Wentworth Institute of Technology
COMP1000 – Computer Science I
Fall 2017, Kreimendahl

Programming Assignment 7

In this assignment you are to write a program to solve the following problems. As with all assignments, remember the following submission steps:

- Make sure your code passes at least all the provided JUnit tests
- Create and test Javadoc code documentation
- Save, commit, and push all code changes
- Confirm the latest code is visible via the "Files" section of your repository website
- Confirm that the repository is private, and that the instructor has Developer access

Not that for these problems, some JUnit tests that will be used for grading have not been provided as a part of the starter code. It is your responsibility to thoroughly test your code.

Problem a (`PA7a.java`)

Write a program that reads a stream of integers from the console and stores them in an array. The array is then analyzed to compute the average of all the values in the array and finally all of the values that are above the average should be printed out to the screen. Specifically, you must write three methods: **`main()`**, **`readIntoArray()`**, and **`printAboveAverage()`**.

`main()` creates a scanner, as well as an array of 100 integers, and outputs a message to the screen asking for a sequence of numbers. **`readIntoArray()`** is then called to read values from the scanner and store them in the array. It must be passed two arguments: the scanner and the array. You should only store as many integers as the array can handle. Note, however, that there might be fewer than 100 values typed at the console – store whichever is *fewer*. This method must return how many integers, up to the length of the array, were read into the array. The **`hasNextInt()`** method of the scanner will be useful to determine if there are additional integers to read from the console. Additionally, when you are testing your code in Eclipse, and are done typing integers, press enter (i.e. to proceed to a new line) and then press **`CTRL+D`** (Mac) or **`CTRL+Z`** (Windows) to indicate to Eclipse that you are done typing (this is code for **`EOF`**, or end-of-file).

Finally, **`printAboveAverage()`** should be called to read through the array, compute the average, and then print out all values in the array that are above the average. In particular, for each value above the average it should print the index in the array, as well as the value itself. **`printAboveAverage()`** should take two arguments: the array and the actual number of values in the array. Note that this second argument is not the total number of elements that the array can hold, but is instead the number of values that are valid (i.e. populated in the **`readIntoArray()`** method). For example, the array should be able to hold up to 100 values, but there might have only been 15 values typed at the console.

You have been supplied JUnit tests for the two methods, as well as the output for several example input sequences.

Problem b (`PA7b.java`)

Write a program that analyzes text written in the console by counting the number of times each of the 26 letters in the alphabet occurs. Uppercase and lowercase letters should be counted together (for example, both 'A' and 'a' should count as an A). Any characters that are not letters should be ignored. You must prompt the user to enter the text to be analyzed. Then, for any letter that appeared at least once in the text, print out the number of times it appeared (and do so in alphabetical order). An effective way to count characters is to read from the console string-by-string, and loop through all of the characters of each of these strings. Similar to Problem a, the **`hasNext()`** method of the scanner will be useful, and when testing in Eclipse, press enter and then, once on the empty line, press **`CTRL-D`** (Mac) or **`CTRL+Z`** (Windows) when you are done typing the text.

You must use an array to keep track of how many times each letter is seen in the text. The array should have 26 elements (one for each letter in the alphabet). Index 0 should be used to track the number of A's, index 1 to track the B's, index 2 to track the C's, etc., up to index 25 for the Z's. You could use a massive if/else block, but the whole reason to use arrays is to make your programs easier. So, instead, think about how to convert each character you read into the correct index and then increment that value in the array. For example, if you read an A, then you should increment the value in index 0. Specifically, you will need to determine if the character is an uppercase letter (between 'A' and 'Z'), a lowercase letter (between 'a' and 'z'), or something else. If it is a letter, convert it into the appropriate index. Recall that characters and integers are interchangeable via the ASCII table conversion[1]. Consider this example to help get you started:

```
char input = 'z';
int index = input - 'a'; // index equals 25, as 'z' is 122 and 'a' is 97
```

You have been supplied JUnit tests for several example input texts, including an empty text, one with no letters, upper/lower/mixed case letters, and Hello World.

[1] See http://www.asciitable.com for a reference.