Okeoghene Excel Omorobe

CMSC 204

Assignment 1 Documentation

Learning Experience

This project involved developing a password validation utility, a JavaFX graphical user interface (GUI) for user interaction, and a JUnit test suite to ensure the utility’s correctness. The utility class was responsible for validating passwords based on specific criteria and throwing custom exceptions when validation failed. The JavaFX GUI provided a graphical way for users to interact with the program, displaying appropriate alerts when errors occurred. To verify the correctness of the utility class, a JUnit test suite was implemented to automate testing.

Through this project, I gained experience in creating and handling custom exceptions. I learned how to define exception classes, throw them when necessary, and handle them appropriately within different components of the program. Additionally, I improved my ability to plan program logic, ensuring that the utility class focused on processing data while the JavaFX GUI managed user interaction and exception handling.

Working with JavaFX, I deepened my understanding of implementing mnemonics and tooltips to enhance usability. I also revisited creating custom alert windows for displaying error messages when exceptions were caught. While working with file handling, I practiced using ArrayLists to process data from files of unknown length, making the program more flexible and dynamic.

In JUnit testing, I discovered how to configure test cases to expect specific exceptions using the @Test(expected = Exception.class) annotation. This allowed for more precise validation of error-handling behavior within the utility class.

A challenge that I faced was that some passwords may meet most criteria but fail in subtle ways (e.g., having a special character but still being too weak). Ensuring the utility class properly validates all possible edge cases is crucial. Also, when an invalid password is entered, the exception should be thrown by the utility class and properly handled in the JavaFX GUI without crashing the program.

This project significantly enhanced my understanding of JavaFX, exception handling, and unit testing. It also reinforced the importance of structuring code efficiently and keeping concerns separate between different components. Moving forward, I aim to explore the reasoning behind serialization requirements for nested exceptions and refine my JUnit testing strategies further to improve the robustness of my programs.

Pseudocode

PasswordCheckerUtility Class

* isValidPassword(password)
  + Throws an exception if the password is less than 6 characters.
  + Calls helper methods to check for:
    - Uppercase letter → Throws NoUpperAlphaException if missing.
    - Lowercase letter → Throws NoLowerAlphaException if missing.
    - Number → Throws NoDigitException if missing.
    - Three consecutive identical characters → Throws InvalidSequenceException if found.
  + Returns true if the password meets all criteria.
* Helper Methods
  + containsNumber(password): Returns true if the password has a digit.
  + containsUpperCaseLetter(password): Returns true if it has an uppercase letter.
  + containsLowerCaseLetter(password): Returns true if it has a lowercase letter.
  + hasTriples(password): Returns true if three consecutive identical characters are found.
* Custom Exceptions (Extend RuntimeException)
  + LengthException: Password too short (< 6 characters).
  + NoUpperAlphaException: No uppercase letter.
  + NoLowerAlphaException: No lowercase letter.
  + NoDigitException: No number.
  + InvalidSequenceException: More than two consecutive identical characters.

PasswordMain Class

* readFile()
  + Opens a file chooser for .txt files.
  + Reads passwords from the file into an ArrayList<String>.
  + Returns the list.
* UI Components
  + Text Fields: passwordText and passwordAText for user input.
  + Buttons
    - checkPwdButton
      * Verifies if passwords match (throws Alert if not).
      * Calls isValidPassword(), displaying:
        + Success message if valid.
        + Error message if an exception occurs.
      * Has a tooltip and mnemonic (‘p’).
    - exitButton
      * Closes the program.
      * Has a tooltip and mnemonic (‘x’).
    - checkPwdsInFileButton
      * Reads passwords from a file.
      * Displays:
        + Invalid passwords with error messages.
        + Success message if all passwords are valid.
      * Handles missing file errors gracefully.
      * Has a tooltip and mnemonic (‘f’).