# 2-Dimensional (2D) Arrays

Arrays in Java are also able to be two-dimensional, or 2D, which means they have multiple rows and multiple columns. Since each row and column must both be indexed for these types of arrays, two indices and two pairs of brackets are needed. The order of the indices in the bracket pairs is row index followed by column index. For example, accessing the row-column position of 0, 1 in an imaginary 2D array, data\_array, would be written as:

Data\_array[0][1]

In the example file, the 2D array “colors” shows how curly brackets can be used to initialize a 2D array.

1. **Given that the 2D array “colors” is 3 by 8, what does this tell you about initializing 1D arrays to non-zero values?**

The colors array defines a separate R, G, and B color value for each light. Because of the array format, the setLightsArray() method in the LightsArray class shows how, in this case, a single simple for loop can be used to navigate through the array and set each light’s RGB color values.

1. **In the for loop in setLightsArray, is it iterating through rows or columns? How can you tell from the for loop statement?**

We could also use 1D arrays to update a row of a 2D array. Let’s start by reorganizing the example file first. Create a 2D integer array private instance variable in LightsArray called colorsMatrix initialized to zeros of size 3 rows and 8 columns.

int[][] colorsMatrix = new int[3][8];

Remove the input from the setLightsArray() method.

1. **Add an Mutator method that takes a 2D integer array and sets colosMatrix equal to it.**

Test your answer from question 3 by calling the Mutator method and passing in the “colors” array. You will also need to remove the passed variable in the setLightsArray() method call now that the definition of it has been changed.

1. **Consider a possible user-caused issue that may happen in using the 2D array Mutator method you created. How would you try to handle the error?**

Now we can focus on what is needed to individually set R, G, or B values from 1D arrays. Create a new public void method in the LightsArray class called setRowColors(). The method should take in two inputs, int[] colorValues and String colorStr. The colorValues will be the 1D array of either R, G, or B values. The colorStr will tell the method which row to use by way of “R”, “G”, or “B”.

Next, create an if-elseif-else if statement to check colorStr as “R”, “G”, or “B”. While we could have the array updating logic in each case, it is easier and shorter to instead declare an integer variable “rowIndex '' before the if-elseif-elseif and initialize it to 0. In each if-elseif-elseif case, set the rowIndex variable to the appropriate number, 0 for R, 1 for G, 2 for B.

1. **Underneath the if-elseif-elseif statement, create a single line to update the correct row of data in colorsMatrix.**
2. **Write a 1D array of new red color values in the main method. Before the first setLightsArray() method call, use the setRowColors() method to update the red color values in the class instance variable colorsMatrix.**