CSCI 1301 – Programming Principles I
Georgia Southern University
Department of Computer Science
Fall 2024

Lab 7
Point Value: 20 points
Due: Friday October 11, 2024, end of lab

Objectives
1. Use different types of loops to write working programs.
2. Use nested loops to complete certain types of tasks.
3. Eliminate break/continue while retaining equivalent functionality.
4. Use the debugger to effectively diagnose and fix errors in code.
5. Archive Java source code files and other documents into a zip file.

Description
Complete the following steps as described and upload requested materials to Folio in the appropriate dropbox in a single zip file. Submit Java source code files to Gradescope.

- Students should work in groups of two for this lab. When submitting your zip file, fill out the comment header and indicate both group member names.

- Your submitted zip file should be named Lab#LastName1LastName2.zip. For instance, Lab7JonesSmith. Any requested screenshots and/or answers to questions should be included in a separate Word document.

- All programs should have a comment header. That comment header should include information similar to:

```
/**
 * File: PAssign0.java
 * Class: CSCI 1301
 * Author: Christopher Williams
 * Created on: Jun 6, 2016
 * Last Modified: Aug 16, 2018
 * Description:  Display three messages to the console
 */
```

- Before writing any code, for each problem, discuss with your partner how you plan to approach solving the given problem. Once you both agree on a way to attempt the problem, only then should code be written.

- Pay attention to names and any other requested material. Failure to include material will result in the loss of points as described below.

Lab Problems
For all problems, use inline comments to describe major actions. Ask for assistance if needed.

1. Nested Loops – Using nested for loops, write a Java program named Lab07Prob01.java that creates the following pattern based on a variable for the maximum number of asterisks in the top line (assume the value is 5 for this output). It may help to complete a hand-trace of the loop-control-variables (i and j) and their values during execution of this program to replicate this pattern.

   Expected Output:
   *****
   ****
   ***
   **
   *

   After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

   After instructor or TA check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

2. break/continue – In a Java program named Lab07Prob02.java rewrite the following code so that the functionality remains the same (it is highly suggested you test the code before modifying it), but so that there is no use of break nor continue. This will require adding a logical operator to the for loop's continuation condition (you may have been advised to avoid this, here is a time where it is needed) to replicate the break.

```
Scanner input = new Scanner(System.in);
int userInput = 0;

System.out.print("Enter a number: ");
userInput = input.nextInt();

for (int i = 0; i < userInput; i++) {
   if (i == 25) {
         break;
   }
   if (i == 3 || i == 7 || i == 18 || i == 23) {
         continue;
   }
   System.out.printf("%d ", i);
}

System.out.printf("%nYou entered %d%n", userInput);
```

```
System.out.println("Program Completed");
```

Expected Output (Input of 5):

```
0 1 2 4
You entered 5
Program Completed
```

Expected Output (Input of 9):

```
0 1 2 4 5 6 8
You entered 9
Program Completed
```

Expected Output (Input of 25):

```
0 1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 19 20 21 22 24
You entered 25
Program Completed
```

Note that Gradescope will provide its own input for this program and your program should correctly work for any given input, not just your tested values.

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

After instructor or TA check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

3. Nested Loops – Write a Java program that uses a for loop to approximate cos(x) using the following Taylor series expansion:

$$\cos(x) \approx \sum_{i=0}^{7} \frac{(-1)^i}{(2i)!} x^{2i}$$

This program should use an x of $\frac{\pi}{6}$, and should calculate the approximate value based on the series expansion. The actual value of cosine and the approximation should both be printed to the console with descriptive text. Recall that the number of terms determines how many times this series adds another calculation to the running sum. Review existing approximations from slides or previous labs. Recall that you will need an inner loop to calculate the factorial and that must be done for each term. **HINT:** Make sure you are calculating factorial(2i), not 2 * factorial(i).

Your code should calculate, then display the approximated cosine value as well as the actual value of the requested cosine. The actual value of cosine and the approximation should both be printed[1] to the console with descriptive text up to 16 decimal places.

Expected Output (numTerms = 7):

```
Actual Cos (PI/6): 0.8660254037844387
Approx Cos (PI/6): 0.8660254037844386
```

When submitting to Gradescope, use a numTerms value of 7 or your program will not pass the tests.

After passing the Gradescope tests, show a working version of this program to the instructor or TA before moving on.

After instructor or TA check-off, BOTH students should submit a final, identical version to Gradescope. This version is used for your final grade.

4. Demonstrate all working programs to the instructor or TA to receive credit for each problem. Turn in each problem inside the zip file detailed below.

5. Name the zip file Lab7LastName1LastName2.zip (e.g. Lab7JonesSmith) and submit that zip file to Folio. Both students must submit the exact same zip file for either to receive a grade.

Grade Breakdown

Demonstrate running application to instructor or TA:	
Problem 1	6 points
Problem 2	6 points
Problem 3	6 points
Zipped source files (in Folio):	2 points
Total Possible:	20 points

Last modified: October 7, 2024

[1] You can hard-code the x-value in the parentheses since it uses PI.