CSC 161 LAB 5-2 BINARY SEARCH

PROFESSOR GODFREY MUGANDA

1. Overview of Lab Assignment

You are going to follow instructions to write a program that can be used to demonstrate the step by step operation of the binary search algorithm. The program will display the array being searched in an array of text fields. The program will also, at each step of the demonstration, display the current values of index variables `lower` and `upper` that indicate the portion of the array currently being searched. There will also be text fields to display the mid value for the middle index, and another text field to allow the user to enter a search value.

2. Variables Needed

You will need the following variables

```
final int COLUMN_COUNT = 3;   // Number of columns in text fields
// Text fields to display the array elements
// as well the indices lower, upper, and mid.
// There is also a text field to display the value being
// searched for.
TextField[] numberTfs = new TextField[20];
TextField lowerTf = new TextField(),
          midTf = new TextField(),
          upperTf = new TextField();
TextField searchValueTf = new TextField();

// Button to create a new random array to start a new search,
// and a button to perform the next search step
Button newRandomButton = new Button("New Random Array");
Button nextStepButton = new Button("Search Step");

// The array of numbers,  with lower and upper bounds of
// portion of the array to search.
int [] numbers = new int[numberTfs.length];
int [] bounds = {0, numbers.length-1};
```

Initially, all text fields should be un-editable.

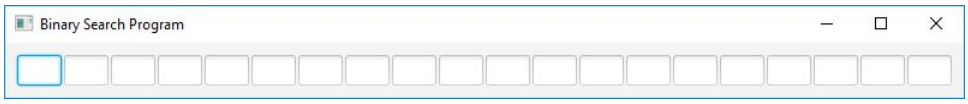
Here we are using an array of two integers

```
int [] bounds = {0, numbers.length-1};
```

to represent the lower and upper bounds of the segment of the array remaining to be searched. This is necessary because lambda expressions (which we will use for the event handlers) cannot access variables that are not effectively final. This will be explained in class.

3. Put an array of TextField at the top of a BorderPane

Use a `BorderPane` for the root of the scene graph. Create a `HBox`, add the `numberTfs` text fields to the `HBox`, using an spacing of 2 pixels. Set some padding around the `BorderPane` to make it look nice. You should now have something like this



4. Add the 4 Text fields to the Center of the frame

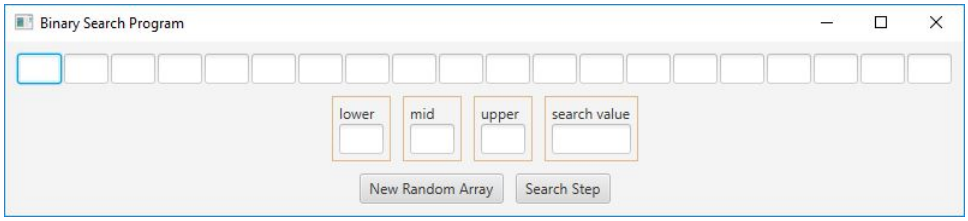
Add the `lower`, `mid`, `upper`, and `search value` text fields to the center of the frame. We want to put descriptive text on each text field so the user knows what it is for. One way to do that is to use a `VBox` that stacks a label on top of the text field and put a border around it. You can do this by adding the following class to the same file:

```
class LabeledTextField extends VBox
{
    public LabeledTextField(String label, TextField tf)
    {
        super.setStyle("-fx-border-color:burlywood");
        super.getChildren().addAll(new Label(label), tf);
        super.setPadding(new Insets(5));
    }
}
```

You can then use instances of this class to wrap each of the four text fields, like so:

5. Add the two Buttons at the bottom of the frame

Add the two buttons at the bottom of the frame so the user interface now looks like this:

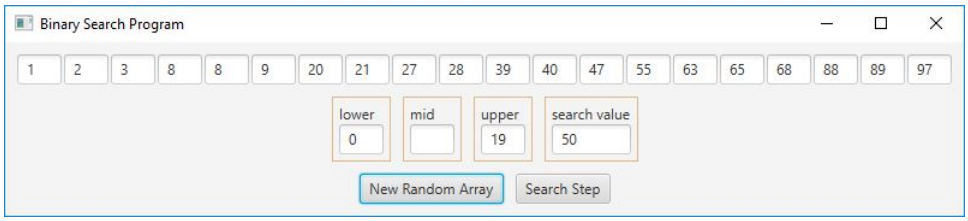


6. Add a handler for the new random array button

The handler for this button will do the following:

(1) fill the array of numbers with randomly generated integers in the range 0..100,
(2) sort the array and display it in the arrays of text fields at the top of the user interface.
(3) set the background colors of all text fields at the top to white.
(4) set `bounds[0]` and `bounds[1]` to point to the lower end and upper end of the array, respectively,
(5) place a search value of 50 in the `search value` text field, and then make that field editable.

Here is what things will look like after the randomization button is clicked.



7. Add a handler for the search value text field

Set an `ActionEvent` handler for the search value text field that is called when ENTER is typed inside the text field. The handler makes the text field un-editable.

8. Add a handler for the search step button

The user can enter a value in the search text field at any time after the new random array button has been clicked.

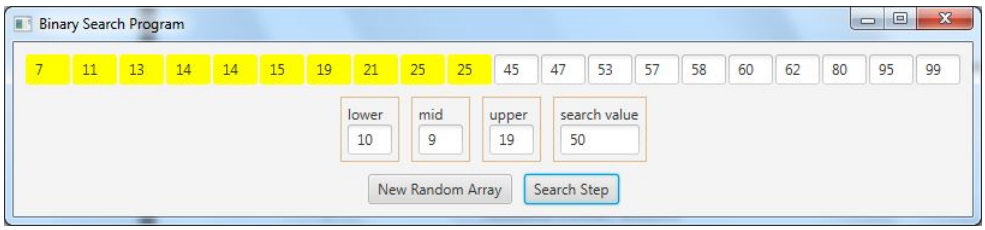
Each time the search step button is clicked, the handler will check to see if there is any segment of the array left to search. If not, the handler displays a `JOptionMessage` saying that the value was not found.

Otherwise, the handler will use the current values of the search boundaries to compute the midpoint. The midpoint will be displayed in the `mid` text field. If the value at the mid point is equal to the search value, the handler display a `JOptionPane` saying the search value has been located at the midpoint. Otherwise,

(1) If the search value is greater than the value at `mid`, then `bound[0]` is set to `mid + 1`; the `lower` text field is updated, and all text fields to the left `mid` are set to a yellow background.
(2) If the search value is less than the value at `mid-1`, then `bound[1]` is set to `mid`; the `upper` text field is updated, and all text fields to the right `mid` are set to a pink background.

Here is a sequence of screen shots showing how the search progresses. Clicking the new Random Button might produce a screen such as this:

At this point, the segment of the array to be searched has a lower bound of 0 and an upper bound of 19, and we are searching for 50. Clicking the search step button will result in this scenario, where the lower half of the array has been ruled out as being to "small" to contain the search value.



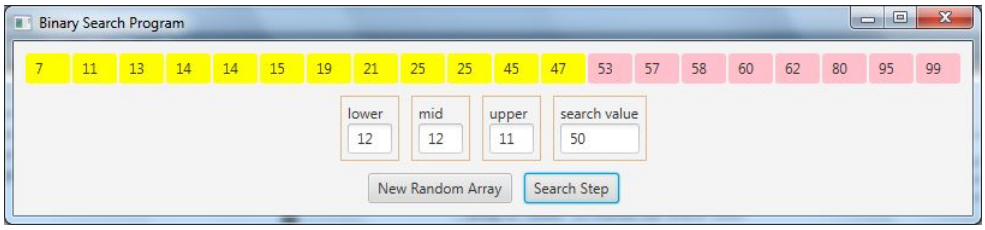
The search for 50 will now focus on the segment from 10 to 19.

Clicking again will result in the scenario below:

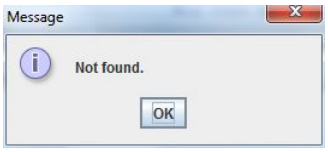
Here the segment from 14 to 19 is ruled out as being too large to contain the search value, and the rest of the search will focus on the segment 10..13.

The next step will present the scenario

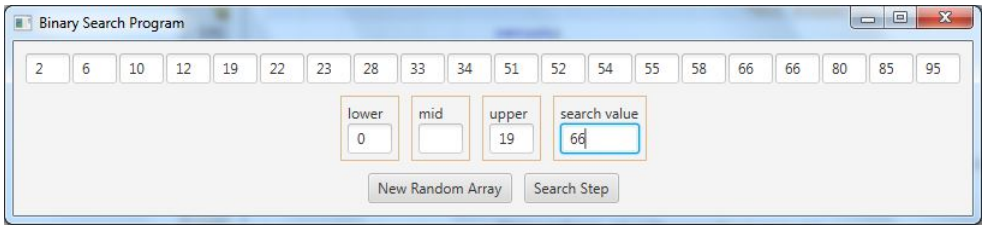
followed by

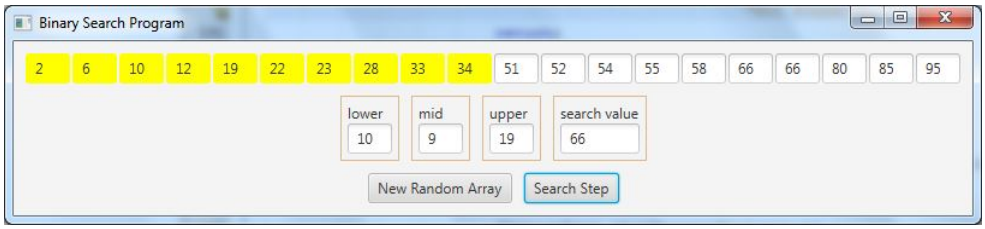


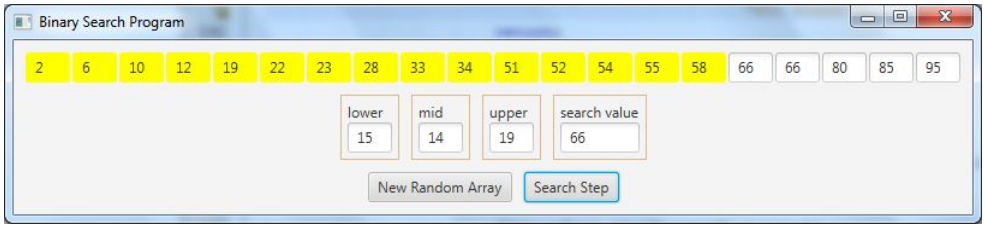
and then eventually

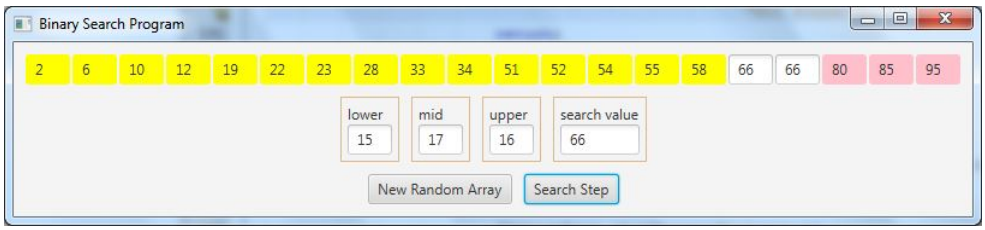


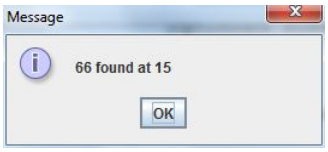
Here is another sequence of scenarios, one that leads to the search value being found.











9. Due Date

Due Monday of Week 6.