Week6Lab – 10 pts

Pre-lab questions

1. Arrays and ArrayLists are similar, but not the same. The differences are important to know
   1. Declare an Array of Strings and an ArrayList of Strings. Highlight the differences.

String[] strArray = **new** String[3]; / /Declaration by specifying the size.

ArrayList<String> list=new ArrayList<String>();// Declaration of arraylist

* 1. Add one element to each of the previously declared collections. Highlight the differences.

strArray[0] = "one"; // adding string one to zero location in array strarray because array start from 0 location.

strArray[1] = "two";

strArray[2] = "three";

list.add("one"); // adding one string to arraylist using method add.

list.add("two");

list.add("three");

* 1. Write a loop to print out each element in the array. What, if anything, would have to be different to do the same task for the ArrayList?

**for** (**int** i = 0; i < strArray.length; i++) {

System.***out***.println(strArray[i]);

}

**for** (String print : list) {

System.***out***.println(print);

}

**for** (**int** i = 0; i < list.size(); i++) {

System.***out***.println(list.get(i));

}

* 1. What happens if you try to add one more element than will fit? Is the result different for the two types of collections?

In array like a define the size to 3 and my array is full now after adding three elements when I will add one more element it will goes to out of bound exception error because it has defined the size already if I have to add one more element, I have to change the size of array.

For arraylist it is easy process to add new element using add method and list.size help to print the list using for loop.

1. Write a method that performs a statistical operation on a set of integers. You do not know how many integers will be passed into the method.

**public** **int** max(**int**[] array) {

**int** max = 0;

**for** (**int** i : array) {

**if** (i > max) {

max = i;

}

}

**return** max;

}

**public** **int** min(**int**[] array) {

**int** min = array[0];

**for** (**int** i : array) {

**if** (i < min) {

min = i;

}

}

**return** min;

}

Choose one of the following to develop into a program that an array. Do not use an ArrayList! Once chosen, do the following:

Understand the problem (restate in your own words, make any assumptions clear):

I understand the problem where I have to create 3 arrays using random library means random numbers. two are one dimensional and one is two dimensional. First array is for 0 and 1. Second is for dice which is 1 to 6. And third is for two dice. Then take average of all the arrays. And print the result.  
  
UML diagrams of any classes needed, including the one with main:  
A screenshot of a phone

Description automatically generated with medium confidence

Pseudocode of any non-trivial methods in each class (no pseudocode needed for basic setters and getters or no args constructors):

RandomArray

In this program I have create three methods which takes array as parameter. Two methods takes array as parameter and third method take two dimensional array as parameter. In that method I have calculated the average and print the random generate array list.

RandomArrayTester

In this program there is main method where I call the methods of RandomArray class and pass the arguments using object of that class while calling the methods.  
  
Name of files (.java) submitted:

RandomArray.java RandomArrayTester.java  
  
White box test plan and results:

I have created the main method in Tester file where I create and test all the methods which I have created in randomarray class.

1

0

1

1

1

0

0

0

0

0

0

1

0

1

1

0

0

0

1

0

1

0

1

0

1

1

0

1

0

1

0

0

0

1

1

0

0

0

0

1

0

0

0

0

0

1

0

1

1

1

1

0

1

0

0

1

1

0

0

1

1

1

0

0

1

0

0

1

0

0

1

0

1

1

0

0

1

0

0

0

1

0

1

0

1

1

1

0

0

0

1

1

0

1

0

1

1

0

1

0

Average of first array: 02

1

4

5

2

2

6

5

2

5

4

1

2

6

4

2

4

3

1

2

6

4

6

3

3

3

5

6

2

5

6

5

3

6

5

5

5

6

1

3

3

1

2

5

4

1

5

2

4

4

4

2

1

3

3

1

5

1

6

1

1

4

6

4

1

1

2

1

5

6

2

4

6

3

1

3

3

6

5

2

3

3

5

3

4

3

4

6

6

1

4

1

3

5

2

1

3

3

4

2

Average of second array: 36

5

5

9

6

7

10

9

5

7

2

6

5

5

5

6

8

3

4

4

10

6

9

9

4

8

2

10

9

3

9

8

3

9

10

8

4

7

12

10

8

5

7

9

10

8

6

6

8

3

9

6

9

10

9

5

4

3

11

7

9

7

9

3

3

3

6

6

6

6

3

10

10

9

3

6

4

7

10

2

10

5

6

6

2

10

9

7

9

5

8

6

5

7

5

6

9

10

8

2

Average of third array: 6

2: \*\*\*\*\*

3: \*\*\*\*\*\*\*\*\*\*

4: \*\*\*\*\*\*

5: \*\*\*\*\*\*\*\*\*\*\*\*

6: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

7: \*\*\*\*\*\*\*\*\*

8: \*\*\*\*\*\*\*\*\*

9: \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

10: \*\*\*\*\*\*\*\*\*\*\*\*\*

11: \*

12: \*

Trail Data

Write a program that contains a list of elevation data for specific hiking trails. The elevation measurements are taken every 10 meters and indicate via positive or negative numbers the difference in elevation from the previous location. The values for one trail should be randomly generated in the range of -5 to 5 for the 100 measures taken of the 1KM trail. The values for a second trail should be entered as all 1’s to the midpoint and all -1’s for the second half (use a loop). Create an interesting (but not too hard) trail by manually entering the data in the range of -5 to 5. For each of the trails, find the elevation difference from start to end. The second one should be 0 as half the trail goes up by one then the other half goes down by one.

Find a way to print the data in a way that is visually informative but not the numbers themselves.

Sales Receipt

Think about the grocery store and how long your receipt can get! Write a program that allows for a receipt for 100 items. Populate the array with random doubles in a range you find typical for grocery purchases. Allow for some ‘small’ negative numbers to represent coupons or deals. Calculate the sum, the sum of the positive numbers and the sum of the negative numbers. Calculate 3% sales tax on the positive numbers only. Print a receipt that includes each value in the array, the positive total, the most expensive item, the negative total (with a you saved type statement), the tax, and the total due to be paid to the store. Test it with random data, only positive numbers and with a small receipt of numbers you enter.

How Random is Random?

Write a program that will test the random number generator in several ways. Create an array to hold 100 coin flips represented as 0 or 1. Populate it with numbers generated randomly with Random class. Create another array to hold 100 rolls of a d6 (6 sided dice) also populated with randomly generated numbers in the range of 1-6. Create one more array to hold 100 results of rolling 2d6 (2 dice). Roll each die separately then add the results and store them in the array.

Calculate and print the average value for each array.

Count the number of each value in the two dice arrays then create a histogram with asterisks (\*) to represent how many times that number was generated. If you need a hint, see example code (<https://courses.cs.washington.edu/courses/cse142/10sp/lectures/5-19/~programs/Histogram.java> )