Anwar Adam

Alan Palayil

**CS 115 Fall 2019 Lab #9**

Due: **Monday, November 18th, 5:00 PM**

Points: **20**

**Instructions:**

1. Use this document template to report your answers and create separate java files for your classes. Enter all lab partner names at the top of first page.

2. You don’t need to finish your lab work during the corresponding lab session.

3. ZIP your Java files and lab report into a single file. Name the file as follows:

LastName\_FirstName\_CS115\_Lab9\_Report.zip

4. Submit the final document to Blackboard Assignments section before the due date. No late submissions will be accepted.

5. ALL lab partners need to submit a report, even if it is the same document.

**Objectives:**

1. (6 points) Demonstrate the ability to use while loops.

2. (7 points) Design, code and test a user-defined object containing an array.

3. (7 points) Demonstrate the ability to read a text file using loops and Java Scanner class.

**Problem 1 [6 points]:**

Write a program to play the game of craps (without the betting). In this game, two die are first rolled (**use the** Die **class provided below for that purpose**) to determine the roller's target. If you roll a sum of 7 on the target roll, you win. Otherwise, the roller keeps rolling the two die to try to match the target sum before he rolls a sum of 7 (and craps out).

|  |
| --- |
| **Java** Die **class**: |
| public class Die {  private int side;    public Die()  {  setSide(1);  }    public Die(int newSide)  {  setSide(newSide);  }    public int getSide()  { return side; }    public void setSide(int newSide)  { side=newSide; }    public void roll()  {  side = (int)(Math.random()\*6+1);  }    public String toString( )  {  return "Die=" + side;  }  } |

Sample Plays:

|  |
| --- |
| **Sample plays**: |
| Target Roll: 6 + 1 = 7  You won on the first roll!  Press any key to continue . . .    Target Roll: 2 + 3 = 5  You rolled: 1 + 2 = 3  You rolled: 4 + 6 = 10  You rolled: 3 + 3 = 6  You rolled: 2 + 1 = 3  You rolled: 4 + 4 = 8  You rolled: 6 + 2 = 8  You rolled: 1 + 2 = 3  You rolled: 6 + 6 = 12  You rolled: 5 + 6 = 11  You rolled: 3 + 5 = 8  You rolled: 2 + 5 = 7  CRAPS! You lost.  Press any key to continue . . .    Target Roll: 4 + 4 = 8  You rolled: 5 + 5 = 10  You rolled: 3 + 6 = 9  You rolled: 1 + 4 = 5  You rolled: 5 + 3 = 8  You rolled your target. You won!  Press any key to continue . . .    Target Roll: 4 + 5 = 9  You rolled: 4 + 4 = 8  You rolled: 5 + 3 = 8  You rolled: 2 + 1 = 3  You rolled: 6 + 2 = 8  You rolled: 6 + 5 = 11  You rolled: 5 + 4 = 9  You rolled your target. You won!  Press any key to continue . . . |

Your task is to:

n Create a Craps class (containing your "main" program) from the description given above.

n Compile and run your program to see if it runs (no run-time errors),

**Problem 2 [7 points]:**

Write a DailySales class to store a collection of a company's daily sales records for a single month (up to 31 days). **All daily sales are of integer data type**. You **do not have to worry about verifying the number of days actually in the month**. Please code a class with the following methods:

n DailySales() - default constructor,

n DailySales (int daysInMonth) - constructor,

n public boolean addSales(int dayNumber, int sales) - add "sales" to the current sales for "dayNumber". Return true of successful, else return false (if invalid sales amount or invalid dayNumber),

n public int maxDay() - return the day number with the maximum sales,

n public int[] daysBelowGoal() - return an array of day numbers that have less than 100 units sold,

Your tasks:

n Provide basic design of your program (define the fields/attributes and methods, include data type and valid ranges for attributes, and access control, arguments and return types for methods) in the box below **[1 out of 7 points]**:

|  |
| --- |
| **Program design**: |
| The fields for the design of the program are:  DailySales() : this is the main class of the program to store the company’s daily sales record for a single month (up to 31 days)  DailySales (int daysInMonth) : this field represents the number of days in a month using the data type int  public boolean addSales(int dayNumber, int sales) : The return type will be in boolean “true” or “false”  The return types for the fields, methods, classes will be in int and boolean.  The range of the Dailysales will be from 0 - infinity and anything below 100 will False in boolean.  The range of the daysInMonth will be from 1-31 |

n Complete the Test Plan table below (the number of test cases is up to you, but should be fairly exhaustive) **[1 out of 7 points]**:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test plan** | | | |
| **Test case** | **Sample data** | **Expected result** | **Verified?** |
| Valid Day  Valid Sales | Day = 3  Sales = 373 | Array[0] = 373 | Yes |
| Invalid Day  Valid Sales | Day = -2  Sales = 172 | Out of Bounds | Yes |
| Invalid Day  InvalidValidSales | Day = -100  Sales = -100 | Out of Bounds | Yes |
| Valid Day  Valid Sales | Day = 25  Sales = 21 | Array[27] = 21 | Yes |
| Invalid Day  Valid Sales | Day = -10  Sales = 15 | Out of Bounds | Yes |
| Invalid Day  Invalid Sales | Day = -100  Sales = 102 | Out of Bounds | Yes |
| Valid Day  Valid Sales  Max Day | Array[0]-Array[28] = 10  Array[29]=35  Array[30]=132 | Max Day Returned = 132 | Yes |
| Valid Day  Valid Sales  daysBelowGoal | Array[0]-Array[28] = 10  Array[29]=35  Array[30]=1132 | daysBelowGoal[0] = 30 | Yes |

n Code your program **[5 out of 7 points]**.

**Problem 3 [7 points]:**

Finding duplicate data in a sorted file is the first step to removing duplicates. Given an input file with each line representing a record of data and the first token (word) being the key that the file is sorted on, we want to load it and output the line number and record for any duplicate keys we encounter. Remember we are assuming the file is sorted by the key and we want to output to the screen the records (and line numbers) with duplicate keys.

Download input1.txt and input2.txt files from Blackboard (see Lab files section). Note: the input text files must be in the same directory as your program. Use Scanner class to load them.

|  |
| --- |
| **Sample run** |
| Enter File Name: input1.txt  FileName:input1.txt  DUPLICATES  12 102380 CS US W 2.8 3.267 125  14 102395 PPCI US W 2.769 2.5 115  25 102567 PPCI US W 3.192 3.412 112  35 102912 CS US Z 3.81 3.667 88  44 103087 CS US Z 2.956 2.688 90  76 103944 CS US W 3.134 3.294 134  77 103944 CS US W 3.698 3.7 94  86 104046 CS US W 2.863 3.133 65  88 104047 CS US W 3.523 3.524 77  89 104047 CS US O 3.825 3.824 49  91 104048 CS US W 3.071 3 94  92 104048 CS US W 3.114 3.111 44  93 104048 CS US W 3.375 3.6 71    Press any key to continue . . . |

Your task is to

n create a FindDuplicates class with the following:

n Declaration of an instance variables for the String filename

n non-default Constructor - creates an object for user passed filename argument

n Accessor methods return the value of each instance variable

n Mutator methods that allows th user to set each instance variable (no validation required),

n a "getDuplicates()" method that reads from the file (until end-of-file) using Scanner class, finds duplicate records based on the first token on each line (the key), and returns as a String the record number and entire duplicate record one to a line (see above Sample output)

n toString() - returns a String message with the value of the instance variable

n Create a FindDuplicatesApp driver class / program to test your FindDuplicates class.

n Compile and run your program to see if it runs (no run-time errors).

**Note: use Java exceptions!**