**Lab – Multidimensional Arrays**

**Animation of Multidimensional Programming Activity**

**Lab Objectives**

 Be able to declare two-dimensional arrays

 Be able to pass two-dimensional arrays to methods

 Be able to return a two-dimensional array from methods

 Be able to process a two-dimensional array in a Java Method

 Be able to work with individual rows and columns in a two-dimensional array

**Introduction**

Writing methods that take multi-dimensional arrays as parameters and/or return multi-dimensional arrays is similar to working with single-dimensional arrays.

The syntax for a method that accepts a two-dimensional array as a parameter is as follows:

**returnType methodName( arrayType[][] arrayParameterName )**

The syntax for a method that returns a two-dimensional array is the following:

**returnArrayType [][] methodName( arrayType [ ][ ] arrayParameterName )**

In this Lab, you will work with a 4-row, 20-column, two-dimensional array of integers. You will write methods that perform the following operations:

1. Fill the 2-D array with random numbers between 50 and 80.
2. Print the array.
3. Set every array element of a given row to a specified value. The value is a parameter of a method.
4. Find the minimum value in a given column of the array. The column is a parameter of a method.
5. Count the number of elements of the array having a specified value. The value is a parameter of a method

To visualize how the two-dimensional array is working, your result will animate a bar graph. The

Code for the bar graph is provided to you as **BarChart.java**. So, the framework for this lab will animate your algorithm, which will support you in checking the accuracy of your Java program.

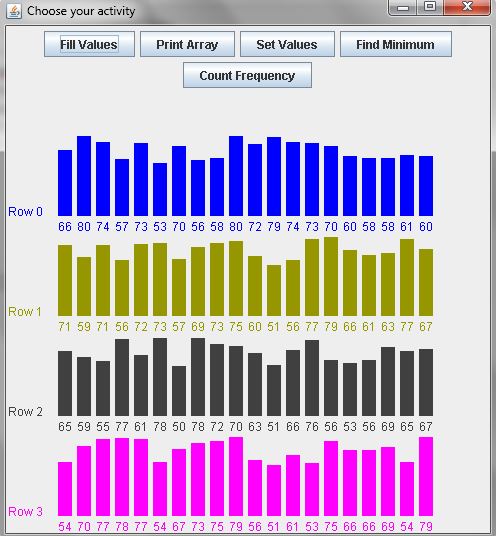
**Lab Instructions**

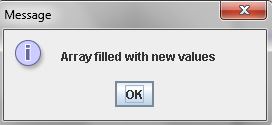
The source files required to complete this activity has been provided. Copy both the files to a folder in your computer. It is important that both the files stay in the same folder.

Open the TwoDimArrayPractice.java source file. For your convenience, the sample code for task number 1 has been provided under comment 1. inside the **fillValues() method**. You may use this as a model for completing the remaining four tasks. You will need to complete those rest of the four tasks in four other methods. In each one of these four remaining tasks, you will fill in the code for a method that will manipulate an existing array of 4 rows and 20 columns. You need not instantiate the two-dimensional array. That has been done for you.

**SAMPLE OUTPUT:**

Following shows a sample output from the **fillValues( ) method**.

****

****

**Task #1 Print Array to Console.**

1. For printing array to console, please make a complete implementation of the method:

**public void printArray( )**

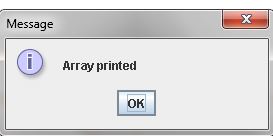
This method is under comment 2.

1. For console printing, elements are separated by a space. The instance variable named **intArray** is the integer array to be printed.
2. To animate the algorithm, put this method call as the last element in your inner for loop  
   **animate( row, column );**  
   Here row is the index of the array's current row, and column is the index of the array's current column.

**OUTPUT:**

Following shows the screen capture of the required output from the **printArray( ) method**.



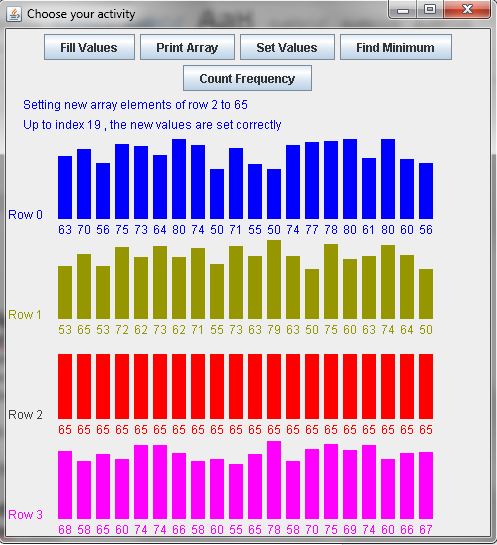


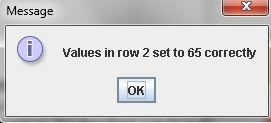
**Task #2 Set All Elements in A Row to A Specified Value**

1. This is under comment 3. Please, implement the **setValues( int value, int row )** method that sets all the elements in the specified row to a specified value.
2. The instance variable named **intArray** is the integer array.
3. To animate the algorithm, put this method call as the last element in your for loop  
    **animate( row, column );**Here row is the index of the array's current row and column is the index of the array's current column.

**OUTPUT:**

Following shows the screen capture of the required output from **setValues(65, 2)**.





**Task #3 Find Minimum Value in the Specified Column**

1. Implement the method **public int findMinimum( int column )** that finds the minimum value in the specified column.
2. Again, the instance variable named **intArray** is the integer array
3. To animate the algorithm, put this method call as the last element in your for loop  
    **animate( row, column );** Here row is the index of the array's current row and column is the index of the array's current column
4. Notice that a dummy return statement (return 0) is provided at the end of **public int findMinimum( int column )** method so that the source code will compile in its present state. In this way, you can write and test each method separately, which is known as the step-wise refinement. When you are ready to write the **public int findMinimum( int column )** method, just replace the dummy return statement with the appropriate return statement for the method.

Following shows screen shot of the output for **findMinimum(3)**.

**OUTPUT:**



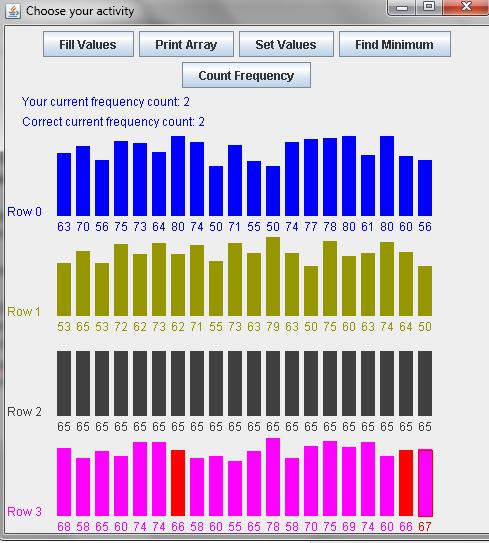


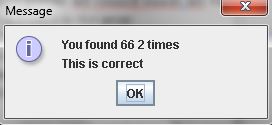
**Task #4 Find Frequency of a Given Value in the Array**

1. Implement the method **public int countFound( int value )** thatcounts the number of times a given value is found in the array.
2. The instance variable named **intArray** is the integer array.
3. To animate the algorithm, put this method call as the last element in your for loop  
    **animate( row, column );** Here row is the index of the array's current row and column is the index of the array's current column
4. Notice that a dummy return statement (return 0) is provided at the end of **public int countFound( int value )** method so that the source code will compile in its present state. In this way, you can write and test each method separately using the step-wise refinement. When you are ready to write the **public int countFound( int value )** method, just replace the dummy return statement with the appropriate return statement for the method.

**OUTPUT:**

Following shows screen capture for **countFound(66)**.





The two .java files (TwoDimArrayPrac.java and BarChart.java) running under the same project will animate your algorithm so that you can visually watch your code work. For this to happen, make sure that your single or nested for loops call the method *animate.* The arguments that you send to *animate* are not always the same and the location of the call to animate will differ depending on the task you are coding.

Since the values in the two-dimensional array are randomly generated, the values will be different each time the program runs. To test any of the methods that you have implemented, please click on the appropriate button.