**Name:**

**CST 183 Lab Assignment #4 (Chapter 4)**

**Loops and Files**

**Lab Objectives**

Be able to convert an algorithm using control structures into Java

Be able to write a while loop

Be able to write an do-while loop

Be able to write a for loop

Be able to use the Random class to generate random numbers.

Be able to use file streams for I/O

Be able to write a loop that reads until end of file

Be able to implement an accumulator and a counter

**Introduction**

This is a simulation of rolling dice. Actual results approach theory only when the sample size is large. So we will need to repeat rolling the dice a large number of times (we will use 10,000). The theoretical probability of rolling doubles of a specific number is 1 out of 36 or approximately 278 out of 10,000 times that you roll the pair of dice. Since this is a simulation, the numbers will vary a little each time you run it.

Check out how to use the random number generator (introduced in section 4.11 of the text) to get a number between 1 and 6 to create the simulation.

We will continue to use control structures that we have already learned, while exploring control structures used for repetition. We shall also continue our work with algorithms, translating a given algorithm to java in order to complete our program. We will start with a while loop, then use the same program, changing the while loop to a do-while loop, and then a for loop.

We will be introduced to file input and output. We will read a file of floating values, line by line. We will then use the numbers to calculate the mean and capture other important information about the dataset. We will then generate program output that is first displayed on the screen and captured in an output data file.

**Task #1 While loop**

1. Copy the file DiceSimulation.java from the Student CD or as directed by your instructor. DiceSimulation.java is incomplete. Since there is a large part of the program missing, the output will be incorrect if you run DiceSimulation.java.
2. I have declared all the variables. You need to add code to simulate rolling the dice and keeping track of the doubles. Convert the algorithm below to Java and place it in the main method after the variable declarations, but before the output statements. You will be using several control structures: a **while** loop and an if-else-if statement nested inside another if statement. Use the indenting of the algorithm to help you decide what is included in the loop, what is included in the if statement, and what is included in the nested if-else-if statement.
3. To “roll” the dice, use the nextInt method of the random number generator to generate an integer from 1 to 6.

*Repeat while the number of dice rolls are less than the number of times the dice should be rolled.*

*Get the value of the first die by “rolling” the first die*

*Get the value of the second die by “rolling” the second die*

*If the value of the first die is the same as the value of the second die If value of first die is 1*

*Increment the number of times snake eyes were rolled Else if value of the first die is 2*

*Increment the number of times twos were rolled Else if value of the first die is 3*

*Increment the number of times threes were rolled Else if value of the first die is 4*

*Increment the number of times fours were rolled Else if value of the first die is 5*

*Increment the number of times fives were rolled Else if value of the first die is 6*

*Increment the number of times sixes were rolled Increment the number of times the dice were rolled*

1. Compile and run. You should get numbers that are somewhat close to 278 for each of the different pairs of doubles. Run it several times. You should get different results than the first time, but again it should be somewhat close to 278.

**Paste in the source code of your DiceSimulation.java program written with the while loop here:**

**Task #2 Using Other Types of Loops**

1. Change the while loop to a **do-while** loop. Compile and run. You should get the same results.

**Paste in the source code of your DiceSimulation.java program written with the do-while loop.**

1. Change the do loop to a **for** loop. Compile and run. You should get the same results.

**Paste in the source code of your DiceSimulation.java program written with the for loop.**

**Paste in a screenshot of your DiceSimulation.java program output here:**

**Task #3 Reading Input from a File**

1. You are going to create a program from scratch called FileStats.java.
2. Copy the file Numbers.txt from the Student CD or as directed by your instructor. This file is going to be the data file used by your FileStats.java program. It contains a list of floating point values that will be used to calculate the mean of the dataset and identify max and min numbers.
3. Open the file for reading by using a scanner object linked to a file object. (Huge Hint: The code to do this is discussed on pages 242 – 244 in your text.)
4. Use a loop of your choice to read the values from the file. To make sure your loop is working correctly, as you read the values output them to the screen. Once this works go on to the next step.
5. The body of the loop needs to do the following:
	1. Track and store the minimum value.
	2. Track and store the maximum value.
	3. Accumulate a total by adding the read values together.
	4. Keep a count of the number of values read.
6. After exiting the loop, you will use the accumulated value and the item count to calculate the mean of the dataset.
7. You will output the number of values read, the mean, the minimum value, and the maximum value. Make it look nice.

**Task #4 Write Output to a File**

1. Now you need to take the output of FileStats.java and write the output to a file: (Hint: See pages 230 – 235 in your text)
	1. Create a FileWriter object passing it the filename “Results.txt” (Don’t forget the needed import statement).
	2. Create a PrintWriter object passing it the FileWriter object.
	3. Since you are using a FileWriter object, add a throws clause to the main method header.
	4. Output the number of items, the mean, the min, and the max to the output file using a three decimal format, labeling each. Make it look nice.
	5. Close the output file.
2. Compile, run, correct/modify as needed.

**Paste in the source code of your FileStats.java program here:**

**Paste in a screenshot of your output data file results.txt here:**