Alvin Tran

CSE 220-50

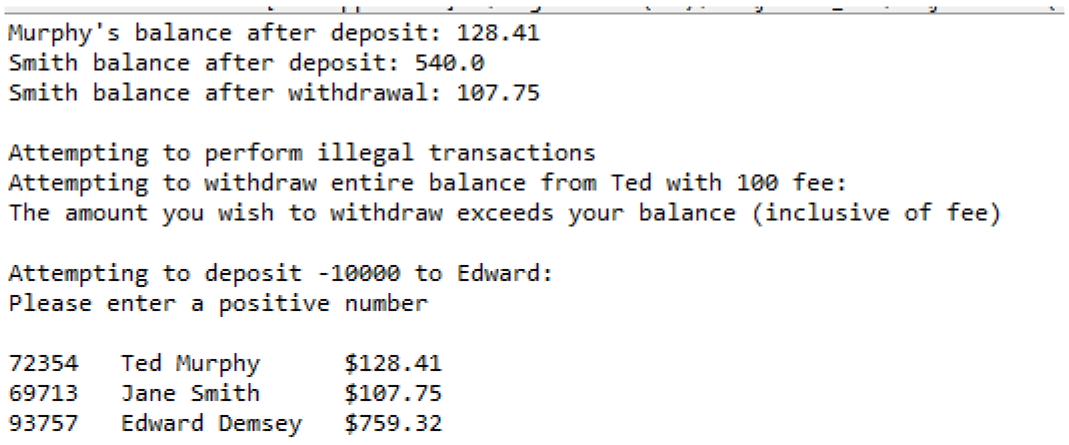
Assignment #3

Submission Date: 6/19/2021

Problem 1

This problem requires use to utilize the given Account class provided from Chapter 4, and change it so that it checks for errors, which is when we withdraw too much money or deposit a negative amount. It should also appropriately give an error message.

We reutilize the given **Account.java** and **AccountTransaction.java** code for our base. After so, we modify the **Withdraw()** and **Deposit()** methods to include an **if** statement. Our condition being the error, which is withdrawing an amount that exceeds the Object's balance (inclusive of fee) and inputting a negative amount for **Deposit()**. In the case there is an error, we simply output an error message and return the balance, otherwise we proceed with normal operations then return balance.



```
Murphy's balance after deposit: 128.41
Smith balance after deposit: 540.0
Smith balance after withdrawal: 107.75

Attempting to perform illegal transactions
Attempting to withdraw entire balance from Ted with 100 fee:
The amount you wish to withdraw exceeds your balance (inclusive of fee)

Attempting to deposit -10000 to Edward:
Please enter a positive number

72354   Ted Murphy      $128.41
69713   Jane Smith      $107.75
93757   Edward Demsey   $759.32
```

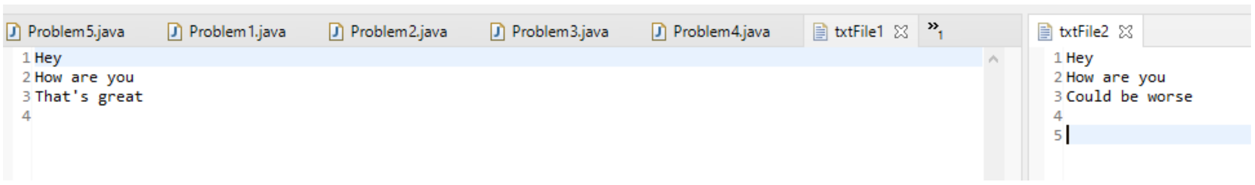
Example Output: Note after/before balances have not changed.

Problem 2

We must design a program that reads lines of 2 different text files, and then compares them. If any are different, then they are printed out.

To preface, the program requires both text files must have equal amount of lines, otherwise program will not work.

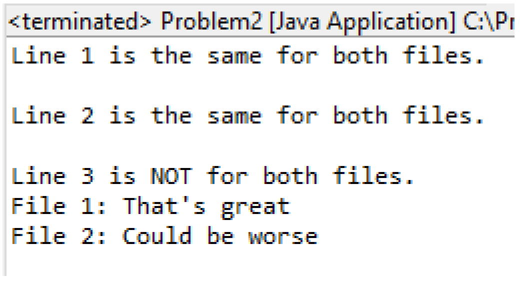
The solve the problem, we require the use of **java.io.file** and **java.util.Scanner**. We create a separate **Scanner** for each file and then use the **File** class to reference the **Path** of our 2 files. After so, we inter a **while** loop that checks if either file has more lines to read, using **scan.HasNext()**. Each time this loop is run, we simply get the **nextLine()** as a String and then use **.equals()** of the String class to check for equality to output the appropriate response.



```
Hey
How are you
That's great

```

```
Hey
How are you
Could be worse


```

Input Text Files; Expected output to throw inequality at Line 3



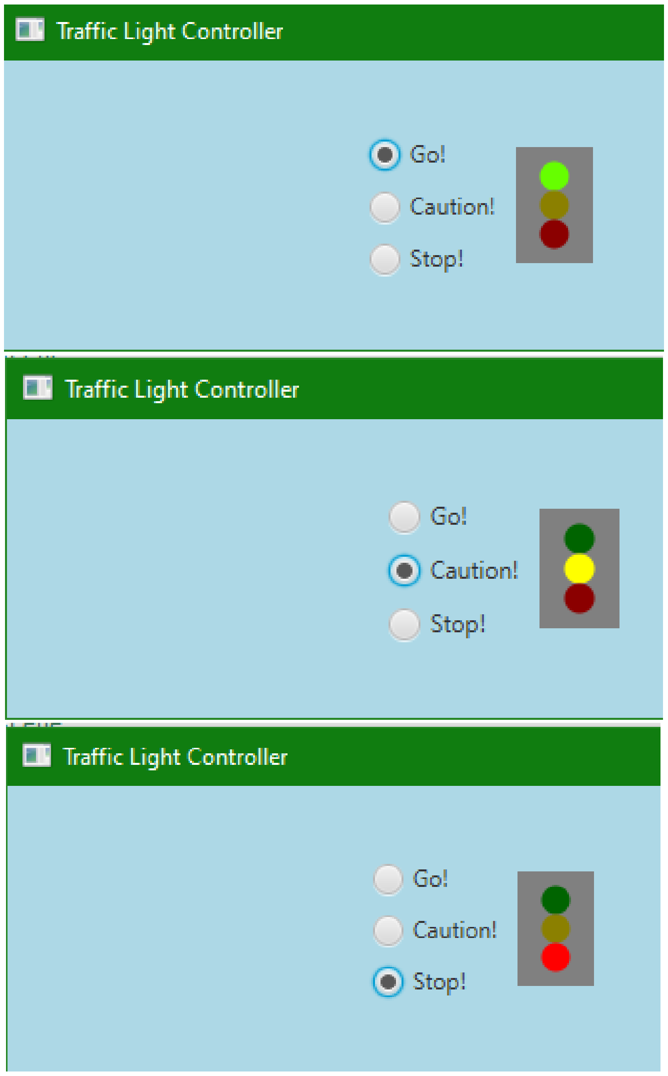
```
<terminated> Problem2 [Java Application] C:\Pr
Line 1 is the same for both files.

Line 2 is the same for both files.

Line 3 is NOT for both files.
File 1: That's great
File 2: Could be worse
```

Matching Output

Problem 3

Problem 3 requires use to utilize the javaFX library to create an interface that controls the state of a traffic light display using radio buttons.

To solve the problem, we create a **Group** with **Rectangles** and **Circles** that represent the drawing of the traffic light. We set them to a darker version of their respective color using **Color.color().** Then, we create a **ToggleGroup** and **RadioButtons** that use that ToggleGroup. We link ALL radio buttons to the same **proccessRadioButtonAction** method. This method first sets all colors to dark, then uses an **if-else** statement to check which button is selected to use **.setFill()** of the **Circle** class to set the color to a lighter color (as if the light is on). We then group the RadioButtons into a **VBox** for vertical formatting and then add both the TrafficLight image group and the RadioButton VBox to a **HBox** to set the image and buttons side-by-side. After that, we proceed with setting Scene and Stage.



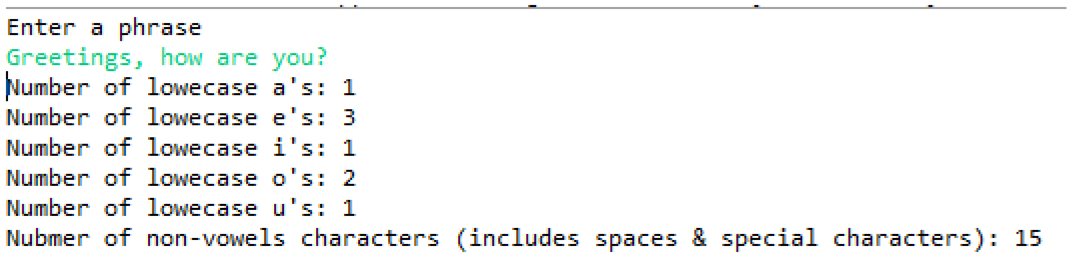
Output: All 3 possible states

Problem 4

This problem requires us to iterate through a user entered String, and then count for each lowercase vowel (aeiou) and every non-vowel.

Note that we do not count uppercase vowels since they do not fall into either category. In addition, special characters like ' ' and '$' are "non-vowel characters"

We utilize the **Scanner** object to read user input, and use a **for** loop to iterate from the 0 index to the last index by setting out condition to be **i < phrase.length()**. After so, we go through a series of **if-else** statements that check if the letter is equal to the appropriate letter we are looking for, and then increase the count variable associated with that character. In the end, we if the letter is not any of the uppercase vowels. Since all the lowercase vowels are iterated over, the only possible options are the uppercase vowels, and every other character possible. We simply just use a series of **!=** and **&&** statements for one last if statement to see if we can count up our "non-vowel" list. In the end, we appropriately output our counts.



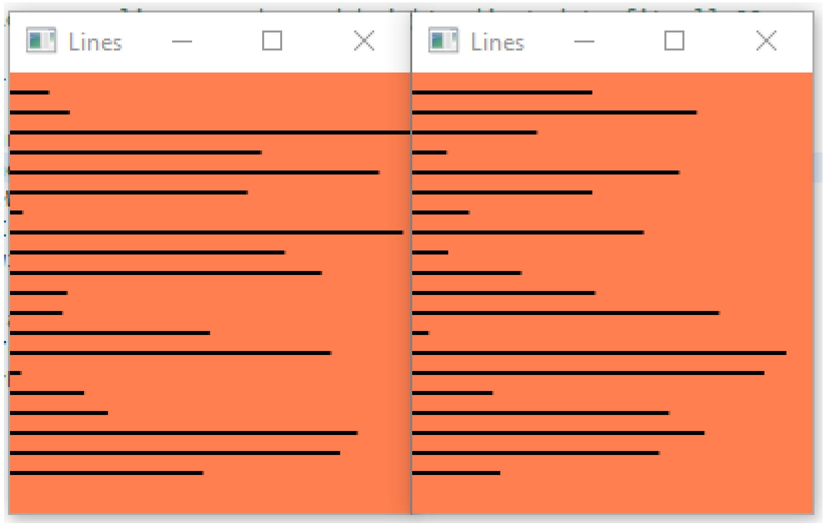
```
Enter a phrase
Greetings, how are you?
Number of lowecase a's: 1
Number of lowecase e's: 3
Number of lowecase i's: 1
Number of lowecase o's: 2
Number of lowecase u's: 1
Nubmer of non-vowels characters (includes spaces & special characters): 15
```

Example Output

Problem 5

Our final problem requires the use of JavaFX to create 20, equally separated, horizontal lines of random length.

We start by creating a **Group** and setting parameters for our initial **yPos** and separation distance **ySep**. After so, we enter a **for** loop that counts from 0 to 19 inclusive (20 total). Each time the loop is created, we create a **new Line()** object. After so, we make sure it starts at **StartX=0** and then set our **Endx** using **Math.Random()** multiplied by our maximum length to create a random length line. We then set our **StartY** and **EndY** to our **yPos** and add our **ySep** to yPos for the next line to create equal seperation. After so, we add out line to our group using **group.getChildren.add(line)**. We then appropriate set the scene, and stage for appearance.



Example Output of 20 Lines: 2 Different Runs