Week6Lab – 10 pts

Pre-lab questions

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

```
ArrayList <String> strArrayList  = new ArrayList();
String [] strArray = new String [2];
```

The Array can only hold a fixed number of strings, whereas an ArrayList can hold any number of strings

   –

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

```
strArrayList.add("s1");
strArray[0] = "s1";
```

   i. The add method will append a new value to the end of a list. With an array, you must predefine the index and it will modify the location with that value
   ii. The ArrayList will grow and have a new size of 1, whereas the Array size will not change

   c. 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 (String str : strArrayList) {
    System.out.println(str);
}

for (String str : strArray) {

    System.out.println(str);

}
```

- You can use these loops to print all values for both the array and the ArrayList, and the arrayList would print normally
- In the array, all the references will be printed as well regardless of whether they are null or not (happens if the size is greater than the number of elements that were populated into it)
- To solve this, you would need to implement a check to see if the value is not null and only then print the element

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

The arrayList will grow to fit that value, whereas the array would throw an ArrayIndexOutOfBounds exception

2. 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 static int max(int ... data){
    int maxVal = 0;
    for (int currVal : data){
        if (currVal > maxVal) {
            maxVal = currVal;
        }
    }
    return maxVal;
}
```

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):

UML diagrams of any classes needed, including the one with main:

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

Name of files (.java) submitted:

White box test plan and results:

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.

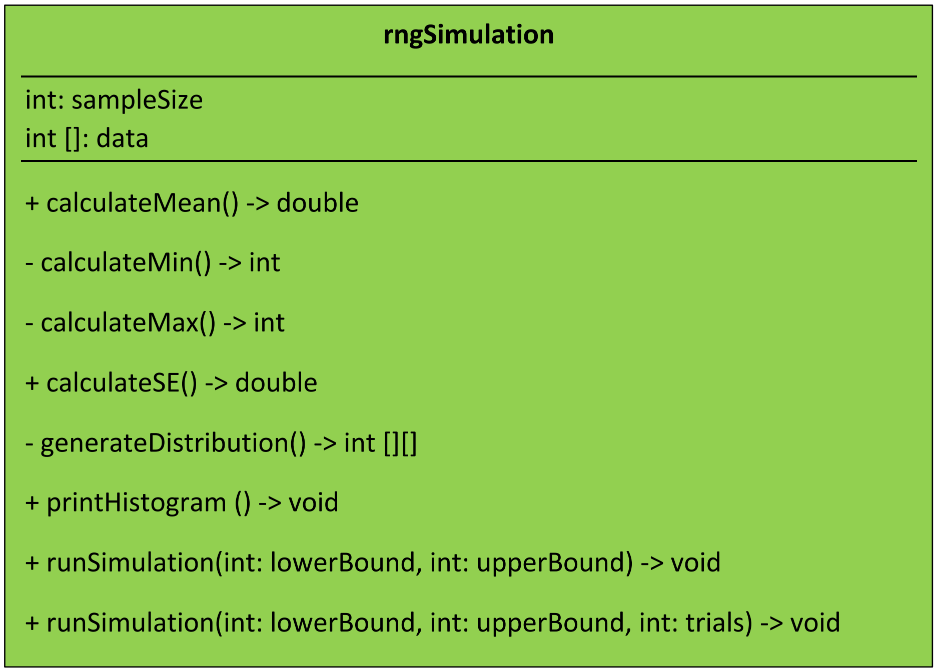
How Random is Random?

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)

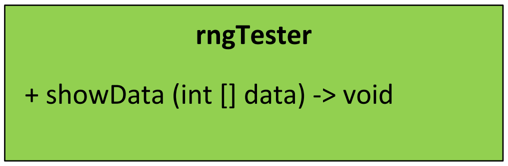
We need to randomly generate 100-coin flips (0 and 1 values), 100 rolls from 1 6-sided dice (values 1-6), and 100 sums from 2 independent 6-sided dice rolls (values 2-12). Then find the averages of the data generated and print a histogram of the counts.



rngSimulation

int: sampleSize
int []: data

+ calculateMean() -> double
- calculateMin() -> int
- calculateMax() -> int
+ calculateSE() -> double
- generateDistribution() -> int [][]
+ printHistogram () -> void
+ runSimulation(int: lowerBound, int: upperBound) -> void
+ runSimulation(int: lowerBound, int: upperBound, int: trials) -> void



rngTester

+ showData (int [] data) -> void

```
def  runSimulation():
    for i in 1…100
        int trial = randomUniform(lowerbound, upperbound);
        data[i] = trial;
```

```
public void runSimulation(int lowerBound, int upperBound, int n){

  // Runs a simulation where the sum of multiple trials is added to the results

  for i in 1…100
```

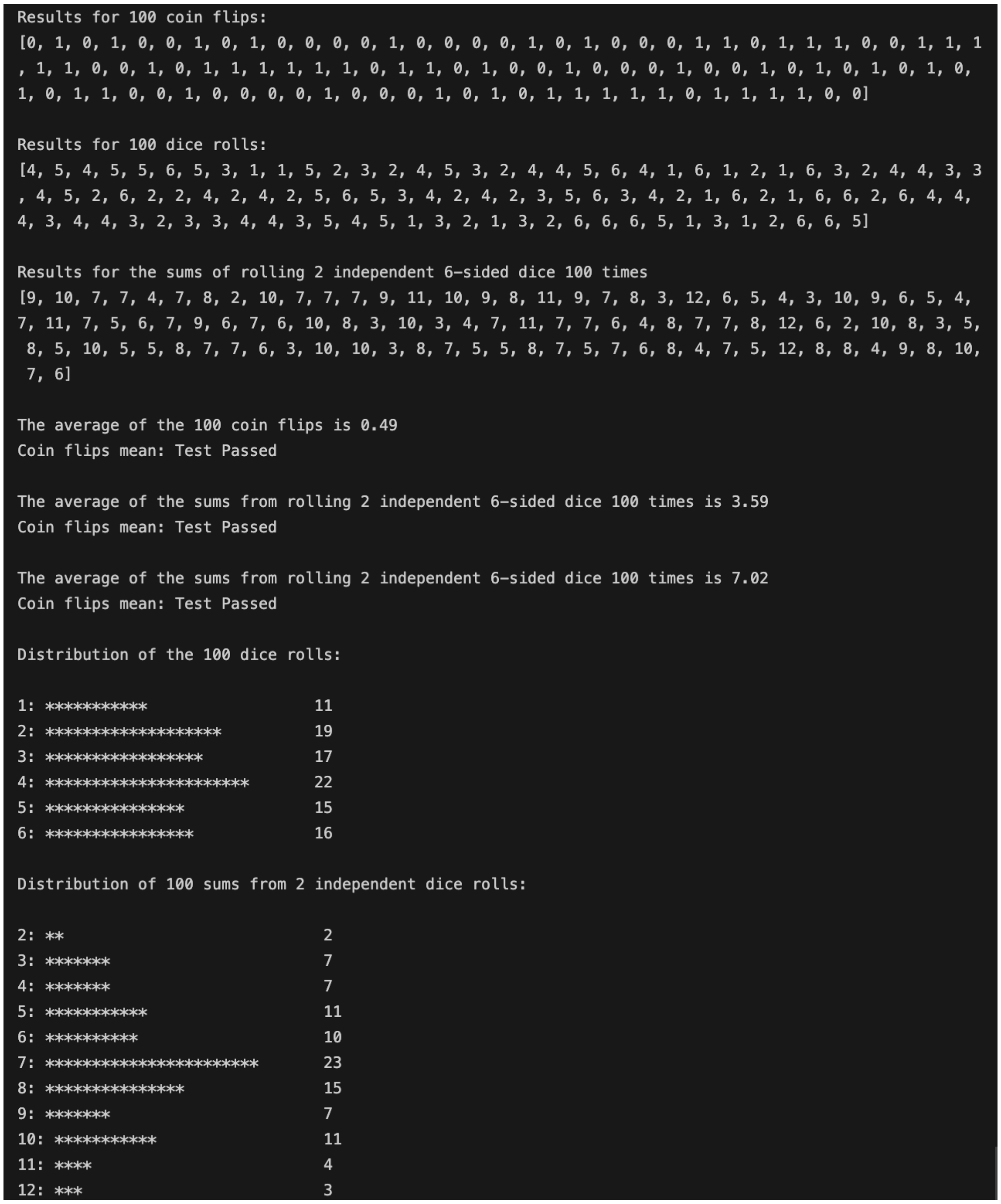
```
      int sum = 0;
      for i in 1…n:
        sum += randomUniform(lowerbound, upperbound)
      data[i] = sum


def calculateMean():
        total = 0.0;
        for num in data:
                total += num
        return total / length(data);

def calculateSE()”
    mean = this.calculateMean();
    double sqDeviation = 0.0;
    for num in data
      sqDeviation += (num – mean)^2

    double variance = sqDeviation / (len(data) - 1);
    return sqrt(variance / data.length);



def calculateMax():
  currValue = 0
  for number in data:
        if number > currValue:
                currValue = number
    return currValue


def calculateMin(int [] data):
    currValue = max(data)
    for number in data:
            if number < currValue:
                    currValue = number
    return currValue




def generateDistribution(int [] data){

        // This determines the proper range of the random variable so that they can be stored by index
       min = this.calculateMin();
       max = this.calculateMax();
        distribution = new int[max - min + 1][2];

        for num in data:
            distribution[num-min][0] = num; // Store all possible values
            distribution[num-min][1]++ // Count the values
        return distribution




    def printHistogram (int [][] distribution):
        for row in distribution:
            int count = row[1]
            int possibleValue = row[0]
    if count > 0: {
       val = possibleValue + ": "
      print val
       for (int j = 0; j < count; j++) {
          print "*"
       print count
       println

def showData(int [] data)

// Displaying the list of results to see if the simulation is was successful
    String ls = "[";
    int i = 0;
    for int num in data
       if i != data.length – 1 :
          ls += num+", "

       else:
         ls += num
       i++

    ls += "]"
    println ls
```

White Box test plan:

To test how close the means are to the expected value, I plan to use a 2-tailed Z-test with a confidence level of 95% (other 5% in the tail of the standard normal distribution). If the absolute value of the Z-statistic is less than or equal to the critical value at 95%, then we cannot reject the null hypothesis and conclude that the sample mean is a reasonable estimate. Otherwise, the data is statistically extreme, which can happen in very rare cases due to a small sample size. Increasing the sample size would allow for closer results.

Histogram will be tested by inspecting the asterisks to see if they match the counts, and counts will be tested by inspecting if they sum to 100.

See rngTester.java for white box tests!



```
Results for 100 coin flips:
[0, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1
, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 1, 0,
1, 0, 1, 1, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 0, 0]

Results for 100 dice rolls:
[4, 5, 4, 5, 5, 6, 5, 3, 1, 1, 5, 2, 3, 2, 4, 5, 3, 2, 4, 4, 5, 6, 4, 1, 6, 1, 2, 1, 6, 3, 2, 4, 4, 3, 3
, 4, 5, 2, 6, 2, 2, 4, 2, 4, 2, 5, 6, 5, 3, 4, 2, 4, 2, 3, 5, 6, 3, 4, 2, 1, 6, 2, 1, 6, 6, 2, 6, 4, 4,
4, 3, 4, 4, 3, 2, 3, 3, 4, 4, 3, 5, 4, 5, 1, 3, 2, 1, 3, 2, 6, 6, 6, 5, 1, 3, 1, 2, 6, 6, 5]

Results for the sums of rolling 2 independent 6-sided dice 100 times
[9, 10, 7, 7, 4, 7, 8, 2, 10, 7, 7, 7, 9, 11, 10, 9, 8, 11, 9, 7, 8, 3, 12, 6, 5, 4, 3, 10, 9, 6, 5, 4,
7, 11, 7, 5, 6, 7, 9, 6, 7, 6, 10, 8, 3, 10, 3, 4, 7, 11, 7, 7, 6, 4, 8, 7, 7, 8, 12, 6, 2, 10, 8, 3, 5,
 8, 5, 10, 5, 5, 8, 7, 7, 6, 3, 10, 10, 3, 8, 7, 5, 5, 8, 7, 5, 7, 6, 8, 4, 7, 5, 12, 8, 8, 4, 9, 8, 10,
 7, 6]

The average of the 100 coin flips is 0.49
Coin flips mean: Test Passed

The average of the sums from rolling 2 independent 6-sided dice 100 times is 3.59
Coin flips mean: Test Passed

The average of the sums from rolling 2 independent 6-sided dice 100 times is 7.02
Coin flips mean: Test Passed

Distribution of the 100 dice rolls:

1: ***********                  11
2: *******************          19
3: *****************            17
4: **********************       22
5: ***************              15
6: ****************             16

Distribution of 100 sums from 2 independent dice rolls:

2: **                            2
3: *******                       7
4: *******                       7
5: ***********                   11
6: **********                    10
7: ***********************       23
8: ***************               15
9: *******                       7
10: ***********                  11
11: ****                         4
12: ***                          3
```