Lab 4 – Iteration

1. Develop algorithms for the following Java programs. Once the algorithms are **documented**, you will then write the code.
2. There is a famous story about a primary school teacher who wanted to occupy his students’ time by making the children compute the sum of 1 + 2 + 3 + … + 100 by hand. As the story goes, the teacher was astounded when one of the children immediately produced the correct answer: 5050. The student, a child prodigy, was Carl Gauss, who grew up to be one of the most famous mathematicians of the eighteenth century.

Write a program called **CountLoop.java** that contains a main method with a loop that will compute and print the sum of all the integers between 1 and 100, inclusive. Execute your program to verify that the output is correct.

1. After you have the program working, refactor, change the code, it so you can compute 1 + 2 + …

+ n where n is any positive integer. In other words, your revised program should prompt the user to enter a number, read the user input, calculate the sum of the integers from 1 to the input value, and output the sum.

1. Java provides three types of loops: *while*, *for*, and *do while*. Theoretically, they are interchangeable - any program you write with one kind of loop could be rewritten using any of the other types of loops. As a practical matter, though, it is often the case that choosing the right kind of loop will make your code easier to produce, debug, and read. It takes time and experience to learn to make the best loop choice, so this is an exercise to give you some of that experience.

You will add to the current **CountLoop.java** program code. First, add a blank output line below the current output. Next, copy your current loop and the blank line below it and paste it two times below the current code. Now, configure the two copies of the loop so that you have three different versions of the loop: a *while* loop, a *for* loop, and a *do while* loop. Please note that your program should ask for user input only once, and it should use the same user input for all three loops. You will output the results from each loop, so you will have three outputs, one from each type of loop. (Note: While you are reconfiguring your loops, remember to make sure any counter variables are reset to zero before starting a new loop.)

Save and run the **CountLoop.java** program showing the three results, one for each loop in the program. Take a screenshot of this code running in the terminal. Make sure that you add comment lines for your algorithm to each loop.

1. Some computations require multiple loops where the “inner” loop is code nested inside the body of the “outer” loop code. Suppose we want to produce the following output using a nested looping control structure that outputs a single ‘\*’ character each time through the inner loop:

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

The outer loop will manage the rows and the inner loop will count the number of ‘\*’ characters to be displayed on that row. Write a program called **NestedLoopTriangle.java** containing the code that will produce the output described above using nested loops.

1. Once you’ve got the right triangle to display correctly, add another set of loops to produce a second triangle that looks like the triangle below. Hint: You might need a third nested loop that will increase the spaces as the number of asterisks decreases.

\*

\*\*\*

\*\*\*\*\*

\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

NOTE: When finished, **NestedLoopTriangle.java** should print out **both** Triangles.

1. Save and run the **NestedLoopTriangle.java** program showing the two triangles. Take a screenshot of this code running in the terminal. Make sure that you add comment lines for your algorithm to each loop.
2. Submit your completed **CountLoop.java** and **NestedLoopTriangle.java** files along with your two screenshots of terminal output to GradeScope.

## 

## Rubric

| Topic | Points |
| --- | --- |
| Includes the two screenshots | 5 |
| CountLoop.java produces correct output with any valid integer input | 10 |
| CountLoop.java contains all three kinds of loops with correct output for each loop | 30 |
| NestedLoopTriangle.java produces both triangles with correct output | 20 |
| NestedLoopTriangle.java can be made to work for triangles of any size with minimal change | 10 |
| Programming Style (e.g. indentation, variable names, etc.) | 5 |
| NestedLoopTriangle.java has a well commented algorithm that demonstrates understanding of iteration | 10 |
| CountLoop.java has a well commented algorithm that demonstrates understanding of initialization and iteration | 10 |
| Total | 100 |

#### 