**CS1401 – Programming Assignment 4**

**Due: Monday September 28, 2015 by 8:00 a.m. for those of you taking the MW labs**

**Tuesday September 29, 2015 by 8:00 a.m. for those of you taking the TR labs**

**To be submitted via Piazza in the folder “lab4”**

**Welcome to Lab 4!**

|  |
| --- |
| This week, you are going to continue getting acquainted with the Java programming language. We have prepared a skeleton code for you to tinker with. This lab will give you the opportunity to increase your fluency with:   * Input / output, including using files * Variables, computations * Conditional statements   And you will start working with Strings.  Have fun!  Ok, let’s get started! Here are the two activities you will be working on. |

**You should expect to work about 2 to 3 extra hours outside the lab session to complete this assignment:**

**This includes completing the activities and taking the time to make sure that your submission is picture perfect!**

**Activity 1.** In this activity, you will implement a little number-guessing game.

First you need to store in a file called mysteryNumbers.txt the 10 numbers that need to be guessed (we will explain how these numbers should be picked). Then you ask the user to guess 10 times. Each time the user guesses, you keep track of his/her score as follows:

* e.g., if the number to be guessed is 8 and the user guesses 10 (or 6), you assign 2 as score
  + then next time, you will keep adding to that.

Ok, so here it goes: your program, and game, should unfold as follows.

|  |
| --- |
| 1. Randomly generate 10 numbers between 0 and 10 2. Store these numbers into a file called mysteryNumbers.txt, a number per line 3. Prompt the user to enter a number between 0 and 10 4. Check this number against the **first** number in the file 5. If the user guesses right, subtract 10 from the score (at the start, the score is 0) 6. If the user guesses wrong, add the correct amount of score to the current score of the user (at the start, the score is 0) 7. Prompt the user to enter a number between 0 and 10 8. Check this number against the **second** number in the file 9. If the user guesses right, subtract 10 from the score 10. If the user guesses wrong, add the correct amount of score to the current score of the user 11. Prompt the user to enter a number between 0 and 10 12. Check this number against the **third** number in the file 13. If the user guesses right, subtract 10 from the score 14. If the user guesses wrong, add the correct amount of score to the current score of the user 15. … etc. … 16. Display the score of the user: of course, the lower the better! |

**Example:** Imagine that the 10 numbers are as follow: 9, 5, 4, 2, 3, 9, 1, 7, 6, 8, and that the user’s sequence of guessed numbers is: 10, 6, 4, 1, 5, 8, 2, 7, 3, 1. Then the final score of the user is: 1 + 1 – 10 + 1 + 2 + 1 + 1 – 10 + 3 + 7 = -3.

Here is why:

Your file: mysteryNumbers.txt

|  |
| --- |
| 9  5  4  2  3  9  1  7  6  8 |

The original score of the user: Score = 0.

The user’s successive (10) guesses:

1st guess: 10 🡺 compared to 9 in the file, this is a difference of 1 🡺 now: Score = 1

2nd guess: 6 🡺 compared to 5 in the file, this is a difference of 1 🡺 now: Score = 2

3rd guess: 4 🡺 compared to 4 in the file, this is a perfect guess 🡺 now: Score = -8

4th guess: 1 🡺 compared to 2 in the file, this is a difference of 1 🡺 now: Score = -7

5th guess: 5 🡺 compared to 3 in the file, this is a difference of 2 🡺 now: Score = -5

6th guess: 8 🡺 compared to 9 in the file, this is a difference of 1 🡺 now: Score = -4

7th guess: 2 🡺 compared to 1 in the file, this is a difference of 1 🡺 now: Score = -3

8th guess: 7 🡺 compared to 7 in the file, this is a perfect guess 🡺 now: Score = -13

9th guess: 3 🡺 compared to 6 in the file, this is a difference of 3 🡺 now: Score = -10

10th guess: 1 🡺 compared to 8 in the file, this is a difference of 7 🡺 now: Score = -3

Of course, the above schematic description of the game is not yet a complete pseudocode: it is missing variables and structures. We ask that you complete the above list of steps in order to make it a complete, unambiguous pseudocode.

Then, move your pseudocode to Java code.

**Extra-credit:** You will be given 10 points of extra credit if you implement this program using a for loop.

**Instructions:**

1. Complete the provided above set of steps to obtain a complete pseudocode of the solution to the number guessing game.
2. Complete the provided java code.

IMPORTANT: do make sure that your code is properly indented.

Failing to indent your code properly will result in 10 points off

1. Explain what the piece of code “**throws** IOException” means. Tell us where you found this information.
2. Explain what a PrintWriter is. Describe 4 fours that it can be used. Tell us where you found this information.

**Now, here is what you have to turn in:**

1. A docx file in which you have the completed pseudocode, as well the answers to questions 3 and 4 above.
2. The java file with the completed code for method called *numberGuessingGame*.

**Activity 2.**  In this activity, you will get to practice again on writing in and reading from files. You will also start working with Strings. You will have to implement the solution code of the exercise that is described below:

|  |
| --- |
| **The Price of an Article**  As an editor of the CS1401 Magazine, you are also in charge of charging authors who want their articles published. The price of an article depends on the number of words and the number of lines. Computing the price of an article can be daunting…  As a computer scientist, your job is to implement a piece of software that allows you to do just this. More specifically, you should design a program that:   * Asks the user for the name of file you want to check (the article you need to check should be in this file) * Reads in this file and keep track of the number of words and the number of lines * Compute the price of the article as follows:   Price = numberOfLines \* $10 + numberOfWords \* $0.25   * Display the price |

**Instructions:**

1. Complete the provided above set of steps to obtain a complete pseudocode of the solution to the article pricing problem.
2. Complete the provided java code.

IMPORTANT: do make sure that your code is properly indented.

Failing to indent your code properly will result in 10 points off

**Now, here is what you have to turn in:**

1. The docx file with completed pseudocode
2. The java file with the completed program articlePricing.

**That’s it! Looking forward to seeing you in lab!**