COSC 102 Spring 2015

Debugging & Loop Lab 2

Due date: **Tonight 11:55 p.m. on Moodle**
Make sure you show your lab instructor the work you accomplished before leaving the lab.

In this lab, you will first fix errors and complete a program that is supposed to analyze character patterns in an input string; then you will write a program from scratch.

Setup

Today you are using `DrJava`. It is important to keep your computer files organized.

In your `cosc102` directory create the following structure to be used for your `.java` files.

- Create two directories: `lab` and `hw`.
- Create a directory `02` in `lab` for today.
- Save all your work of `Lab 2` in `cosc102\lab\02`.
- Repeat and use such structure for all your weekly homework and labs.

Ask advices to your lab instructor if you encounter problems setting up `DrJava` and the Java SDK on your personal computer. Use the lab laptop if your computer environment isn't yet functional. Drop in during office hours and Open Labs this week to get help configuring your computer.

Part 1: First Hour

The file StringAnalyzer.java contains a program that is supposed to adhere to the following specifications.

- Ask for a line of input from the console
- Produce the output for the following four tasks.
  - The first "repeated" character in the string, i.e., the first character to appear more than once but in adjacent positions;
  - The first "multiple" character, i.e., the first character to appear more than once (in not necessarily adjacent positions);
  - The number of repeated-character groups (i.e., the string "abbabbbcc" has three, for "bb", "bbb", and "cc");
  - The most frequently occurring character.

Unfortunately, the file has *syntax and logical errors* and *is incomplete*.

Your job is to take over the coding process from the original author. You have to

- fix the errors so that the program compiles,
- update and complete the program so that the output is correct for each of the four tasks and
- fully test each method by entering different input: write as comments in the `main` method the output for each input you entered.

In this part you are practicing debugging, so

- read the comments describing the process to follow, the method description and giving hints,
- think carefully about what might break the code and
- trace the code, i.e., follow the execution of code by reading the source code directly, run the code with different test inputs to understand the problem.

Make sure to consider different types of strings as inputs to test your program. In particular you should always tests the following particular `String`s:

- the empty string
- single character strings
- strings with relevant characteristics at different location: in our case repeats and multiples in different places (e.g., the beginning, the end, etc.)

Your program should *never* crash – meaning, you should never see an interruption where the virtual machine reports an *exception*. So, test thoroughly! and report your tests with comments in the `main`method.

Learning testing is fundamental to become an experienced programmer.

Hints

- Programming languages often have explicit ways to change the control flow inside of a loop.
  Two such statements in Java are `break` and `continue`. Check from Appendix B the segment **B.66**.
  Experiment with either of them to understand how it exactly works.
- You have seen many `String` methods. Search through the Java API of the `String` class to find out details about the methods you need.

Reference Implementation

The file RefStringAnalyzer.class contains bytecode for a similar working version of the program. To run this version, type

```
java RefStringAnalyzer
```

in the `Interactions` pane of `DrJava`. The `RefStringAnalyzer.class` file should be put in the directory `cosc102\lab\02`, i.e. the one you are currently working in. Feel free to compare the behavior of the two programs to help you understand the expected behaviour.

Part 2: Second Hour

Your goal is to write a program that displays on the console a structure as shown on the next page.

Observe the relation between the numbers on each line: the ones on line $n+1$ are deduced from the ones on line n. The first line contains 1 centered according to the number of lines used to produce the triangle.

Work with a partner on a piece of paper

- to understand the structure layout: draw many triangles,

- to design your code: what are the helper methods needed? which constants to define to make your code adaptable and
- to quickly read through how to use `System.out.printf`.
  Use `System.out.printf("%4d", intVal);` to format output into fields of width 4.

```
                   1
                 1   1
               1   2   1
             1   3   3   1
           1   4   6   4   1
         1   5  10  10   5   1
       1   6  15  20  15   6   1
     1   7  21  35  35  21   7   1
   1   8  28  56  70  56  28   8   1
 1   9  36  84 126 126  84  36   9   1
```

Individually create a new `.java` file and implement the methods you devised in turn. Test each. I strongly recommend getting first the numbers on each line and working on the specific centered layout after. You should know how to compute the spacing by hand before trying to program it.

Submission

Submit only your modified source code for `StringAnalyzer.java` (and the `.java` for the triangle if you desire).

Be careful **not to submit the bytecode (.class) or backup versions (.java~)**.