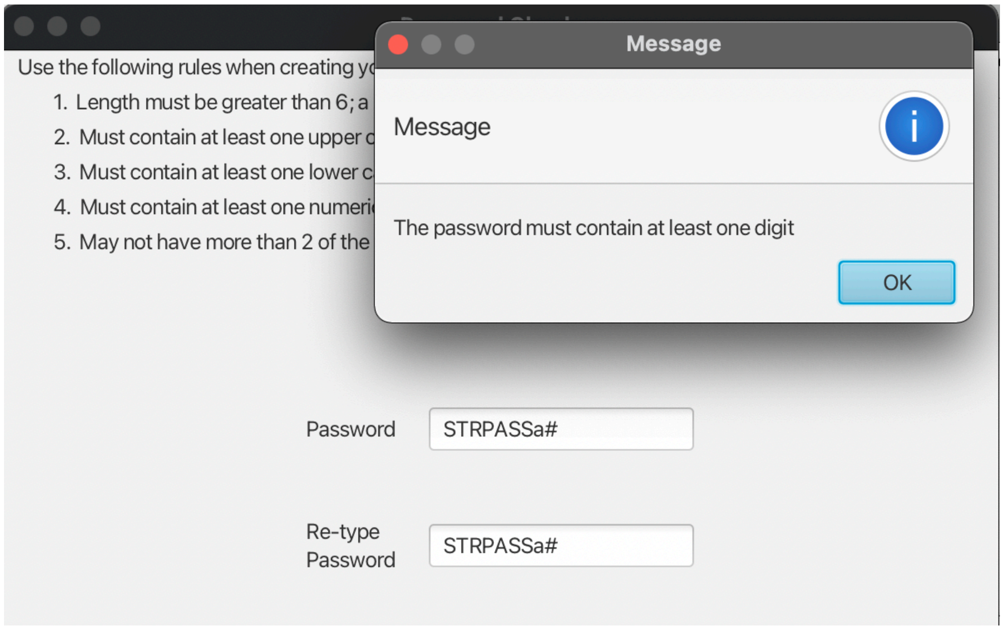
The password must contain at least one lowercase alphabeti...

Password: STRPASS1#

Re-type Password: STRPASS1#

NoDigitException

Input	Expected Output	Actual Output
Password - STRPASSa# **Retype - STRPASSa#**	The password must contain at least one digit.	The password must contain at least one digit.


Use the following rules when creating yo
1. Length must be greater than 6; a
2. Must contain at least one upper c
3. Must contain at least one lower c
4. Must contain at least one numeri
5. May not have more than 2 of the

Message

Message

The password must contain at least one digit

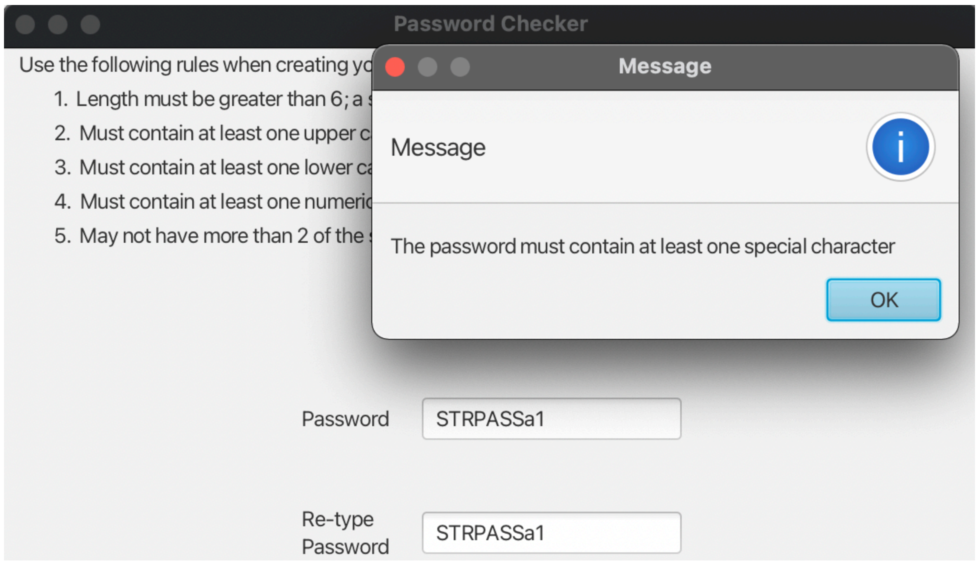
OK

Password STRPASSa#

Re-type Password STRPASSa#

NoSpecialCharacterException

Input	Expected Output	Actual Output
Password - STRPASSa1 **Retype - STRPASSa1**	The password must contain at least one special character.	The password must contain at least one special character.


Password Checker

Use the following rules when creating yo
1. Length must be greater than 6; a
2. Must contain at least one upper c
3. Must contain at least one lower ca
4. Must contain at least one numeric
5. May not have more than 2 of the

Message

Message

The password must contain at least one special character

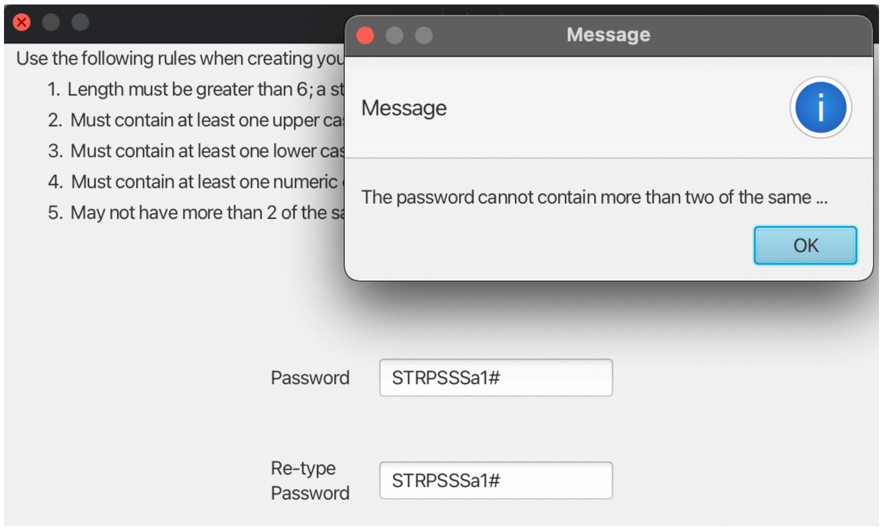
OK

Password STRPASSa1

Re-type Password STRPASSa1

InvalidSequenceException

Input	Expected Output	Actual Output
Password - STRPSSSa1# **Retype - STRPSSSa1#**	The password cannot contain more than two of the same character in sequence.	The password cannot contain more than two of the same character in sequence.


Use the following rules when creating you
1. Length must be greater than 6; a st
2. Must contain at least one upper ca
3. Must contain at least one lower cas
4. Must contain at least one numeric
5. May not have more than 2 of the sa

Message

Message

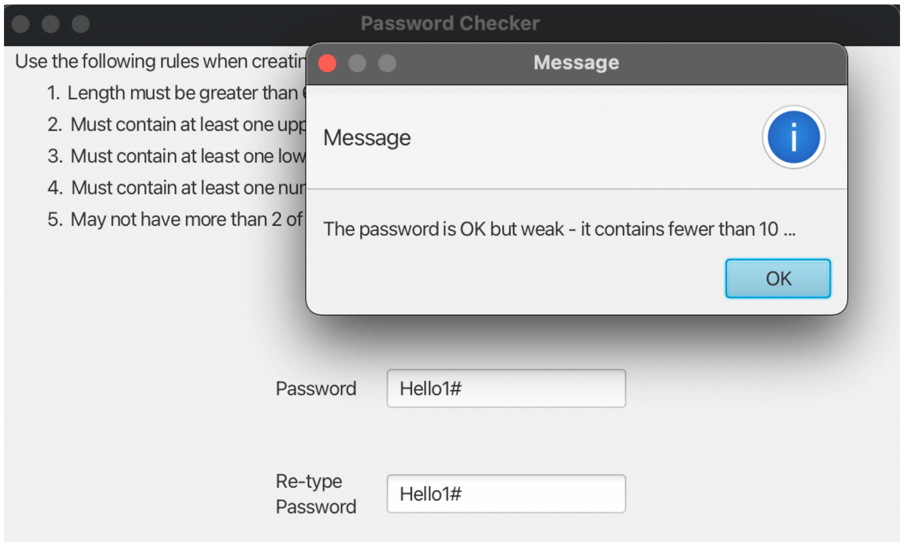
The password cannot contain more than two of the same ...

Password STRPSSSa1#

Re-type Password STRPSSSa1#

WeakPasswordException

Input	Expected Output	Actual Output
Password - Hello1# **Retype - Hello1#**	The password is OK but weak - it contains fewer than 10 characters.	The password is OK but weak - it contains fewer than 10 characters.


Password Checker

Use the following rules when creatin
1. Length must be greater than
2. Must contain at least one upp
3. Must contain at least one low
4. Must contain at least one nur
5. May not have more than 2 of

Message

Message

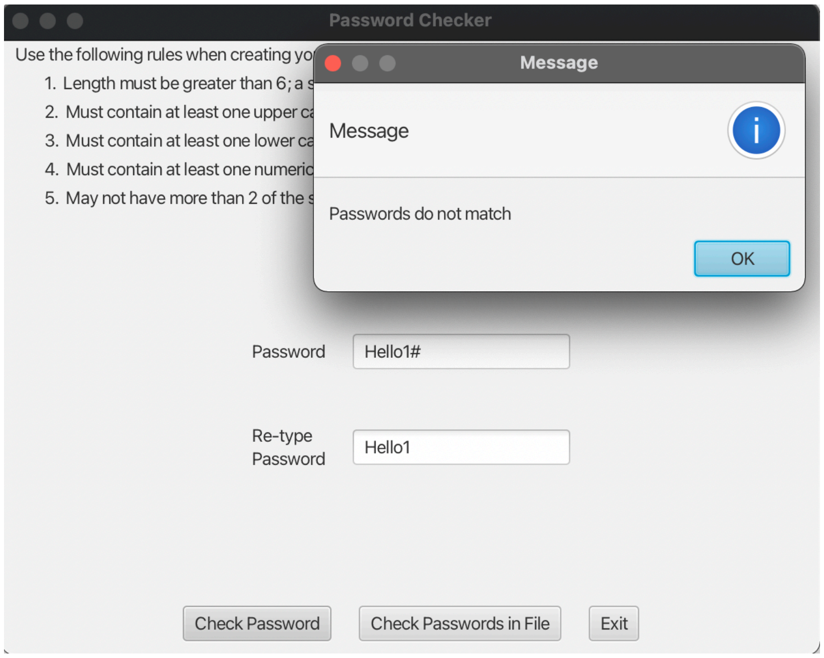
The password is OK but weak - it contains fewer than 10 ...

Password Hello1#

Re-type Password Hello1#

UnmatchedException

Input	Expected Output	Actual Output
Password - Hello1# **Retype - Hello1**	Password do not match	Password do not match



Use the following rules when creating yo
1. Length must be greater than 6; a s
2. Must contain at least one upper ca
3. Must contain at least one lower ca
4. Must contain at least one numeric
5. May not have more than 2 of the s

Message

Passwords do not match

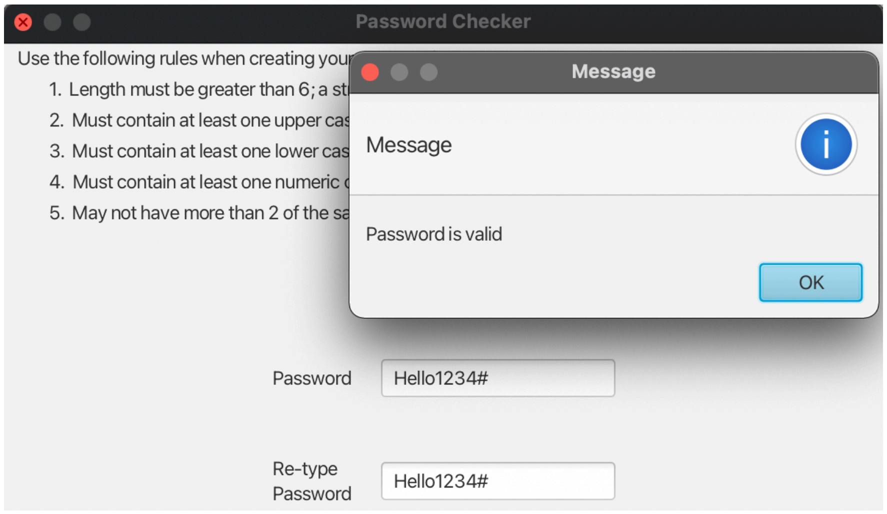
OK

Password Hello1#

Re-type Password Hello1

Valid Password

Input	Expected Output	Actual Output
Password - Hello1234# **Retype - Hello1234#**	Password is valid	Password is valid



Use the following rules when creating your
1. Length must be greater than 6; a st
2. Must contain at least one upper cas
3. Must contain at least one lower cas
4. Must contain at least one numeric c
5. May not have more than 2 of the sa

Message

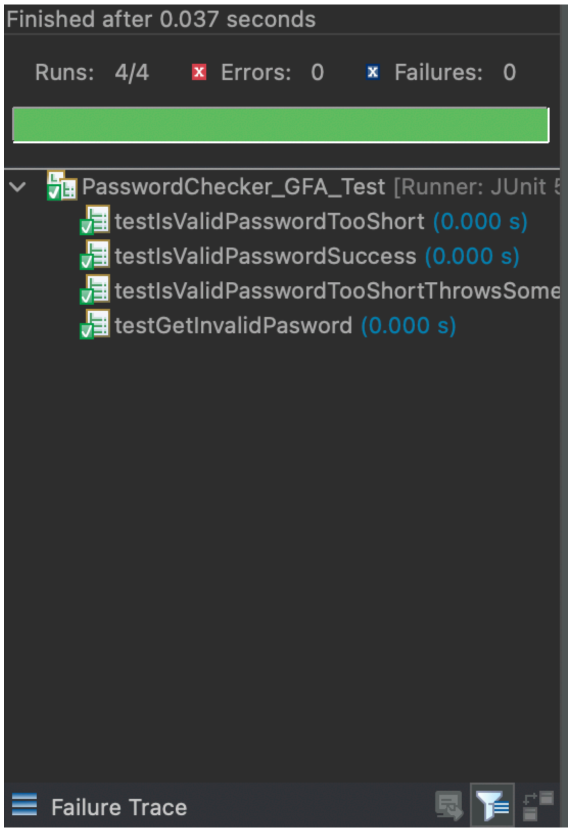
Password is valid

OK

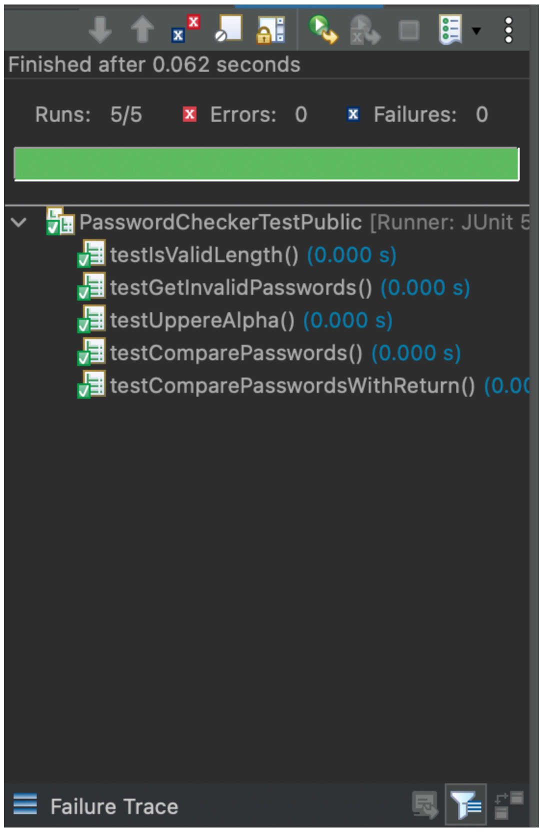
Password Hello1234#

Re-type Password Hello1234#

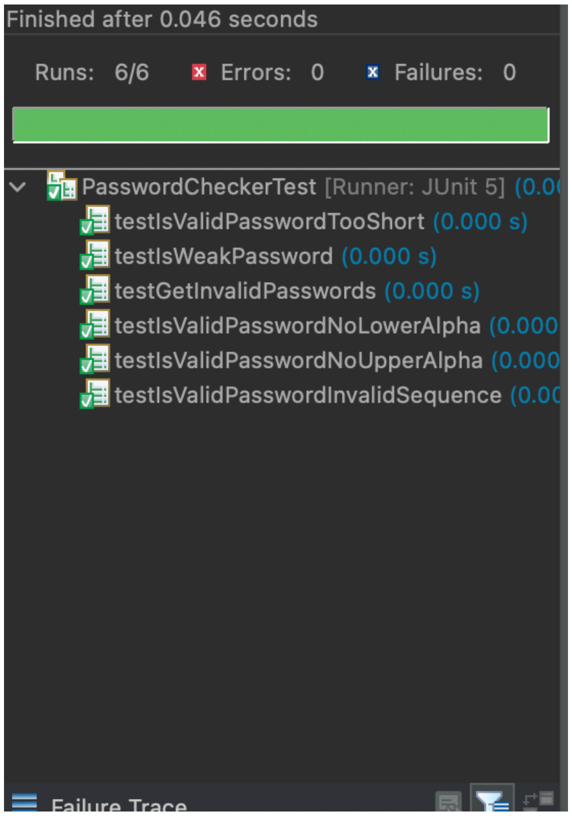
Junit tests



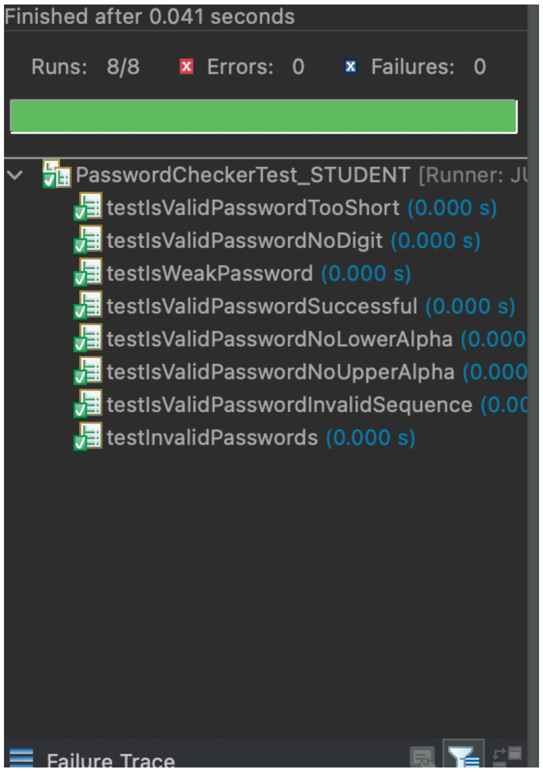
```
Finished after 0.037 seconds
Runs: 4/4   Errors: 0   Failures: 0
PasswordChecker_GFA_Test [Runner: JUnit 5
    testIsValidPasswordTooShort (0.000 s)
    testIsValidPasswordSuccess (0.000 s)
    testIsValidPasswordTooShortThrowsSome
    testGetInvalidPasword (0.000 s)
Failure Trace
```



```
Finished after 0.062 seconds
Runs: 5/5   Errors: 0   Failures: 0
PasswordCheckerTestPublic [Runner: JUnit 5
    testIsValidLength() (0.000 s)
    testGetInvalidPasswords() (0.000 s)
    testUppereAlpha() (0.000 s)
    testComparePasswords() (0.000 s)
    testComparePasswordsWithReturn() (0.00
Failure Trace
```



```
Finished after 0.046 seconds
Runs: 6/6   Errors: 0   Failures: 0
PasswordCheckerTest [Runner: JUnit 5] (0.00
    testIsValidPasswordTooShort (0.000 s)
    testIsWeakPassword (0.000 s)
    testGetInvalidPasswords (0.000 s)
    testIsValidPasswordNoLowerAlpha (0.000
    testIsValidPasswordNoUpperAlpha (0.000
    testIsValidPasswordInvalidSequence (0.00
Failure Trace
```



```
Finished after 0.041 seconds
Runs: 8/8   Errors: 0   Failures: 0
PasswordCheckerTest_STUDENT [Runner: JU
    testIsValidPasswordTooShort (0.000 s)
    testIsValidPasswordNoDigit (0.000 s)
    testIsWeakPassword (0.000 s)
    testIsValidPasswordSuccessful (0.000 s)
    testIsValidPasswordNoLowerAlpha (0.000
    testIsValidPasswordNoUpperAlpha (0.000
    testIsValidPasswordInvalidSequence (0.00
    testInvalidPasswords (0.000 s)
Failure Trace
```

Learning Experiences

From this project, I have learned about regex expressions and always remembering to review the for loops. I have never seen or heard of them before but it was interesting to learn how they would be used to help us for our project. It seems to be a pretty efficient method of checking for special characters in a string.

From now on, I will make it a priority to look at my for loops if there is an issue with the Junits etc. I feel it would be smart to go there first and rule out anything. It would save a lot of headaches in the future.

Overall, I felt pretty confident with this assignment. It would good practice with exceptions along with Junit testing. It was nice to have a template to go off of and make sure they we were on the right track. I will definitely save the information for special character searching.

Assumptions

- User will be using Junit 5 and has JavaFX
- All weak passwords are also valid
- Assumes user isn't using special character that are unreadable
- I am assuming that we don't have to worry about the txt file. There was no file in the zip file. I tested a test.txt file and kept getting the spinning wheel of death. I think there might be an issue with the GUI code but I'm not 100%.

Enhancements

The only thing I did notice was that when I add isValidPassword to if statement in the isWeakPassword method, it kept having a red line under it. It kept saying to add the throws from that method but on our Javadoc, it didn't mention that. I went ahead and added those throws to make the method work. The only other way was to delete the isValidPassword from the if statement but then I wouldn't be checking it correctly.