CMSC204

Assignment 1

Passwords

Concepts tested by this program:

ArrayList

Tooltips

Mnemonic

Read Files

Javadoc

JUnit Tests

Exceptions

GitHub

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

1. At least 6 characters long
2. 10 or more characters is a strong password, between 6 and 9 characters is a weak (but acceptable) password.
3. At least 1 numeric character
4. At least 1 uppercase alphabetic character
5. At least 1 lowercase alphabetic character
6. 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

**Utility Class**

Create a PasswordCheckerUtility class based on the Javadoc given you. The PasswordCheckerUtility class will have at least three methods: One method will check the validity of one password and return true if the password is valid and throw an exception if invalid.

One method will check for a “weak password”, i.e., one whose length is between 6 and 9, inclusive. Do NOT throw an exception.

One method will check an ArrayList of passwords and return an ArrayList with the status of any invalid passwords (weak passwords are not considered invalid). The ArrayList of invalid passwords will be of the following format:

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

The methods will have the following headers:

static boolean isValidPassword(String pwdString);

static boolean isWeakPassword(String pwdString);

static ArrayList<String> invalidPasswords(ArrayList<String> passwords);

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 (provided)**

* Buttons are provided to allow user to check validity of one password or a file of passwords.
* The user may be asked to enter the password and to re-type the password. If the two are not the same, the user is informed.
* A tool tip and a mnemonic are available for each of the buttons.
* A FileChooser asks the user to select the input file.
* Methods of PasswordCheckerUtility are used to process the passwords.
* Try/catch structures catch exceptions thrown by PasswordCheckerUtility methods

**Exceptions**

Create exception classes for each exception listed in PasswordCheckerUtility Javadoc and listed below:

1. Length of password is less than 6 characters (class LengthException)

Message – The password must be at least 6 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 UnmatchedException)

Message – The passwords do not match

Throw this exception from the GUI, not the utility class.

A screenshot of a cell phone

Description automatically generated

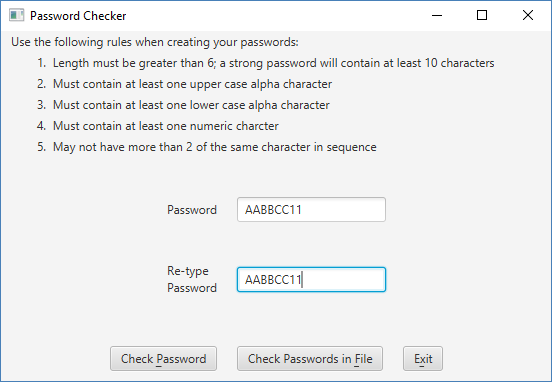
If more than one error is present in a password, use the above order to throw exceptions. For example, if a password is “xxyyzzwwaa”, it fails rules 2 and 4 above. Throw a NoUpperAlphaException, not a NoDigitException.

* The file will be in the following format (one password per line):

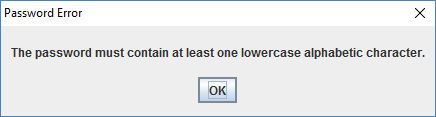


Examples:

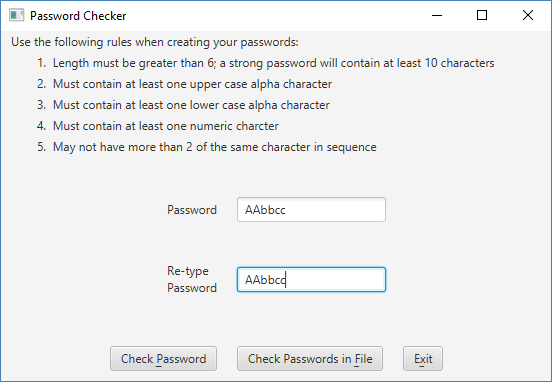
1. No lowercase alphabetic character



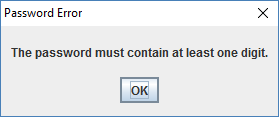
Displayed to user:



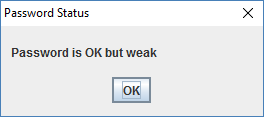
1. No digit



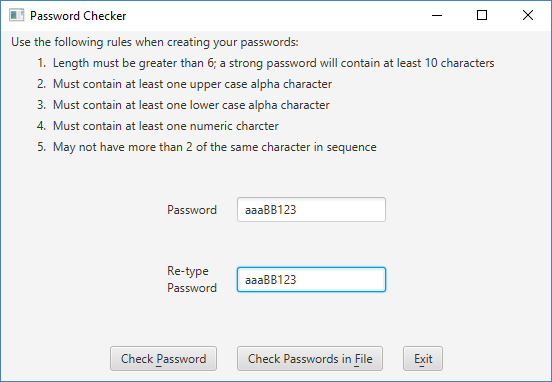
Displayed to user:



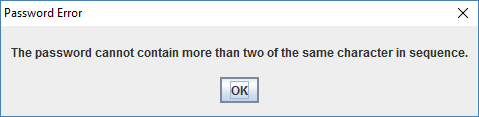
1. If the password is OK, but between 6 and 10 characters, you will see:



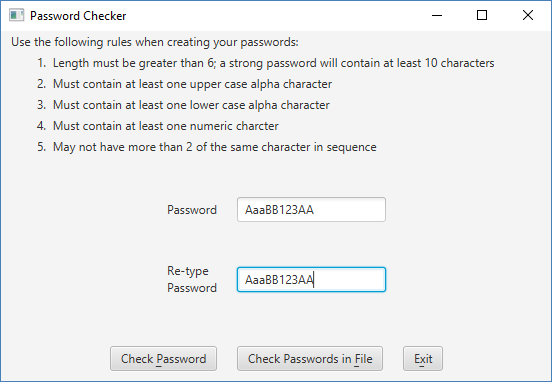
1. If password has more than two of the same characters in a row



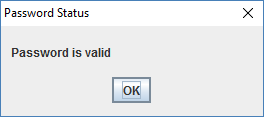
Displayed to user:



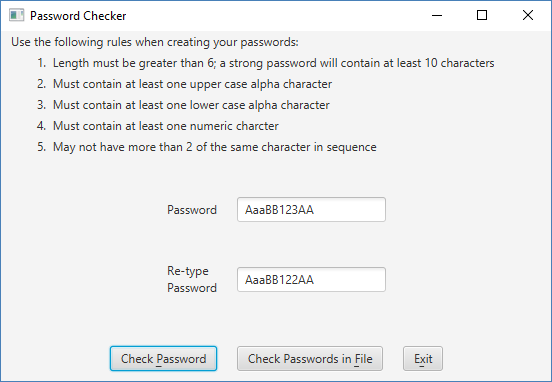
1. If password is valid



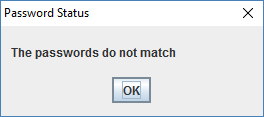
Displayed to user:



1. If the passwords do not match:

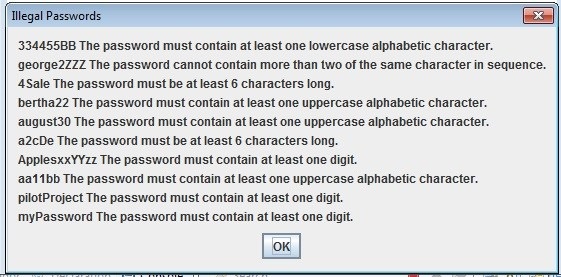


Displayed to user:



6. Based on the file above, when the user selects “Check Passwords in File”:

Displays errors to user when selecting Check Passwords in File. Note that valid passwords are NOT displayed.



Upload the initial files to the GitHub repo you created in Lab 1 in a directory named Assignment1\_Passwords. Upload the final files as well. Then take a screen shot of the repo with the directory and files.

Deliverables:

Design:

Initial design document

Implementation:

Final design document

Java files - The src folder with your driver (javafx application), data manager, exceptions and Junit Test (.java) files

Learning Experience document

Screen shot of the repo in GitHub with the initial and final files uploaded.

Deliverable format: The above deliverables will be packaged as follows. Two compressed files in the following formats:

LastNameFirstName\_AssignmentX.zip [compressed file containing following]:

doc [a directory] *include the entire doc folder with the javadoc for*

*student generated files*

file1.html (example)

file2.html (example)

class-use (example directory)

LearningExperience.doc *or other text format*

Screen shot of the GitHub repo.

src [a directory] *contains your driver (javafx application), data manager, exceptions and Junit Test (.java) files*

File1.java (example)

File2.java (example)

File\_Test.java (example)

LastNameFirstName\_AssignmentX\_**Moss**.zip [compressed file containing only]:

*.java file which includes the driver (javafx application), data manager, exceptions and Junit Test (.java) files – NO*

*FOLDERS!!*

File1.java (example)

File2.java (example)