Lab 5: Loops, ArrayLists, and Arrays

Before You Begin!

In this lab, you will be learning about and working with arrays and ArrayLists, two extremely convenient ways of storing data in Java. In order to complete this lab, you will need to do some debugging along the way! The reading, debugging guide, and lectures on this material will be really helpful. If you’re at all confused about the content of those materials, review before beginning!

Coloring Arrays

You’re going to initialize and populate a few javafx.scene.paint.Color arrays of specified dimensions, using different colors to form patterns. You’ll also have to write code to allow you to visualize the arrays you’ve created.

1. Run cs0150_install lab5 to install this lab’s stencil code. Open Eclipse and check out the contents of the lab5 package (refresh Eclipse if you don’t see it). For the first exercise, you’ll be working within the file ArrayBuilder.java. Open it up now.

2. The ArrayBuilder class contains one method called displayArray followed by five methods which will each build a 2D array of colors. Two methods, buildStripeArray and buildGreenArray, have been written for you. Run the program by right-clicking on lab5’s App.java class and choosing “Run as >> Java Application”. Click the Stripes button once the window appears. Nothing shows up!

This is because the buildStripeArray method only initializes/returns an array of colors. It does not actually create Nodes that can be added to a parent Pane, or add those Nodes. Remember - arrays allow you to logically organize objects within your program. They don’t have anything to do with JavaFX graphics.

3. To display this array we need to convert it into an array of rectangles with colors corresponding to the array created in buildStripeArray, and then add these rectangles to a Pane. You will need to complete the displayArray method in order to accomplish this. We have already handled converting an array of Colors into an array of Rectangles for you.

   ○ In the displayArray method, clear all of the children of parentPane to get rid of any previously displayed arrays. Remember, you can get the children of a Pane by calling the getChildren() method. This returns a List object. Not sure how to clear a List? Check out the Javadocs!
   ○ Loop through the array of Rectangles, rects, adding each one to
parentPane. To do this, copy the following code, making sure to fill in the commented portion!

```java
for (int i = 0; i < rects.length; i++) {
    for (int j = 0; j < rects[i].length; j++) {
        // Code to add to the pane goes here!
    }
}
```

4. In order to check that your `displayArray` is working, run the program to check that the stripe array returned by `buildStripeArray` is displayed correctly. Once the window appears, click the “Stripes” button (under the “Arrays” tab). You should see this array:

![Stripe Array](image)

However, there are four (intentional!) bugs in our `buildStripeArray` method. Use your knowledge of arrays and print lines to find and correct these errors, in order to create an array that looks like this: *(Hint: what do you know about the bounds of arrays? Between i and j, which represents rows and which represents columns?)*

![Corrected Stripe Array](image)

5. Next, you will be fixing the `buildGreenArray` method. Click the “green” button in the visualizer (uh oh, there’s an error!). Look through the stack trace to find the source and fix the bug. *(Note: to show line numbers, navigate to Windows > Preferences > General > Editors > Text Editors, and check the box that says “Show Line Numbers”)*

Once this is resolved, you will notice that there are four red squares in your array. Going back to the `buildGreenArray` method, there is a 2D array of ints named “intArr” with hard coded values. This method works by comparing the values of “intArr” to another array we have already filled in in the `Constants` class, displaying a green square if the value matches and a red square if it does not. Your goal is to find what the correct values should be and update “intArr” so that it displays a fully green array.
Note: Initializing an array with the syntax

```java
int[][] myArray = { {1, 2, 3}, {4, 5, 6}, {7, 8, 9} }
```

will produce the following:

```
[1, 2, 3]
[4, 5, 6]
[7, 8, 9]
```

6. Fill in the remaining three stencil methods, `buildCheckerArray`, `buildTwoColorArray`, and `buildDiagonalStripeArray`.
   - In each method:
     i. Create an array of Colors of the dimensions specified in the method comments.
     ii. Next, loop through the array you’ve created, figuring out the appropriate Color at each index and storing it in the array.
     iii. Finally, return the array you’ve created.
   - If you’re having trouble getting the expected results, try adding print statements throughout the method that’s giving you issues, and analyze the method at each statement to make sure that the method is doing what you expect it to be doing! This can help you analyze certain variables!

Expected Results:

![Checkerboard](image)
![Two-color](image)
![Diagonal](image)

7. Run the program and test each method by clicking the corresponding button in the GUI, which will call your method and visualize the array of Colors it returns. The images above show what each array should look like-- when your arrays match them, move on to the next part of the lab.

Check Point 1: Call over a TA to check your arrays!
Practice with ArrayLists

You’re going to fill in a series of methods (“Add White”, “Add Three Pink”, “Change Pinks to Blues”, and “Remove Third”), each of which performs a particular operation on an ArrayList<Color> called _colors. Respectively, these will add one white block to the arraylist, add three pink blocks to the arraylist, change all of the pink blocks in the arraylist to blue ones, and remove the third block (if there is one) from the arraylist. Again, you’ll first write code to visualize the ArrayList!

1. Open up the file ArrayListBuilder.java. The ArrayListBuilder class contains several stencil methods, each of which should perform an operation on the ArrayList<Color> instance variable called _colors.

2. Again you will need to fill in the method displayArrayList before anything will show up on the screen. Clear the parentPane and add all the elements of rects to the parentPane.

3. Run the app and navigate to the “ArrayLists” tab. Each of the buttons at the bottom of the interface calls the corresponding method in the stencil code. Play around with the “Add Pink” and “Clear” buttons (we’ve filled in these methods for you-- the other buttons won’t do anything until you fill in their methods)! If you’ve implemented displayArrayList correctly, you should see the visualization of _colors update as elements are added and removed.

4. Fill in the remaining stencil methods according to the descriptions in the method comments. You can reference the Javadocs for a full description of the ArrayList’s methods. Run the app and play around to test your work.

Check Point 2: Call over a TA to check you off!