**Lab 21 – Simple Histogram**

Open BlueJ, and create a new BlueJ project titled **Lab21-SimpleHistogram** in your CS\LABS folder.

Create a new class, open it (double click), delete BlueJ's starter code, and **type** in our code skeleton:

//Name:

**import java.util.\*;**

public class PracticeProblems

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

**int[] nums = {2, 4, 0, 8, 6, 2, 3, 5, 9, 0, 9, 6};**

}

}

An array is a single variable (of particular **type**) that stores a **list** of values. The way we access / change values is by referring to the **index** (or position) of the value in the array. We use ‘square bracket notation’ to refer to elements in an array. For an array called *nums*, the value of the element in the first position would be *nums[0]* (Java, like most languages, starts index numbers at 0). Arrays can be confusing, but they are too useful not to use!

**Before each problem, insert a COMMENT with the problem number.**

1. Note the red bolded addition to our skeleton code above. Add this variable to your program – the variable nums will be used to test the code you write in the following problems. Print the **value** of the first element in nums to the console (the element at index 0).
2. Print the value of the element in nums at **index 1**.
3. (Riddle) Can you divide the numbers from 1 to 9 (1 2 3 4 5 6 7 8 9) into two groups so that the sum of the numbers of each group is equal? Note - 9 cannot be turned over to make it 6.
4. Print “true” if there is a 6 in the 5th spot (index 4) of numsor“false” if there is not.
5. Ask the user to guess what number is in the 6th spot (index 5) of nums. Make them keep guessing until they get it right.
6. Change the value of the element at index 3 to be 50. Now print that element to the screen.
7. Double the value of the element at index 7. Now print that element to the screen.
8. Change the value of the 3rd element so that it is ten more than the 1st element.
9. Type this line of code to see the current values stored in your array. (I’ll tell you more about it later.)

System.out.println(Arrays.toString(nums));

1. Declare a new array of doubles called otherNums and give it a size of 4. (Let Java automatically assign default values.) Type this line of code to ensure that you see the default values:

System.out.println(Arrays.toString(otherNums));

1. One at a time, change the values in the array to be 1.2, 3.4, 5.6, and 7.8. Type this line of code to ensure that you set all the values correctly.

System.out.println(Arrays.toString(otherNums));

**Simple Histogram app**

Create a new class with this code skeleton:

//Name:

import java.util.\*;

public class SimpleHistogram

{

public static void main(String[] args)

{

Scanner console = new Scanner(System.in);

}

}

A histogram represents the distribution of numerical data - essentially a count of the number of occurrences of a particular piece of data. Write a program that creates a basic “number histogram”.

**First, create an integer array with 10 elements.** Those elements will be used to count how many times certain numbers appear. We are going to generate some random numbers. If a 7 is randomly generated, we will increase our count of how many sevens have come up!

Ask the user how many random numbers should be generated. **Use a for loop to handle generating all the numbers.** The user can request as many as they want, but the random numbers should always be from 0 to 9.

Since this is random, the array should contain different values every time your program runs.

Each time a new random number is generated display it to the screen and count it in your array!

If you are making variables for each value (e.g. int num0 = 0, num1 = 0, num2 = 0, etc.), you’re doing it wrong!

Hint: you don’t have to store all the different random numbers. You just have to store the counts!

**A great, efficient solution doesn’t require any if statements!**

After all the random numbers have been generated and tallied, print out your data! Input/Output should look like this:

How many single digit numbers should I generate? >> 5

Simulation 1: 1

Simulation 2: 7

Simulation 3: 1

Simulation 4: 2

Simulation 5: 8

\*\* Here's the final count \*\*

0s found: 0

1s found: 2

2s found: 1

3s found: 0

4s found: 0

5s found: 0

6s found: 0

7s found: 1

8s found: 1

9s found: 0