CMSC204 – Prof. Gary Thai

Project 1 – PasswordCheckerUtility

**Psuedocode**

**Class PasswordCheckerUtility:**

* Method comparePasswords
  + If passwords do not match, throw an Unmatched Exception
  + Check if passwords match
    - If they do not, throw the Unmatched Exception
* Method comparePasswordsWithReturn
  + Check if passwords match
    - If they do, return true
    - If not, return false
* Method getInvalidPasswords
  + Create an ArrayList to hold invalid passwords
  + Read the ArrayList passed as a parameter using a for loop
    - For each password, call isValidPassword()
    - If not, add password to invalid passwords ArrayList
  + Return invalid passwords ArrayList
* Method hasBetweenSixAndNineChars
  + Find out how long the password is
  + Check if the length of the password is between 6 and 9 characters
    - If so, return true
    - If not, return false
* Method hasDigit
  + Look for any digits in the password
    - Make password into char array
    - Use isDigit() method of Character class
  + If there are any, return true
  + If not, throw NoDigitException
* Method hasLowerAlpha
  + Look for any lowercase letters in the password
    - Make password into char array
    - Use isLowerCase() method of Character class
  + If there are any, return true
  + If not, return throw NoLowerAlphaException
* Method hasSameCharInSequence
  + Check if there are more than 2 characters in sequence
    - Convert password into char array
    - Compare characters that are next to each other and keep counter for when characters repeat consecutively
    - If counter reaches more than 2, return true
      * If not, return false and throw InvalidSequenceException
* Method hasSpecialChar
  + Check password if there are any special characters
    - If so, return true
    - If not, throw NoSpecialCharacterException
* Method hasUpperAlpha
  + Look for any uppercase letters in the password
    - Make password into char array
    - Use isUpperCase() method of Character class
  + If there are any, return true
  + If not, throw NoUpperAlphaException
* Method isValidLength
  + Check length of password
  + If password is at least 6 characters long, return true
    - If not, throw LengthException
* Method isValidPassword
  + Call isValidLength()
  + Call hasUpperAlpha()
  + Call hasLowerAlpha()
  + Call hasDigit()
  + Call hasSpecialChar()
  + Call hasSameCharInSequence()
  + If any exceptions are thrown, return false
    - If not, return true
* Method isWeakPassword
  + Get length of password
  + Check if password is between 6-9 characters
    - If so, return true
    - If not, throw WeakPasswordException

**Class UnmatchedException:**

* Extends class Exception
* Constructor UnmatchedException
  + Error message: “The passwords do not match”
* Constructor UnmatchedException, parameters: String message
  + Error message: String sent as parameter

**Class NoDigitException:**

* Extends class Exception
* Constructor NoDigitException
  + Error message: “The password must contain at least one digit”
* Constructor NoDigitException, parameters: String message
  + Error message: String sent as parameter

**Class NoLowerAlphaException:**

* Extends class Exception
* Constructor NoLowerAlphaException
  + Error message: “The password must contain at least one lowercase alphabetic character”
* Constructor NoLowerAlphaException, parameters: String message
  + Error message: String sent as parameter

**Class InvalidSequenceException:**

* Extends class Exception
* Constructor InvalidSequenceException
  + Error message: “The password cannot contain more than two of the same character in sequence”
* Constructor InvalidSequenceException, parameters: String message
  + Error message: String sent as parameter

**Class NoSpecialCharacterException:**

* Extends class Exception
* Constructor NoSpecialCharacterException
* Error message: “The password must contain at least one special character”
* Constructor NoSpecialCharacterException, parameters: String message
  + Error message: String sent as parameter

**Class NoUpperAlphaException:**

* Extends class Exception
* Constructor NoUpperAlphaException
  + Error message: “The password must contain at least one uppercase alphabetic character”
* Constructor NoUpperAlphaException, parameters: String message
  + Error message: String sent as parameter

**Class LengthException:**

* Extends class Exception
* Constructor LengthException
  + Error message: “The password must be at least 6 characters long”
* Constructor LengthException, parameters: String message
  + Error message: String sent as parameter

**Class WeakPasswordException:**

* Extends class Exception
* Constructor WeakPasswordException
  + Error message: “The password is OK but weak - it contains fewer than 10 characters”
* Constructor WeakPasswordException, parameters: String message
  + Error message: String sent as parameter

**Lessons Learned**

This project was centered around building a utility class and multiple custom exception classes. The utility class had several methods to determine if a password was valid by considering several criteria including the inclusion of certain characters and length. The GUI allowed for an easy way to interact with the program and the JUnit tests provided and created allowed for testing of different methods of the utility class and the exception classes.

While I made the utility class, I learned several new ways to work with String values. I used .toCharArray to turn a String into a char array that could be iterated through in order to check for different criteria. I also learned to implement different methods of the Character class including isUpperCase() and isLowerCase(). Additionally, I learned about Pattern and Matcher which allowed me to search for special characters.

One of the main issues I struggled with was setting up the project to work with JavaFX. I learned that in order to get JavaFX to work with my IDE I needed to download the relevant files, build a path, and adjust the run configurations. I also had some logical issues in between that I recognized through testing. For example, when my isWeakPassword() method was used, it would return true instead of throwing a isWeakPassword() exception if the password was between 6 and 9 characters.

Overall, I enjoyed working on this project and it made me more confident in JUnit testing, managing Strings, and creating and handling exceptions.

**Assumptions**

* All characters used are within the English language
* The UnmatchedException was supposed to be thrown in the utility class, not the GUI
* The spelling for exception messages was correct in the project description word document (one test given had a misspelling with lower case instead of lowercase)
* Exceptions were supposed to be thrown in the order stated. The program did not need to throw multiple exceptions for the same password.

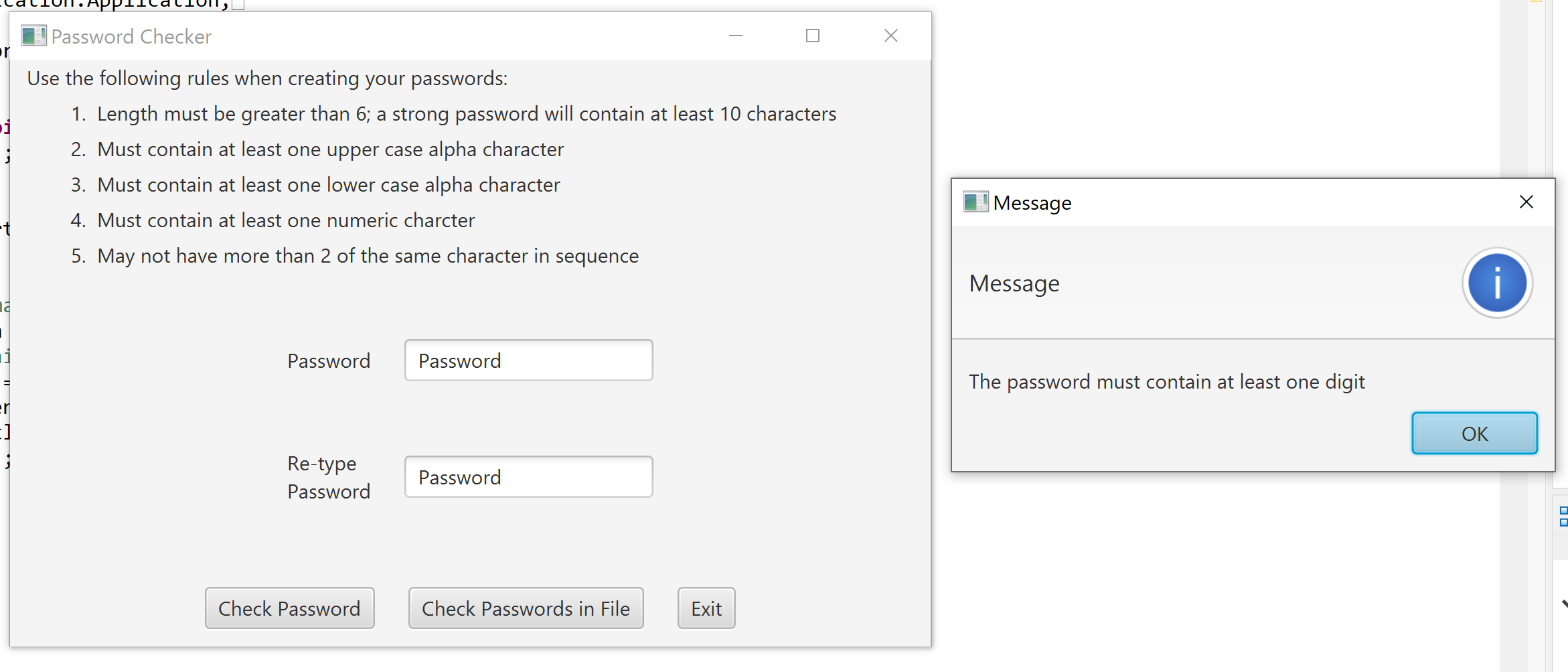
**Test Plan**

The following tests were created to test different scenarios with passwords that should pass and passwords that should not pass with their correct exception printed.

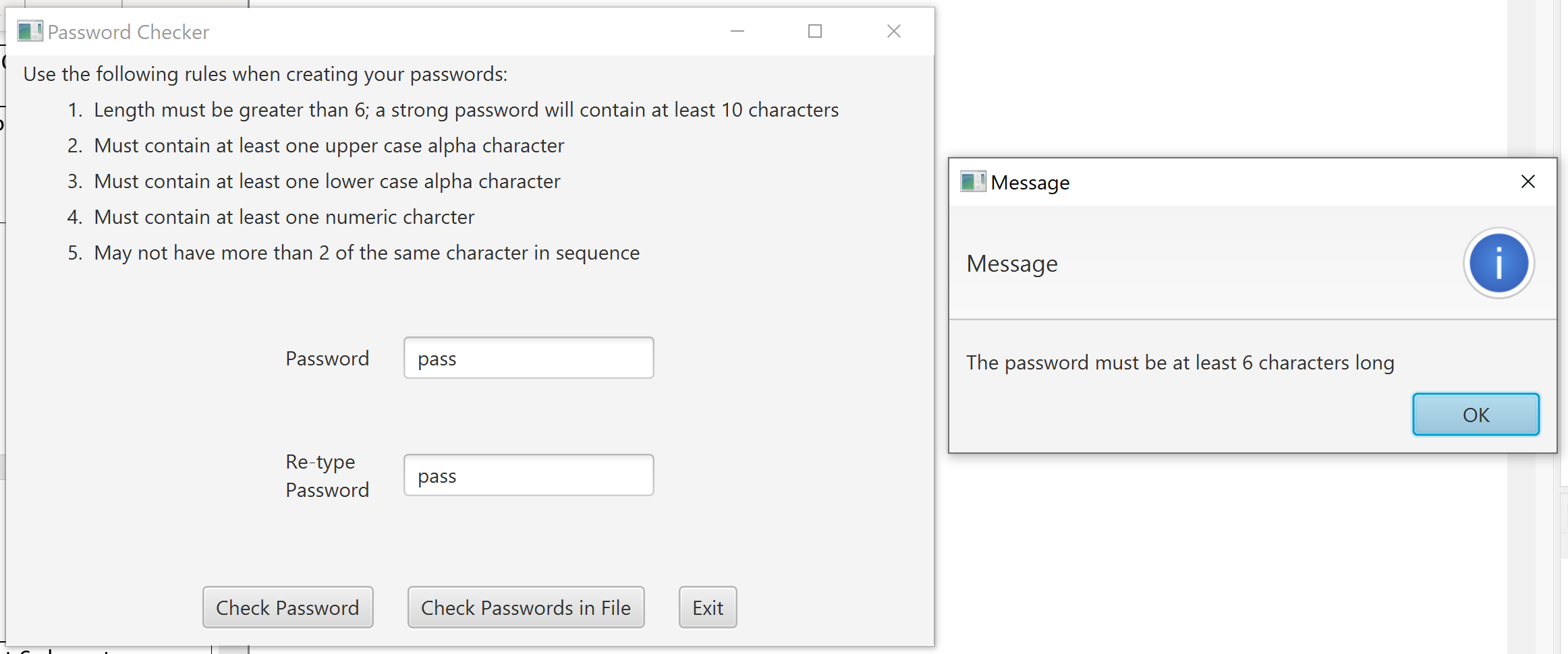
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Test # | Input | Expected Output | Actual Output | Pass? |
| 1 | Entry: Password Re-entry: Password | Fail: no digit | Fail: no digit | Y |
| 2 | Entry: pass  Re-entry: pass | Fail: not 6 characters | Fail: not 6 characters | Y |
| 3 | Entry: password Re-entry: password | Fail: no uppercase letter | Fail: no uppercase letter | Y |
| 4 | Entry: PASSWORD Re-entry: PASSWORD | Fail: no lowercase letter | Fail: no lowercase letter | Y |
| 5 | Entry: Password9 Re-entry: Password9 | Fail: no special character | Fail: no special character | Y |
| 6 | Entry: Passsword9!  Re-entry: Passsword9! | Fail: too many characters in sequence | Fail: too many characters in sequence | Y |
| 7 | Entry: Apple9!  Re-entry: Apple9! | Pass: Weak password | Pass: Weak password | Y |
| 8 | Entry: pass  Re-entry: pinecone | Fail: passwords must match | Fail: passwords must match | Y |
| 9 | Entry: Abc!123456  Re-entry: Abc!123456 | Pass: valid password | Pass: valid password | Y |
| 10 | Read File: myTest.txt Note: file contains all valid passwords | Pass: No invalid passwords found | Pass: No invalid passwords found | Y |
| 11 | Read File: myTest2.txt Note: file contains one invalid passwords and two valid passwords | Pop up window with invalid password “pass” and exception that the password is too short | Pop up window with invalid password “pass” and exception that the password is too short | Y |

**Test Screenshots**

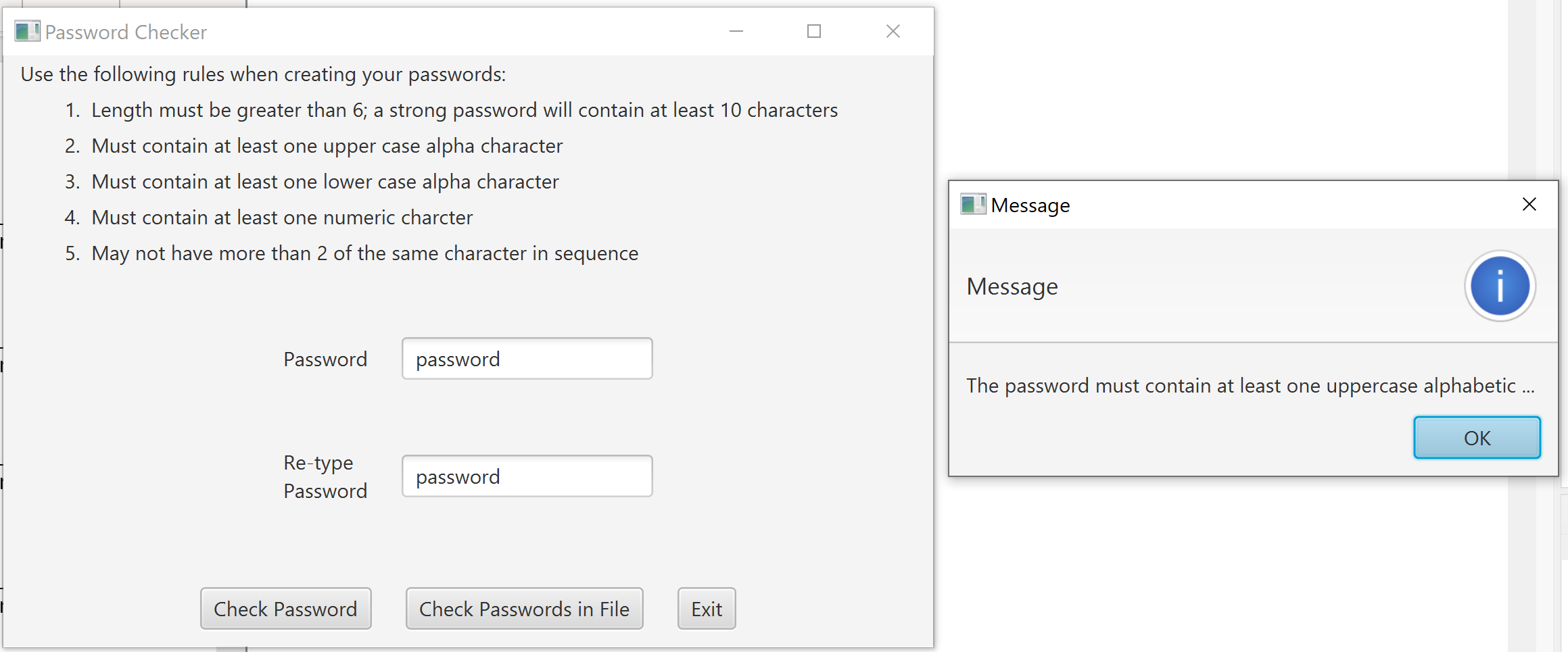
Test 1



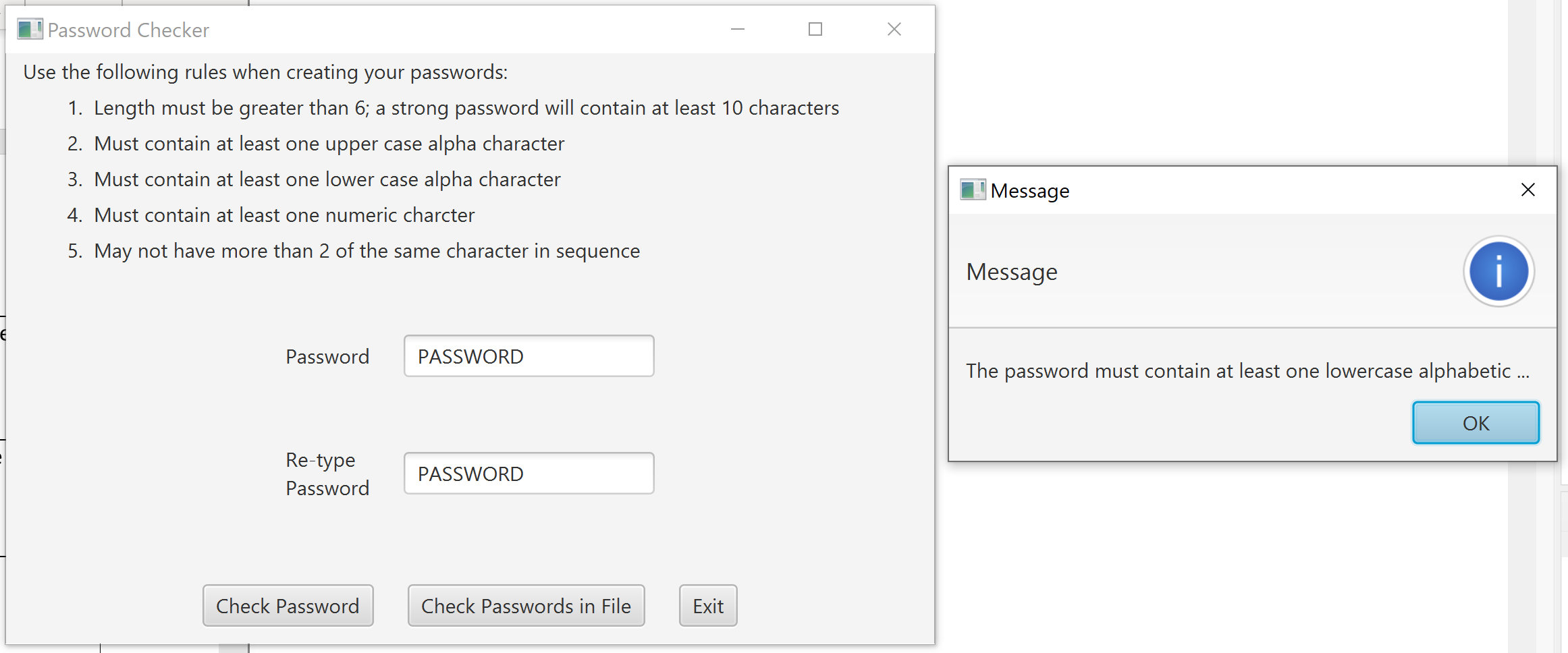
Test 2



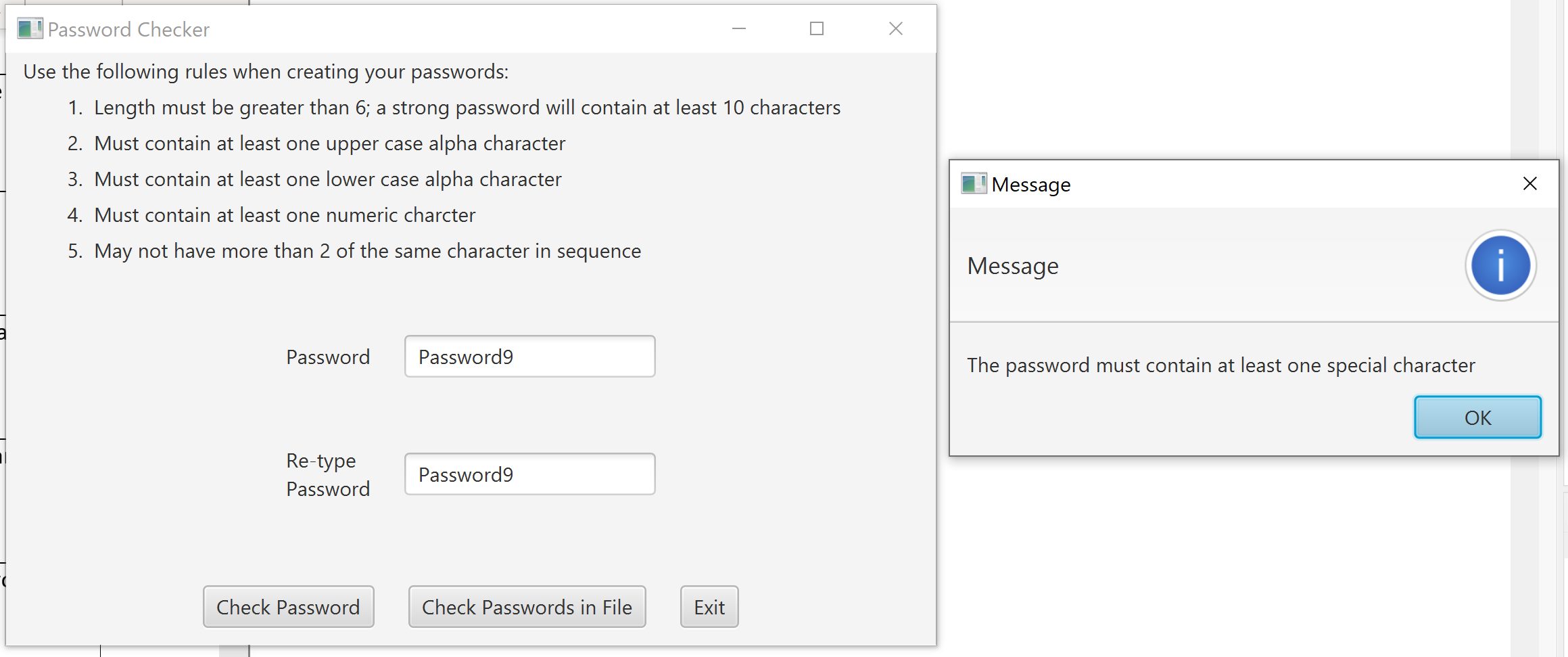
Test 3



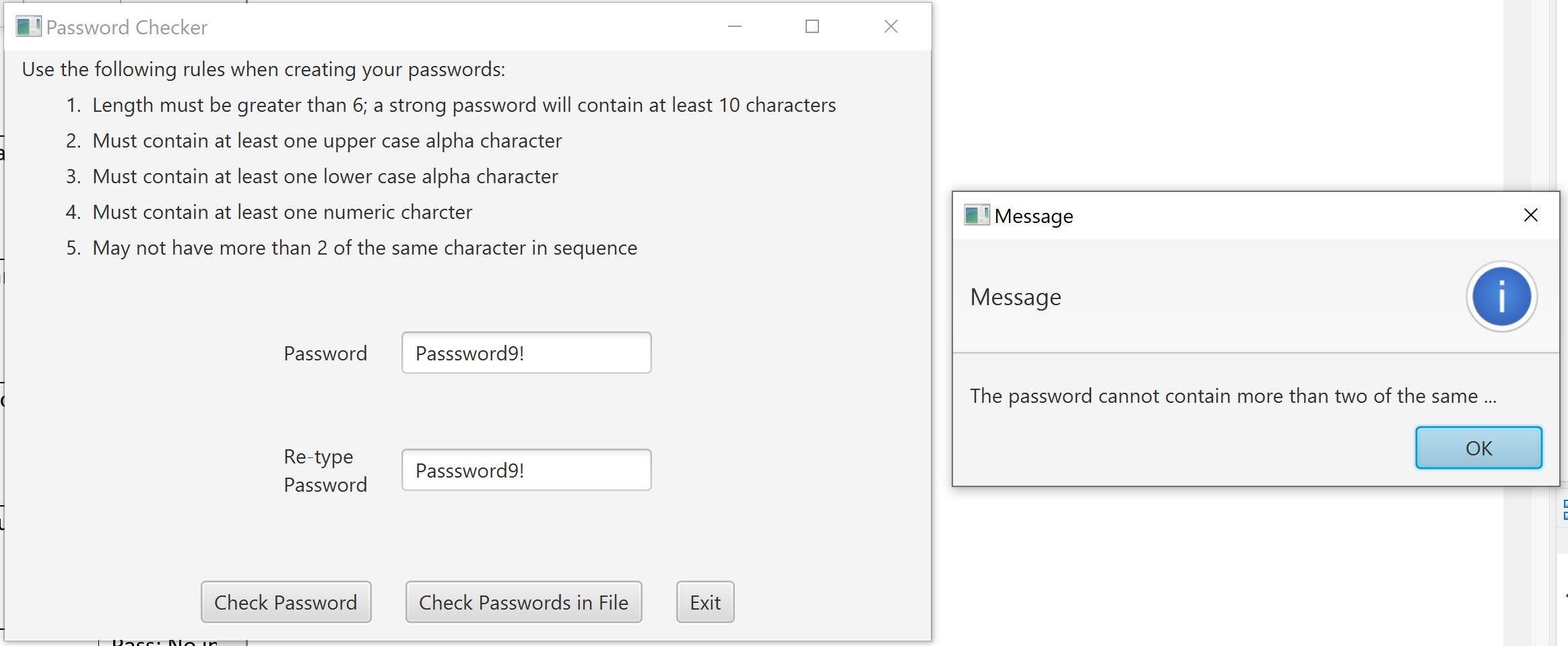
Test 4



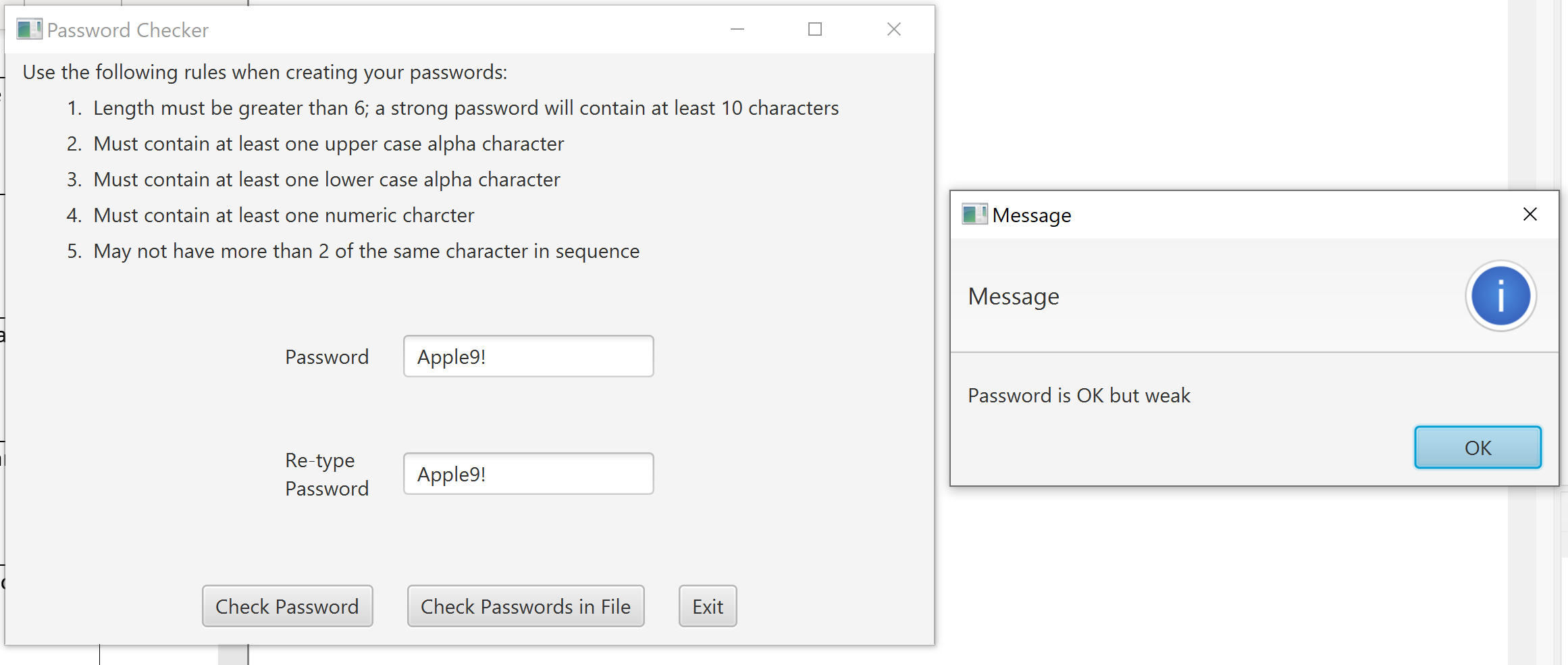
Test 5



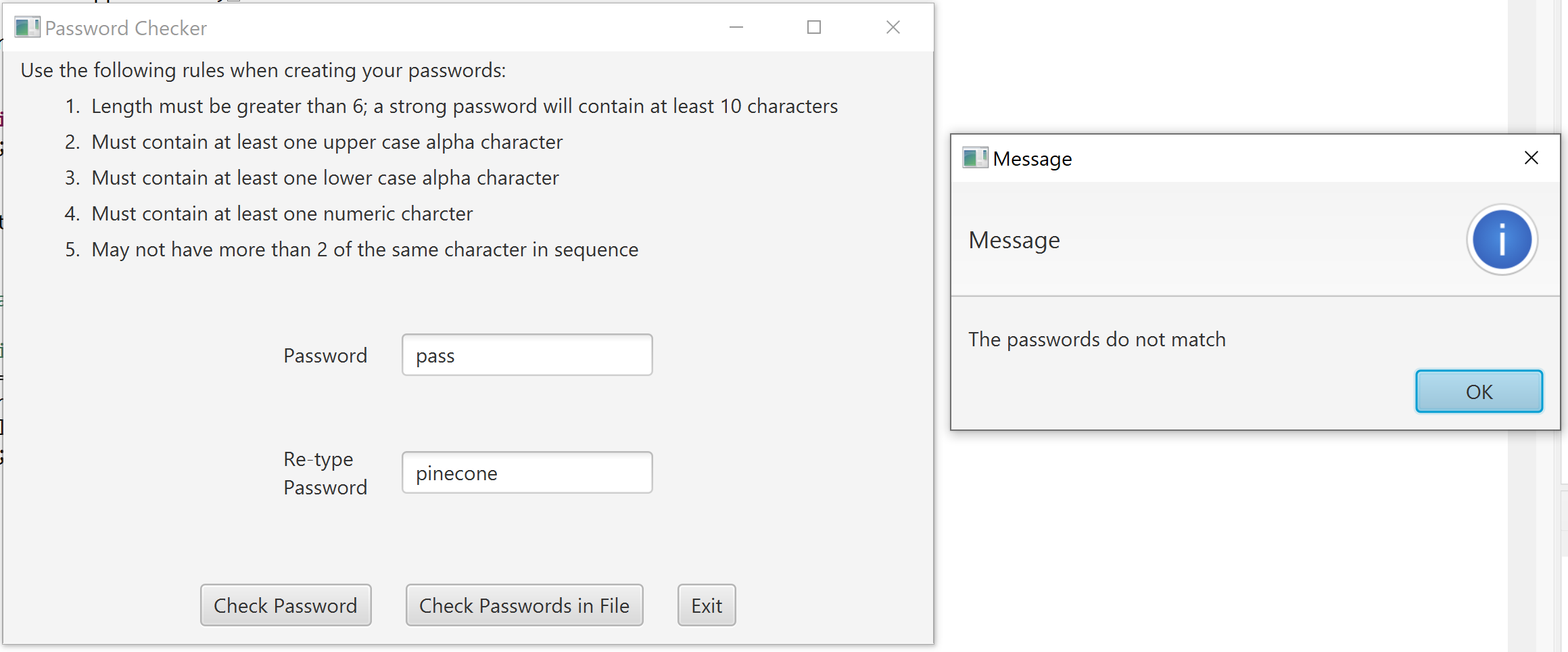
Test 6



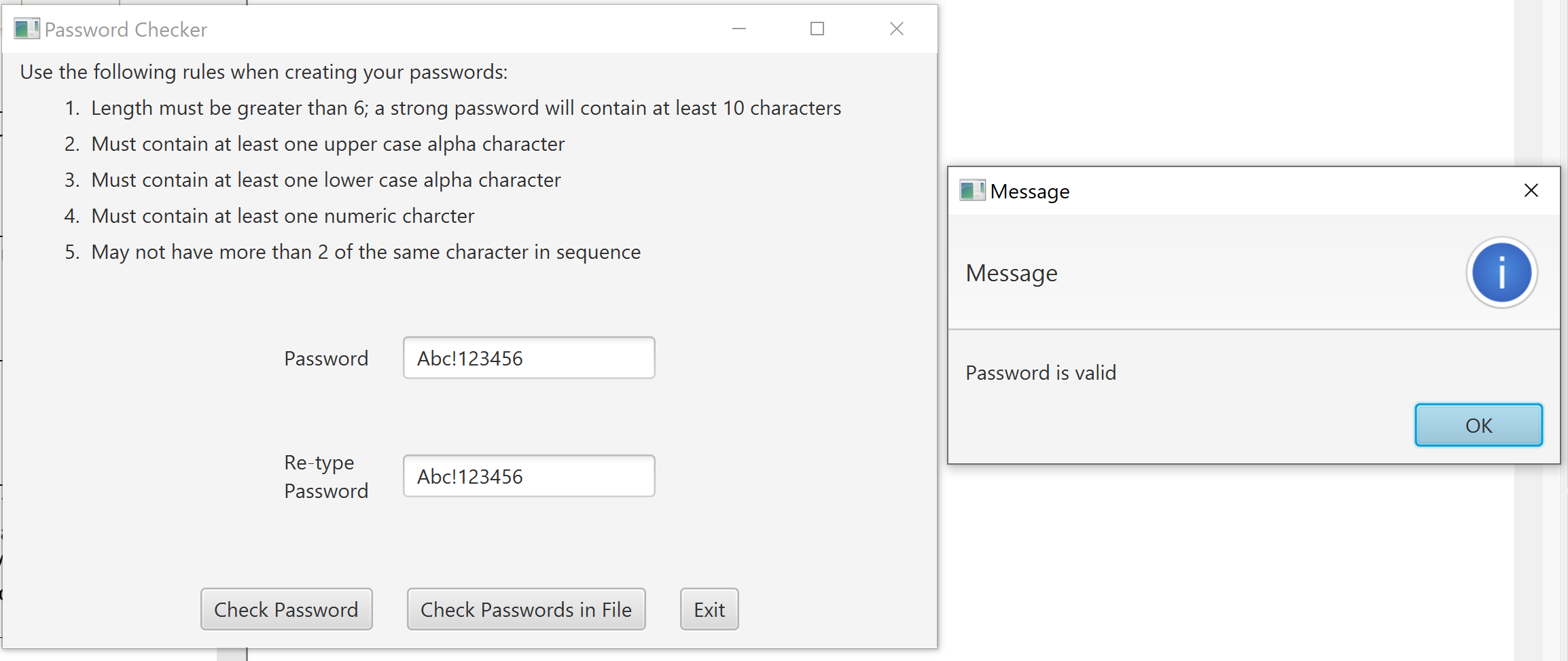
Test 7



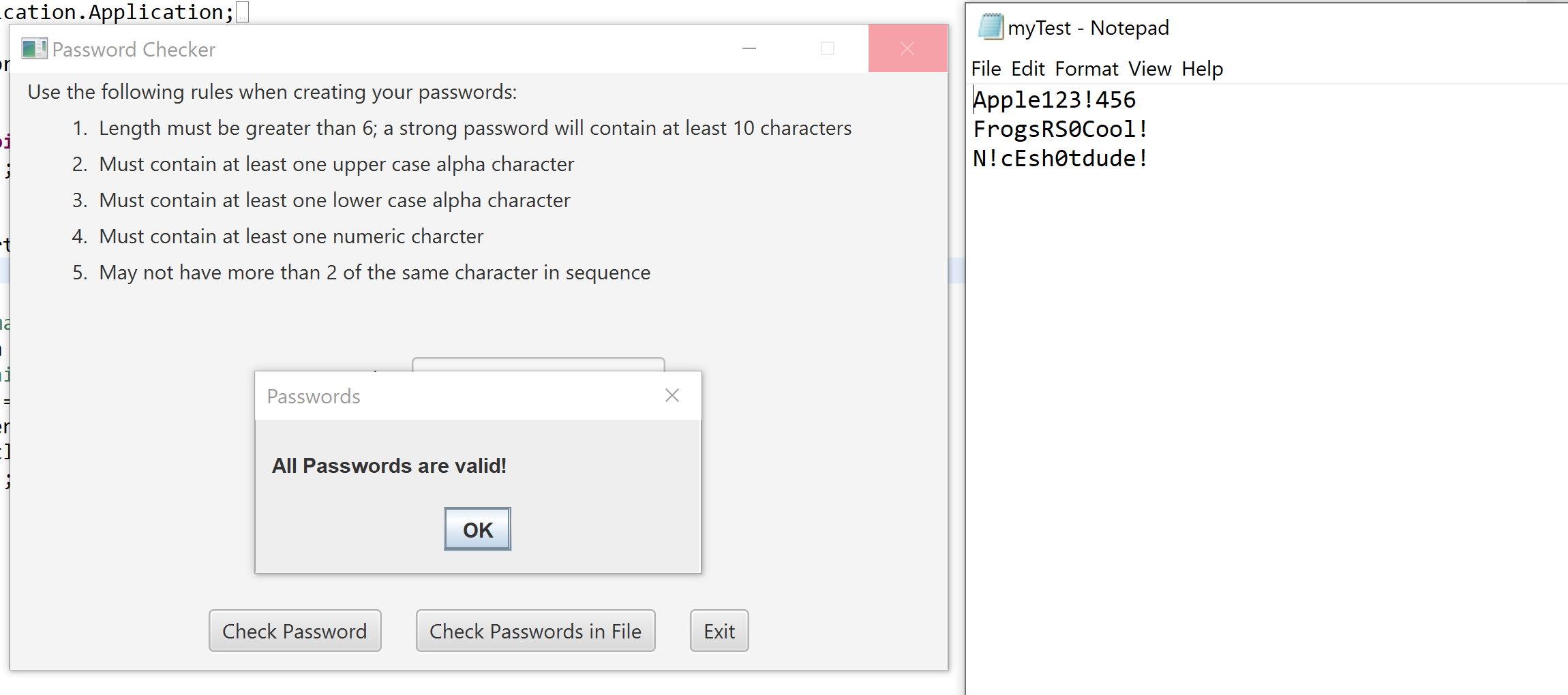
Test 8



Test 9



Test 10



Test 11

