Chapter 5 Lab
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 the Dice class to see how the random number generator (introduced in section 4.13 of the text) works 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, line by line, converting each line into a number. We will then use the numbers to calculate the mean and standard deviation.

First we will learn how to use file output to get results printed to a file. Next we will use file input to read the numbers from a file and calculate the mean. Finally, we will see that when the file is closed, and then reopened, we will start reading from the top of the file again so that we can calculate the standard deviation.

Task #1 While loop

1. Copy the files Dice.java (see code listing 5.1) and DiceSimulation.java (see code listing 5.2) from the Student CD or as directed by your instructor. Make sure to

place them both in the same directory. You can compile both programs. Dice.java is complete and will not be modified in this lab, but 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. You will be modifying the DiceSimulation class **only**. I have declared all the variables. You need to add what the method does. 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.

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

Roll the first die
Get the value of the first die
Roll the second die
Get the value of 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

3. 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.

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

Task #3 Reading and Writing Using Files

1. Copy the files FileStats.java (see code listing 5.3) and Numbers.txt from the Student CD or as directed by your instructor. You can compile FileStats.java. It will compile without errors so that you can use it to test out the StatsDemo class you will be creating.
2. Create a class called StatsDemo which consists of a main method to do the following:
   a) Create a DecimalFormat object so that we can format our numbers for output with 3 decimal places (Don't forget the needed import statement).
   b) Create a Scanner object to get the file name input from the user (Don't forget the needed import statement).
   c) Prompt the user and read in the file name (Remember to declare any needed variables).
   d) Create a FileStats object passing it the file name.
   e) Create a FileWriter object passing it the filename "Results.txt" (Don't forget the needed import statement).
   f) Create a PrintWriter object passing it the FileWriter object.
   g) Since you are using a FileWriter object, add a throws clause to the main method header.
   h) Print the mean and standard deviation to the output file using a three decimal format, labeling each.
   i) Close the output file.
3. Compile, debug, and run. You should get no output to the console, but running the program will create a file called Results.txt with your output. The output you should get at this point is: mean = 0.000, standard deviation = 0.000. This is not the correct mean or standard deviation for the data, but we will fix this in the next tasks.

Task #4 The calculateMean Method

1. Open FileStats.java for editing. You will notice that the calculateMean and calculateStdDev methods do not do any calculations yet. They simply return a 0 to the constructor to initialize the instance variables. We need to add lines to each of these methods to have them return the correct value. Let's work with the calculateMean method first.
2. Create a FileReader object passing it the filename (Don't forget the needed import statement).
3. Create a BufferedReader object passing it the FileReader object.
4. Since you are using a FileReader object, add a throws clause to the calculateMean method header as well as the constructor method header (since it calls the calculateMean method).
5. Declare local variables for an accumulator of type double, a counter of type integer, and line of type String. Initialize all number variables to 0.
6. Write a priming read to read the first line of the file.

7. Write a loop that continues until you are at the end of the file.
8. The body of the loop will
   a) convert the line into a double value and add the value to the accumulator
   b) increment the counter
   c) read a new line from the file
9. When the program exits the loop close the input file.
10. Calculate and return the mean instead of 0. The mean is calculated by dividing the accumulator by the counter.
11. Compile, debug, and run. You should now get a mean of 77.444, but the standard deviation will still be 0.000.

Task #5 The calculateStdDev Method

1. Do steps 2-7 as above in the calculateMean method but add another local variable called difference of type double.
2. The body of the loop will
   a) convert the line into a double value and subtract the mean, store the result in difference
   b) add the square of the difference to the accumulator
   c) increment the counter
   d) read a new line from the file
3. When the program exits the loop close the input file.
4. Calculate and return the standard deviation instead of 0. The standard deviation is calculated by taking the square root of the accumulator divided by the counter (Use Math.sqrt ( ) to take the square root).
5. When the program exits the loop, the variance is calculated by dividing the accumulator (sum of the squares of the difference) by the counter.
6. Compile, debug, and run. You should get a mean of 77.444 and standard deviation of 10.021.

Code Listing 5.1 (Dice.java)

```
// This class simulates a fair, standard die that is used in many games.

import java.util.Random;                    // to use random number generator

public class Dice
{
        private int spots;              // the number of spots up on the die
        private static Random generator;// a random number generator used in
                                        // simulating rolling a dice, shared by all dice
                                        // so that it will be as random as possible.

        //Constructor creates a single die, initially with no spots
        public Dice()
```

```
        {
           generator = new Random();      //creates an instance of the random
           spots = 0;
        }

        //simulates rolling the die and stores the number rolled
        public void roll()
        {
                spots = generator.nextInt(6) + 1;   //returns 1,2,3,4,5,or 6
        }

        //returns the value of the die
        public int getSpots()
        {
           return spots;
        }
}
```

Code Listing 5.2 (DiceSimulation.java)

```
// This class simulates rolling a pair of dice 10,000 times and
// counts the number of times doubles of are rolled for each different
// pair of doubles.

public class DiceSimulation
{
        public static void main(String[] args)
        {
                final int NUMBER = 10000;//the number of times to roll the dice

                Dice die1 = new Dice(); // the first die
                Dice die2 = new Dice(); // the second die
                int die1Value;          // number of spots on the first die
                int die2Value;          // number of spots on the second die
                int count = 0;          // number of times the dice were rolled
                int snakeEyes = 0;   // number of times snake eyes is rolled
                int twos = 0;           // number of times double two is rolled
                int threes = 0;         // number of times double three is rolled
                int fours = 0;          // number of times double four is rolled
                int fives = 0;          // number of times double five is rolled
                int sixes = 0;          // number of times double six is rolled

                //ENTER YOUR CODE FOR THE ALGORITHM HERE

                System.out.println ("You rolled snake eyes " + snakeEyes +
                        " out of " + count + " rolls.");
```

```
            System.out.println ("You rolled double twos " + twos +
                  " out of " + count + " rolls.");
            System.out.println ("You rolled double threes " + threes +
                  " out of " + count + " rolls.");
            System.out.println ("You rolled double fours " + fours +
                  " out of " + count + " rolls.");
            System.out.println ("You rolled double fives " + fives +
                  " out of " + count + " rolls.");
            System.out.println ("You rolled double sixes " + sixes +
                  " out of " + count + " rolls.");
      }
}
```

Code Listing 5.3 (FileStats.java)

```
//  To calculate the statistics on a file of numbers

public class FileStats
{
      private double mean;        //the arithmetic average
      private double stdDev;  //the standard deviation

      //Constructor calls calculateMean and calculateStdDev methods
      //and store the results in the respective instance variables
      public FileStats(String filename)
      {
            mean = calculateMean(filename);
            stdDev = calculateStdDev(filename);
      }

      //returns the mean
      public double getMean()
      {
            return mean;
      }

      //returns the standard deviation
      public double getStdDev()
      {
            return stdDev;
      }

      //returns the calculated arithmetic average
      public double calculateMean(String filename)
      {
            //ADD LINES FOR TASK 4
```

```
            return 0;
        }

        //returns the calculated standard deviation
        public double calculateStdDev(String filename)
        {
            //ADD LINES FOR TASK 5
            return 0;
        }
}
```