Pseudocode for RmvText.java:

* Non-Constant variables:
  + toRemove to hold what string to remove
  + fileName for the filename to load
  + fileReader to read the file
* Try to do the following:
  + Set toRemove to the first argument passed. If an error occurs, tell the user that there weren’t enough arguments passed and skip the rest.
  + Set fileName to the second argument, if an error occurs, tell the user that there weren’t enough arguments passed and skip the rest
  + Set the fileReader to a new Scanner that looks into a file at the location of fileName in the current folder. If an error occurs, tell the user that the file was not found and skip the rest
  + Check if there’s another line to read
    - If there is, set a temp variable to hold the next line
    - Print out the temp variable once you replace all items toRemove with a blank string.

Test Cases:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Cases** | **Input** | **Expected Result** | **Actual Result** | **Did it pass?** |
| Case 1 | Argument 1: hello  Argument 2:  test.txt  Contents of test.txt:  hello, mike!  My name is hello!  This is to test hello! | , mike!  My name is !  This is to test ! | , mike!  My name is !  This is to test ! | Y |
| Case 2 | Argument 1: random  Argument 2:  test.txt  Contents of test.txt:  I wonder random what would random happen if random  I litterd random the random word r-a-n-d-o-m  in random this text random file? | I wonder what would happen if  I litterd the word r-a-n-d-o-m  in this text file? | I wonder what would happen if  I litterd the word r-a-n-d-o-m  in this text file? | Y |
| Case 3 | Argument 1: blueberry  Argument 2:  test.txt  Contents of test.txt:  I love blueberry raspberry pie!  I once had a blueberry raspberry pie with my family  I am blueberry running out of ideas here.. blueberry..? | I love raspberry pie!  I once had a raspberry pie with my family  I am running out of ideas here.. ..? | I love raspberry pie!  I once had a raspberry pie with my family  I am running out of ideas here.. ..? | Y |

Screenshots:

A picture containing drawing

Description automatically generatedA screenshot of a cell phone

Description automatically generatedA screenshot of a cell phone

Description automatically generated

UML Class Diagram:

A screenshot of a cell phone

Description automatically generated

Lesson’s Learned

For this assignment, I learned how to work with Exceptions and how to catch any errors my programs are making. I used various exception types to catch user errors and attempted to current them/let them know of their errors. Exception catching is very important when making programs for the real world because you can never trust the end user with a simple task, so you must make it as stupidly easy and problematic as possible, as shown in InputMismatch.

Checklist:

|  |  |  |  |
| --- | --- | --- | --- |
| **#** |  | **Y/N** | **Comments** |
|  | **Source java files** | **Y** |  |
|  | **Compressed files:** | **Y** |  |
|  | FirstInitialLastName\_Project8\_Moss.zip | **Y** |  |
|  | FirstInitialLastName\_Project8\_doc.zip | **Y** |  |
|  | **Program compiles** | **Y** |  |
|  | **Program runs** | **Y** |  |
|  | **Checklist is completed and included in the Documentation** | **Y** |  |
|  | **Documentation file:** | **Y** |  |
|  | **Comprehensive Test Plan** | **Y** |  |
|  | **Screenshots based on Test Plan** | **Y** |  |
|  | **UML Diagram** | **Y** |  |
|  | **Algorithms/Pseudocode** | **Y** |  |
|  | **Flowchart** | **Y** |  |
|  | **Lessons Learned** | **Y** |  |