CMSC204

Assignment 1

Spring 2016

Concepts tested by this program:

ArrayList

Tooltips

Mnemonic

Read Files

Javadoc

JUnit Tests

Exceptions

Create an application that will check for valid passwords. The following rules must be followed to create a valid password.

1. At least 8 characters long
2. At least 1 numeric character
3. At least 1 uppercase alphabetic character
4. At least 1 lowercase alphabetic character
5. No more than 2 of the same character in a sequence

Hello123 – OK

AAAbb123 – not OK

Aaabb123 – OK

**Operation:**

When the application begins, the user will be presented with a screen that states the above instructions for creating a password, two text entry boxes for typing in a password, and three buttons.

If the user wants to check a single password, they will type in the password in both boxes and select the “Check Password” button.

If the user wants to read in and check a list of passwords, they will select the “Check Passwords in File” button, be presented with a file explorer, and select the file to read from. Those passwords that failed the check will be displayed, with their error message.

If the user presses the “Alt” key, a letter will be underlined in each button label. That letter is the “mnemonic” that can used as a shortcut(Alt plus the letter) to execute the button.

If the user hovers his cursor over a button, a tooltip will be shown.

**Specifications**:

**The Data Element**

String

**The Data Structure**

ArrayList of Strings

**The Data Manager**

Create a PasswordChecker class that implements PasswordCheckerInterface. The PasswordChecker class will have at least two methods: One method that checks the validity of one password that returns true if the password is valid and that throws an exception if invalid. One that checks an ArrayList of passwords and returns an ArrayList with the status of any invalid passwords. The ArrayList of invalid passwords will be of the following format:

<password><space><message of exception thrown>

Create exception classes for each exception listed in PasswordCheckerInterface.java.

Always check for the length of the password first, since that is the easiest and fastest check. Once the password fails one rule, you do not need to check the rest of the rules.

**The GUI**

* Provide buttons to allow user to check validity of one password or a file of passwords.
* Ask the user to enter the password and to re-type the password. If the two are not the same, inform the user.
* Create a tool tip and a mnemonic for each of the buttons.
* Use a FileChooser for the user to select the input file.
* Use methods of PasswordChecker to process the passwords.
* Use try/catch structure to catch exceptions thrown by PasswordChecker methods

**Exceptions**

Provide exception classes for the following:

1. Length of password is less than 8 characters (class LengthException)

Message – The password must be at least 8 characters long

1. Password doesn’t contain an uppercase alpha character (class NoUpperAlphaException)

Message – The password must contain at least one uppercase alphabetic character

1. Password doesn’t contain a lowercase alpha character (class NoLowerAlphaException)

Message – The password must contain at least one lowercase alphabetic character

1. Password doesn’t contain a numeric character (class NoDigitException)

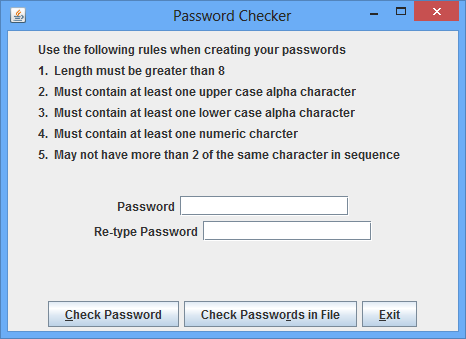
Message – The password must contain at least one digit

1. Password contains more than 2 of the same character in sequence (class InvalidSequenceException)

Message – The password cannot contain more than two of the same character in sequence.

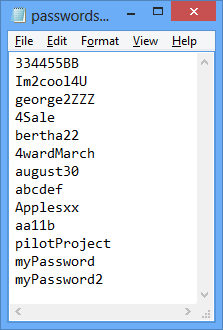
1. For GUI – check if Password and re-typed Password are identical (class UnmatchedExcpetion)

Message – The passwords do not match



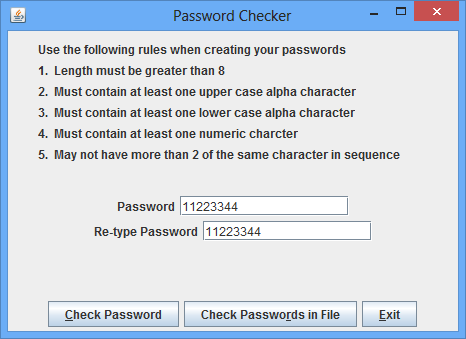
* The file will be in the following format:

One password per line

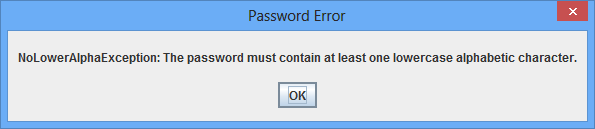


Examples:

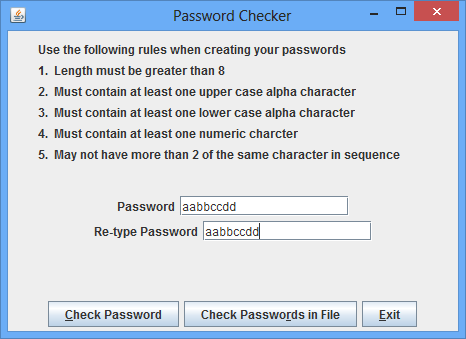
1. Note: If you check for the uppercase before the lowercase – you will receive the uppercase exception



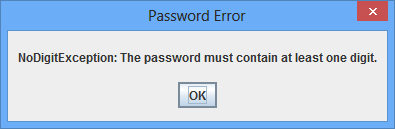
Displayed to user:



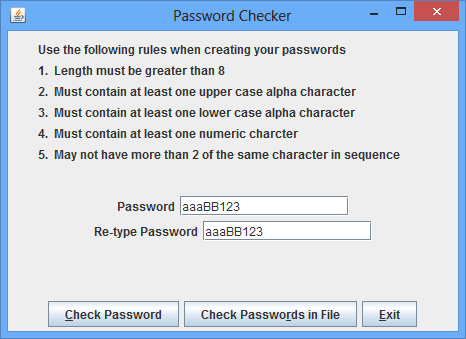
1. Note: If you check for the uppercase before the digit – you will receive the uppercase exception



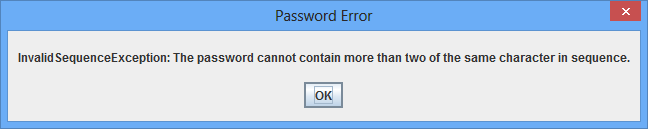
Displayed to user:



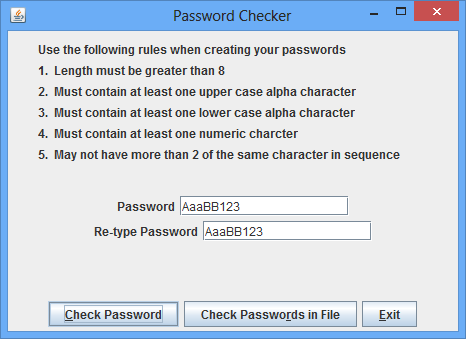
3.



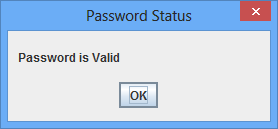
Displayed to user:



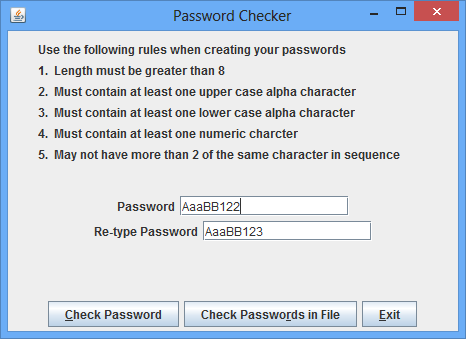
4.



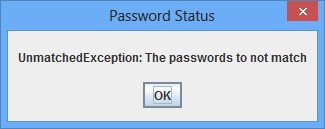
Displayed to user:



5.



Displayed to user:



6. Based on the file above:

Displayed to user when selects Check Passwords in File



**Assignment 1 Grade Sheet – Spring 2016**

Name \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

## DOCUMENTATION 20 points

# Javadoc for the user defined classes 6 pt \_\_\_\_\_

Test Cases 10 pt \_\_\_\_\_

JUnit Test Class

Implement STUDENT methods of the

PasswordCheckerTest class

UML Diagram 4 pt \_\_\_\_\_

## PROGRAMMING 80 points

Internal class documentation (within source code) 16 pts \_\_\_\_\_

Class description using Javadoc

Author’s Name, Class, Class Time, @author

Methods commented using Javadoc, @param, @return

Additional comments as needed

Program user interface 6 pts \_\_\_\_\_

Clear to user how data is to be entered

Output is easy to understand

Accuracy

Received correct output

JUnit tests succeed 22 pts \_\_\_\_\_

Program Detail

1. PasswordChecker class 12 pts \_\_\_\_\_
   1. Method to check validity of password
   2. Method to check validity of ArrayList of passwords
   3. Implements PasswordCheckerInterface
2. GUI classes 12 pts \_\_\_\_\_
   1. Uses methods of PasswordChecker
   2. Displays status of password entered by user
   3. Buttons have tooltips and mnemonics
   4. Reads from a file and displays illegal passwords
   5. Stores contents of file in ArrayList
   6. check if user entered password and

Retyped are not the same

1. Exceptions classes 12 pts \_\_\_\_\_
   1. Class for each invalid password rule
   2. Class if password and re-type password don’t match

Total 100 pts \_\_\_\_\_