Lab 5 – Counted Loops

Part 1: Common Loop Errors

Find the *BlueJ* project folder named `Lab5_Start` in the `Resource` folder. Copy this folder to your home directory. Rename your copy to `Lab5_lastname_firstname`. This project contains a program which you will use to examine some common loop errors.

Processing is Incorrect for the First and/or Last Iteration (Off By One Error)

Open the class and look at the program. It is supposed to write the integers from 1 to 10 on a line. Trace the first and the last iterations of the loop. Verify your trace by running the program. (Compile the program if necessary).

Fix the program so that it produces the correct output.

The Loop Executes Zero Times

A loop will execute zero times if the loop control condition is false the first time it is tested. Sometimes this is the expected behaviour, but it can also be caused by an error in logic.

To see how this can happen, change the loop control condition from (i < = 10) to (i > = 10). Recompile the program and run it. What happens? Can you explain why this happened?

The Loop Executes Infinitely Many Times

If something does not happen in the body of the loop to eventually make the loop control condition become false, then the loop will execute infinitely many times or until the program is manually terminated.

To see how this can happen, change the loop control condition back to (i < = 10) and then change the line i = i + 1 to i = i – 1. Recompile the program and run it. What happens? To terminate the program, right click on the rotating red bar at the bottom of the main *BlueJ* window and then click on *Reset Java Virtual Machine*.

Can you explain what happened when you made this change?

Carelessness or poor coding habits can be the cause of an infinite loop. For example, return the code to its correct state. Compile and run the program to check that it produces the correct results. Now that it is running properly, edit the code by removing the braces ("{" and "}") which define the block that is the body of the while loop. Recompile the program and run it. What happens? Terminate the program as before.

Can you explain why this became an infinite loop?

Lab 5 – Counted Loops

Part 2: Designing Loops

Case Scenario:

Teachers in many areas are paid on a schedule which provides a salary based on number of years of teaching experience. For example, a beginning teacher in a given school district might be paid 30000.00 the first year. For each year of experience after this up to 10 years, a 2% increase over the preceding value is received.

For example, if the input is 30000.00, 0.02 and 10 the output should look like:

```
      SALARY SCHEDULE
Years of Service    Salary
        1           30600.0
        2           31212.0
        3           31836.24
        4           32472.9648
        5           33122.424096
        6           33784.87257792
        7           34460.570029478404
        8           35149.781430067975
        9           35852.77705866934
        10          36569.832599842724
```

A partial algorithm for this is given below. Fill in the missing parts shown by ????.

```
read salary
read pctgIncrease
read numYears
count = 1

while (????)
{
   salary = salary + salary * pctgIncrease
   print salary (according to the format above)
   count = ????
}
```

Required:

Once you have completed the above algorithm, translate it into a Java program as follows:

1. Add a class called `PartA` to your `Lab3` project. Don't worry about formatting the output until you have the program running.

2. To test your program, enter the values used in the above example and see if your results are the same.

3. Once your program is producing the correct results, format the output so that it is in columns as shown above. Add the heading lines if you have not already done so.

Submission:

Submit the entire Java folder to the submit drive

Extra Challenge:

For those who may have more time or want an additional challenge, modify your program so that the table has a first line as:

```
0        30000.0
```

To do this, add a new class called `PartB` to your project. Copy all the `PartA` code from `public static void main()` to the end of your working program and paste it into `PartB`, replacing everything in `PartB` from its `public static void main()` to the end. Now if you make errors, `PartA` will still be OK. (We will show you another way of keeping the previous version of your program without having to create a new class at a later time).

Note: There are two basic ways to do this, one more difficult than the other. Examine the algorithm and think about how to modify it (and then desk check the changes) ***before*** starting to change any code