Programming Assignment 5

In this assignment you are to write programs 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 each problem, 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 (`PA5a.java`)

Write a program that can print a banner with rotated letters, such that they can be read from top to bottom. Specifically, each letter in the message should be printed using multiple lines composed of **`*`** symbols. The code to print each letter should go into a separate method just for that letter (no parameters, no return value). Don't worry, you don't need to make 26 different methods, just 8: the seven letters in the message "Hello World" (D, E, H, L, O, R, and W) and a space. In your main method, you should ask the user to type a message. Then, loop through each character in the message and, for each character, if you have a corresponding method, call the method to print out the letter to the screen, ignoring any other characters. To help get you started, here is an example method for the letter D:

```
public static void d() {
	System.out.printf("%n");
	System.out.printf("*******%n");
	System.out.printf("*     *%n");
	System.out.printf("*     *%n");
	System.out.printf(" *   * %n");
	System.out.printf("  ***  %n");
	System.out.printf("%n");
}
```

And an example method for the blank space (notice each has a blank line before/after):

```
public static void blank() {
	System.out.printf("%n%n%n");
}
```

Reference the JUnit tests to see how to print the other letters. Here are a few tips for your main method:

- To read an entire message, including spaces, use a Scanner variable's nextLine() method.
- To get the number of characters in a string variable, use that String variable's length() method.

- To get a particular character in a string, use a String variable's charAt() method. Note that this method will want to know *which* character, and so you should provide it as an argument. Remember that in Java we begin counting at 0, so to get the third character of the String variable myMessage, you would write myMessage.charAt(2).
- You should ignore case – that is, call the d() method whether the message contains a "d" or a "D".

Here is an example run of the program (notice the missing characters):

```
Input message: Hello Bye!

*******
   *
   *
   *
*******


*******
*  *  *
*  *  *
*  *  *
*  *  *


*******
*
*
*
*


*******
*
*
*
*


*******
*     *
*     *
*     *
*******





*******
*  *  *
*  *  *
*  *  *
*  *  *
```

You have been supplied JUnit tests for some messages that contain no valid letters (and thus produce no output), messages that are entirely composed of blank spaces, messages only composed of one valid letter, and a "Hello World!" example.

Problem b (`PA5b.java`)

Write a program that calculates the average of a stream of non-negative numbers. The user can enter as many non-negative numbers as they want, and they will indicate that they are finished by entering a negative number. For this program, zero counts as a number that goes into the average. Of course, the negative number should not be part of the average (and, for this program, the average of 0 numbers is 0). You must use a method to read in the numbers, keep track of the running sum and count, compute the average, and return the average to the main() method. Note that you must use a loop to do this, as you don't know how many numbers the user will enter ahead of time.

The main() method must print the average (which was returned from your method) using two decimal places, and then ask the user if they want to repeat the entire process. The user should be allowed to repeat the process as many times as they wish. This means you will have a loop in your main() method that instructs the user what to do, calls your average method, reports the result, and finally asks the user if they want to calculate another average for a different set of numbers.

Importantly, your program will not function correctly if is more than one Scanner object looking to the keyboard. Thus, since both main() and your averaging method need to get input from the keyboard, you will have to declare and initialize a Scanner variable in main(), then pass it as an argument to the averaging method. Furthermore, note that your method that reads in the numbers and computes the average must have no System.out.printf() statements – all output must happen in main().

Be sure you take the time to think through the pieces and, more importantly, implement in small steps. I suggest you proceed like so:

1. Get the structure of defining and calling your method down first, with no loops involved.
2. Fill in the method with a loop that runs while the user enters non-negative numbers.
3. Modify the loop to keep track of the sum and count, then print the average in the function.
4. Modify the method to actually *return* the average.
5. Modify main() to use another loop to ask the user if they want to repeat the entire calculation.

At each step, run your program, test it out, and convince yourself that it is correct before moving on. Here is an example run of the program:

```
Enter a stream of non-negative numbers (negative when finished): 1 2 3 -1
The average is: 2.00
Do you want to compute another average (y/n)? Y
Enter a stream of non-negative numbers (negative when finished): 1 2 0 3 -2
The average is: 1.50
Do you want to compute another average (y/n)? y
Enter a stream of non-negative numbers (negative when finished): 0 -1
The average is: 0.00
Do you want to compute another average (y/n)? y
Enter a stream of non-negative numbers (negative when finished): -2
The average is: 0.00
Do you want to compute another average (y/n)? N
```

You have been supplied JUnit tests for number sequences that stop immediately (i.e. before any non-negative values are entered), as well as single sequences of both integer and decimal numbers.