Jonah Largen

Assignment #03

6/07/17

Problem 1

Description: A modified account class that performs validity checks on deposits and withdrawals.

Logic: In the deposit method, check if the amount is greater than 0 to add it to the balance. If not warn the user. After either case, return the balance. In the withdraw method, set the total withdrawal to the amount and the fee. If this is greater than the balance, warn the user. If not subtract this total from the balance. After either case, return the balance.

Input/Output:

Run 1 - {Input Given: None} {Expected Output: 750, 0, 0, 0, 207, 207} {Actual Output: 750, 0, 0, 0, 207, 20} {Successful Output}. The output is as follows.

```
<terminated> AccountDriver [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Jun 7, 2017, 1:44:58 PM)
Demsey Starting balance: $750.0

New Demsey balance after withdrawl: $0.0

Error: Withdrawal amount with fees cannot exceed your balance.
New Demsey balance after withdrawl: $0.0

Error: Must be a valid positive deposit.
New Demsey balance after withdrawl: $0.0

New Demsey balance after interest: $207.0

Final Demsey account information:
93757   Edward Demsey   $207.00
```

Conclusion: The code works as expected and prints the values. The six values are separated by new lines and stem from the following: 1. Starting balance; 2. Withdrawing $748.50 w/ $1.50 fee; 3. Withdrawing $5 w/ $1.50 fee; 4. Depositing $-550; 5. Depositing $200 and adding

interest; 6. Final balance. The balance after steps 3 and 4 remains zero because of invalid operations.

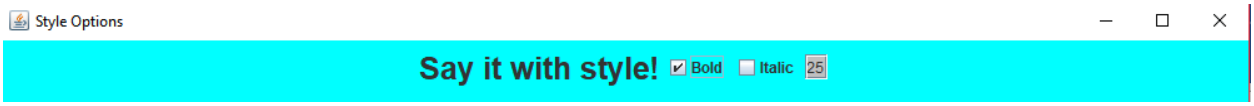
Problem 2

Description: A modified style options class that allows the user to specify the font.

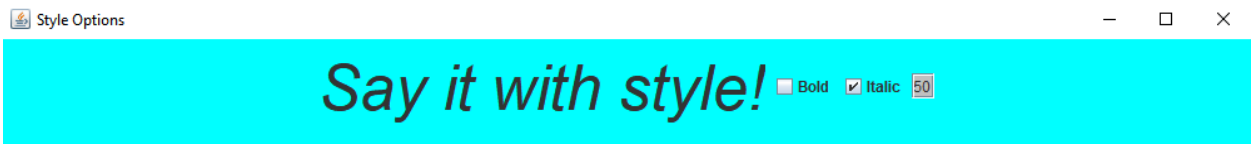
Logic: Add a JTextField to specify the font. Set the background to light gray and set the value to 36, which is the default font size. Add a KeyListener to check when the 'enter' key is pressed, and update the label under such conditions. Add the font to the panel. The font size is obtained by parsing the text box to an integer value.

Input/Output:

Run 1 - {Input Given: Bold – Yes, Italic – No, Font Size – 25} {Successful Output}. The output is as follows.



Run 2 - {Input Given: Bold – No, Italic – Yes, Font Size – 50} {Successful Output}. The output is as follows.



Conclusion: The code works as expected and allows the user to update the font size of the label.

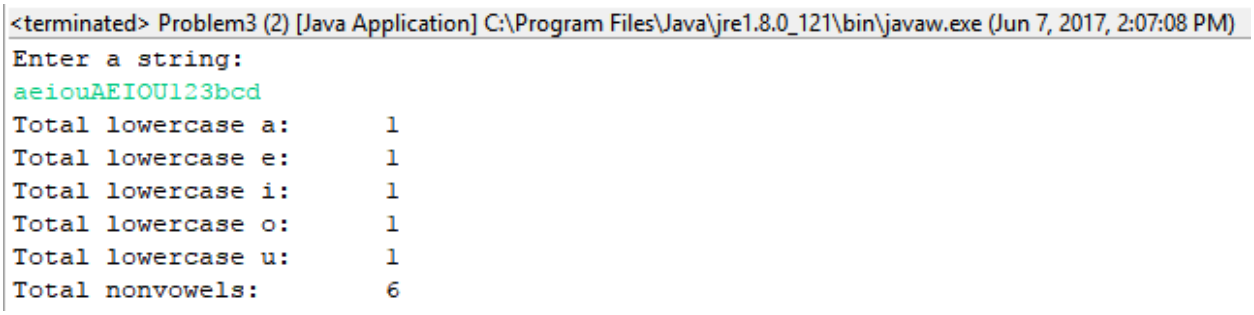
Problem 3

Description: Prints the number of lowercase vowels and non-vowel characters.

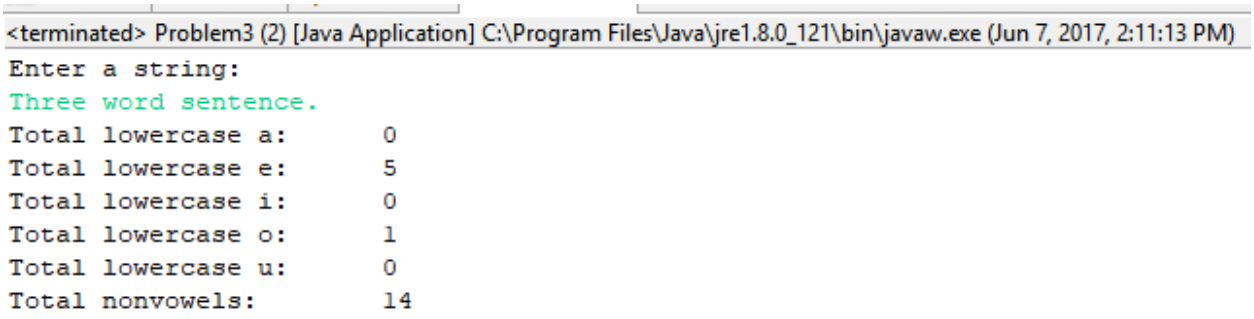
Logic: Scan in a string. Loop through the length of the string one character substring at a time, and check if it is a vowel, an uppercase vowel, or a different character, and add to counters accordingly. Print out all the character counts and close the scanner.

Input/Output:

Run 1 - {Input Given: "aeiouAEIOU123bcd"} {Expected Output: "a" – 1, "e" – 1, "i" – 1, "o" – 1, "u" – 1, non-vowel, "6"} {Actual Output: "a" – 1, "e" – 1, "i" – 1, "o" – 1, "u" – 1, non-vowel, "6"} {Successful Output}. The output is as follows.



Run 2 - {Input Given: "Three word sentence."} {Expected Output: "a" – 0, "e" – 5, "i" – 0, "o" – 1, "u" – 0, non-vowel, "14"} {Actual Output: "a" – 0, "e" – 5, "i" – 0, "o" – 1, "u" – 0, non-vowel, "14"} {Successful Output}. The output is as follows.



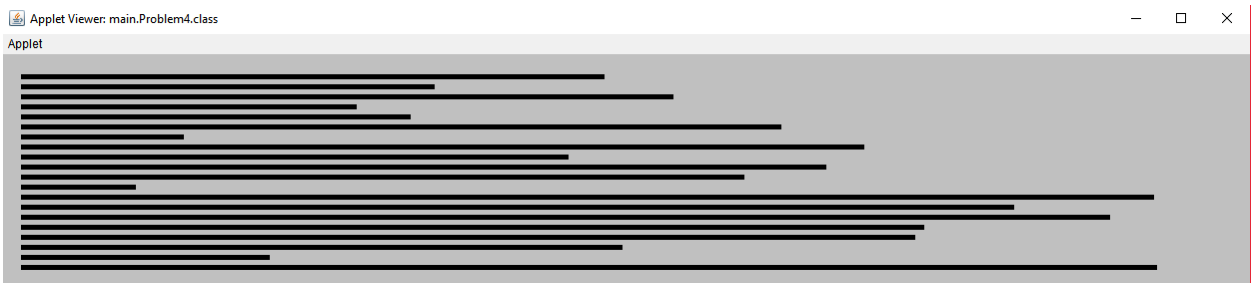
Conclusion: The code works and correctly prints the number of each vowel and the number of non-vowels.

Problem 4

Description: A program that draws 20 horizontal and evenly spaced lines of random length.

Logic: Create an applet with dimensions of 1250 x 250 and set the background color to light gray. Afterwards set the stroke color to black. Loop 20 times, and each time generate a random number between 10 and 1209. Continue in the loop and generate a rectangle at an x of 20, a y of 20 plus the counter times 10, a width of the random number, and a height of 5.

Input/Output: There is no input for this problem. The output is as follows.



Conclusion: The code works and prints the 20 lines.

Problem 5

Description: Create a course class that represents a school class. Create an instance of this class, add students, print the roll, and print the course test average.

Logic: Add a textScoreAverage variable to the pre-made student class. Add it to the constructor, then create getters and setters for all the instance variables. Add a student array and a name as the instance variables of the course class. Make the constructor only accept the name, while setting the array as a blank array instance. Create the getters and setters of the course class. Make an addStudent method that will add the passed student instance to the student array. Add an average method that loops through all the students in the student array and adds his or her test

average to the total. Divide the total by the number of students in the array to get the average and return the value. Add a roll method that prints all the students in the class, their information, and the name of the class. In the driver class, add a decimal format to bring the average to two decimal places. Create an instance of the course class with the name ‘Math 101’. Add 4 students with differing names, home and school addresses, and test averages. Print the roll of the class and the class test average.

Input/Output: There is no input. The output is as follows.

```
<terminated> CourseDriver [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Jun 7, 2017, 2:27:34 PM)
Roll [Math 101]:
---------------
John Smith
Home Address:
1328 4th Street
Louisville, KY  40208
School Address:
345 Brook Street
Louisville, KY  40208
---------------
Jane Doe
Home Address:
951 Green Street
Nashua, NH  1584
School Address:
5893 Main Street
Nashua, NH  1584
---------------
Don Johnson
Home Address:
85 Bohemia Lane
Dorchester Center, MA  1108
School Address:
786 Grape Road
Dorchester Center, MA  1108
---------------
Sarah Jones
Home Address:
87 Wakehurst St
Park Ridge, IL  60068
School Address:
9148 Mountain Road
Park Ridge, IL  60068
---------------
Test Score Average: 84.93%
```

Conclusion: The code works as expected creates a course, prints the roll, and prints the test average.

Problem 6

Description: A program that creates tasks with certain priorities, lists them, and prints the ones of the most and least priority.

Logic: Create the priority interface with a setPriority and a getPriority method. Create a task class that implements this priority interface. Write the required override methods with an integer instance variable for the priority. Add a string variable for the name as well. Create getters and setters for these variables. Create a task driver class. Create an array of tasks of different priorities and names. Loop through all the tasks and print its name and priority. Loop through the tasks again, and check if the task has the highest priority so far. If so, set it as a temporary variable. Upon finishing the loop print the task with the highest priority, and do the same process for the lowest priority.

Input/Output: There is no input for this problem. The output is as follows.

```
<terminated> TaskDriver [Java Application] C:\Program Files\Java\jre1.8.0_121\bin\javaw.exe (Jun 7, 2017, 2:33:04 PM)
To Do List:
---------------
Task: Do Dishes [Priority=1]
Task: Walk Dog [Priority=5]
Task: Cut Grass [Priority=3]
Task: Grocery Shopping [Priority=2]
Task: Clean Bathroom [Priority=4]
---------------

Highest Priority Task: Walk Dog [Priority=5]
Lowest Priority Task: Do Dishes [Priority=1]
```

Conclusion: The code works and prints the tasks with their respective priority, as well as the tasks with the highest and lowest priority.