CSC 161 LAB 6-1 SORTING

GODFREY MUGANDA

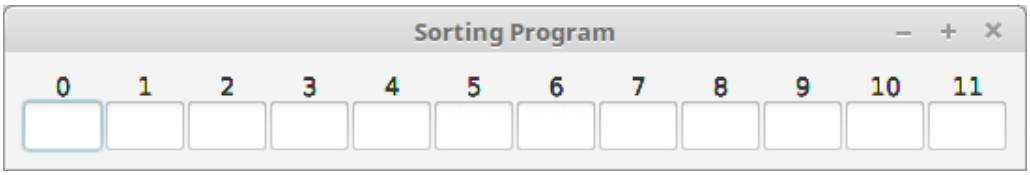
We are going to write a program that allows a user to see how an array sorting algorithm works.

1. Create the Textfields to display the array elements

Begin by a creating a `BorderPane` to use for the root of the scene graph. Next, create a `GridPane` and place in the grid pane an array of 12 text fields and an array of 12 labels:

```
final int SIZE = 12;
TextField[] tFs =  new TextField[SIZE];
Label [] pos = new Label[SIZE];
```

The `pos` labels show the position of each text field within the array. Add the labels and text fields to the `GridPane`, and then put the `GridPane` in the top section of the `BorderPane`, so that the user interface looks like this:

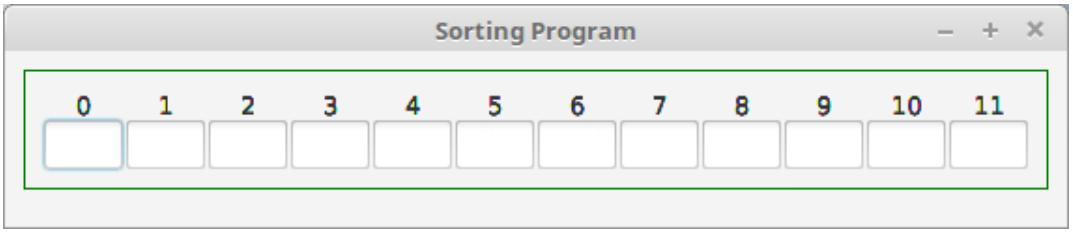


Make all the text fields un-editable and set their preferred column count to 2. You should center the labels in the grid cell. For example, to center the label in column c in the grid, you can use this code

```
GridPane.setHalignment(pos[c], HPos.CENTER);
```

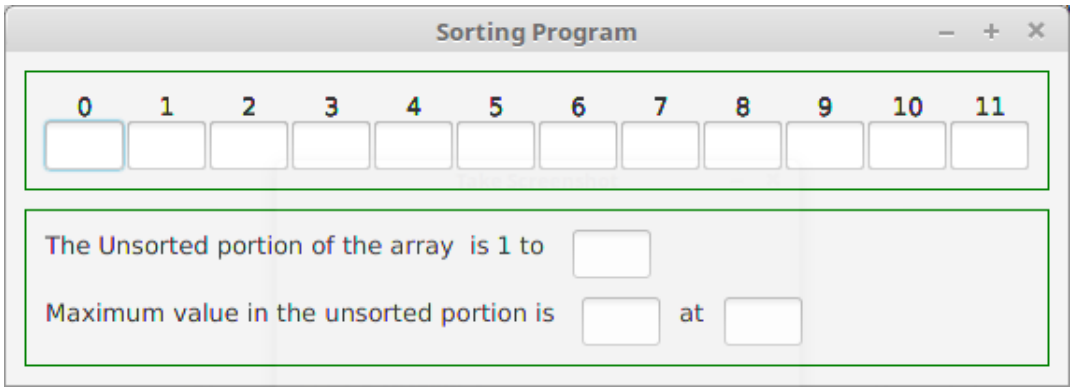
You should put some padding in the top-level `BorderPane` to make the user interface look nice.

Finally, set a border around the grid:



2. Tracking the Progress of the Sorting Process

We want to keep track of the portion of the array that has not yet been sorted, the portion of the array already sorted, and the maximum array element in the unsorted portion of the array. The *sort progress panel* will have some labels and text fields arranged to look like the lower half of the following screenshot:



Here are some of the variables you will need:

```
TextField upperBoundTF = new TextField();
upperBoundTF.setPrefColumnCount(2);
upperBoundTF.setEditable(false);

Label upperBoundLabel =
      new Label("The Unsorted portion of the array  is 1 to " );

TextField maxFoundTF = new TextField();
maxFoundTF.setPrefColumnCount(2);
maxFoundTF.setEditable(false);

Label maxFoundLabel =
      new Label("Maximum value in the unsorted portion is " );
TextField maxFoundPosTF = new TextField();
maxFoundPosTF.setPrefColumnCount(2);
maxFoundPosTF.setEditable(false);
```

The *upper bound* text field will display the upper bound of the portion of the array that remains to be sorted: originally this will be the length of the array minus 1.

The *max found* text field will display the maximum value in the part of the array that is not yet sorted, and the *maximum found position* text field is the position in the array of that maximum value.

To achieve this look, use two `HBox` panes, where one contains a label and the *upper bound* text field and the other contains a couple of labels with the *max Found* and *max Found Pos* text fields.

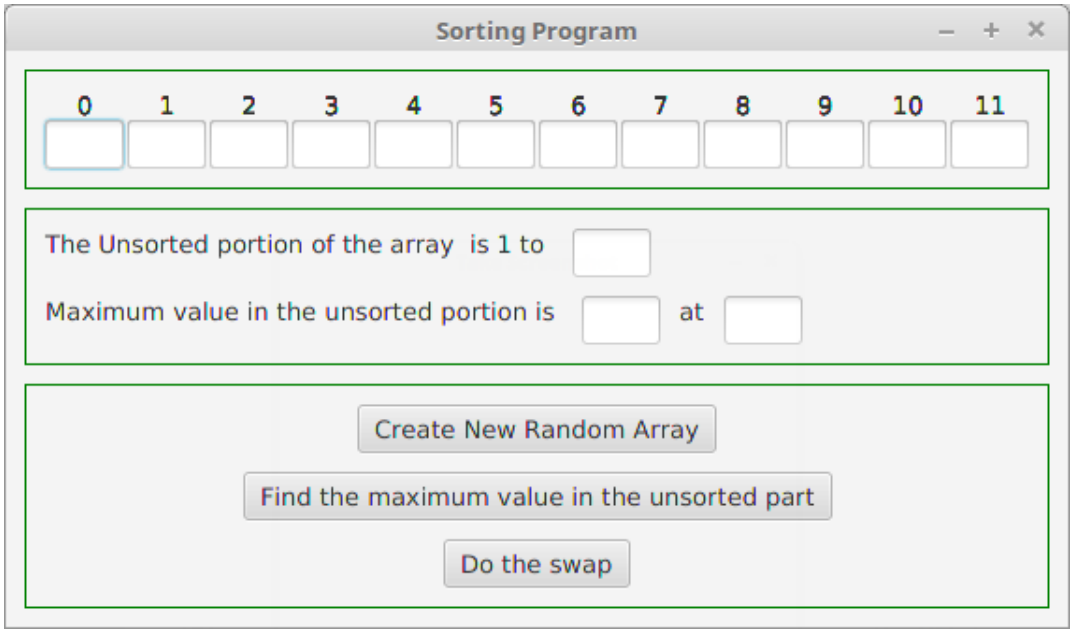
Put all these in a `VBox` and set this `VBox` in the center of the `BorderPane`. You will need a border aroung the `VBox` and some padding.

You will find that this center `VBox` abuts against the grid pane in the top section of the border pane. Create some breathing room by giving the grid pane a bottom margin of 10 pixels whenever it is added to border pane:

```
BorderPane.setMargin(grid, new Insets(0, 0, 10, 0));
```

3. Controlling the Sort Process

Next, create three buttons to control the sorting process in a step-by-step fashion, and add them to a `VBox` that you then add to the border pane. This `VBox` must have a border and appropriate padding, as in the bottom part of the following screenshot:



4. The State of the Sort Process

We need an array of numbers to hold the array to be sorted, and two values

```
int [] numbers = new int [SIZE];
int upperBound;
int maxPosition;
```

The *upper bound* variable is the position of the last entry in the unsorted portion of the array, and the *max position* variable is the position of the maximum value in the unsorted part of the array.

Because these values which are not effectively final, have to be accessible to lambda expressions that implement the event handlers, we must wrap them in an object of some class. Therefore, define a class

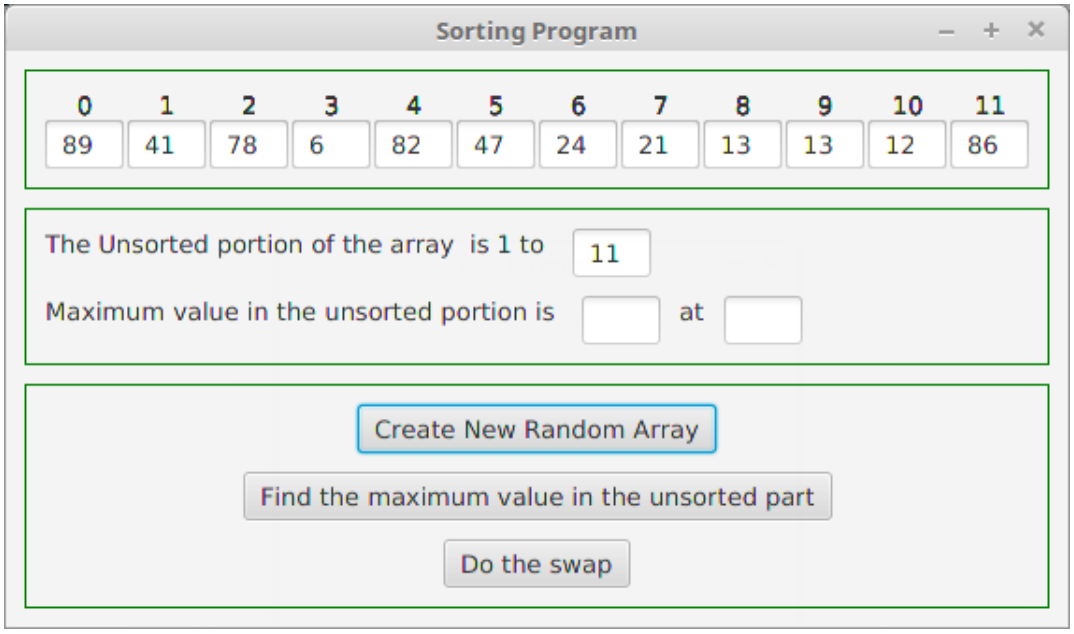
```
class SortState
{
    int maxPosition;
    int upperBound;
}
```

and then define

```
int [] numbers = new int[SIZE];
SortState state = new SortState();
```

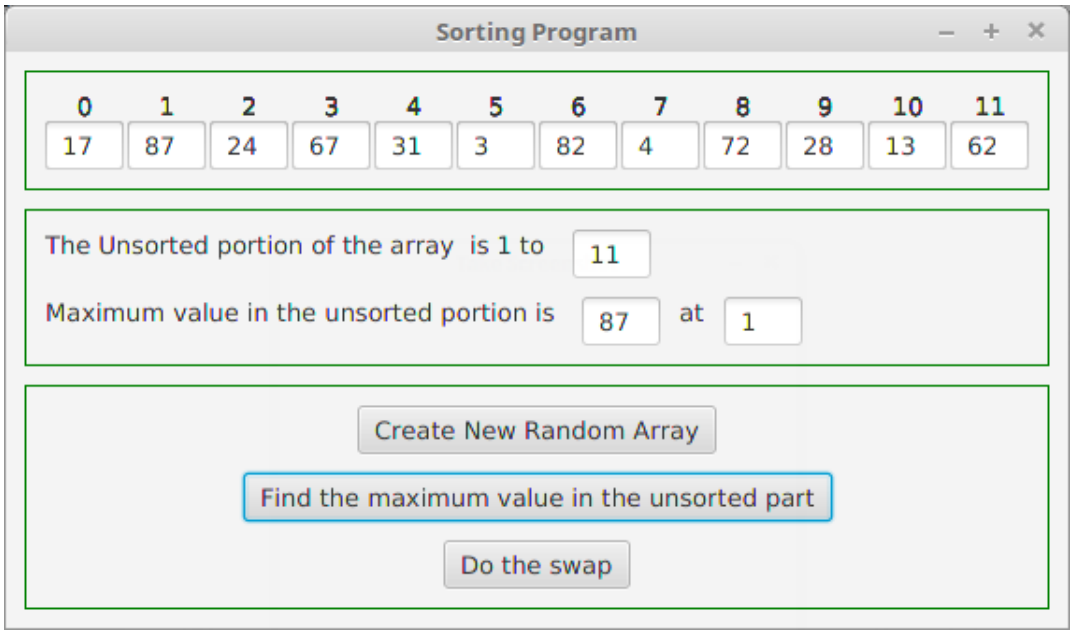
5. Add an Event Handler: New Random Array

Add an event handler for the `ActionEvent` on the *Create New Array* button. This handler will fill the `numbers` array with random integers in the range 0..99 and set the *upper bound* field in the `state` object to the length of the array minus 1. In addition, it will display the array entries in the text fields, set the state object, and display the value of the state *upper-bound member* in the upper bound text field.



6. Add an Event Handler: Maximum Value in Unsorted Part

Next, add an event handler that finds the position of the maximum entry in the unsorted part of the array, which in the range [0..., `state.upperBound`]. This position of this value is stored in `state.maxPosition`, and both maximum value and its position are displayed in the max found and max found position text fields:

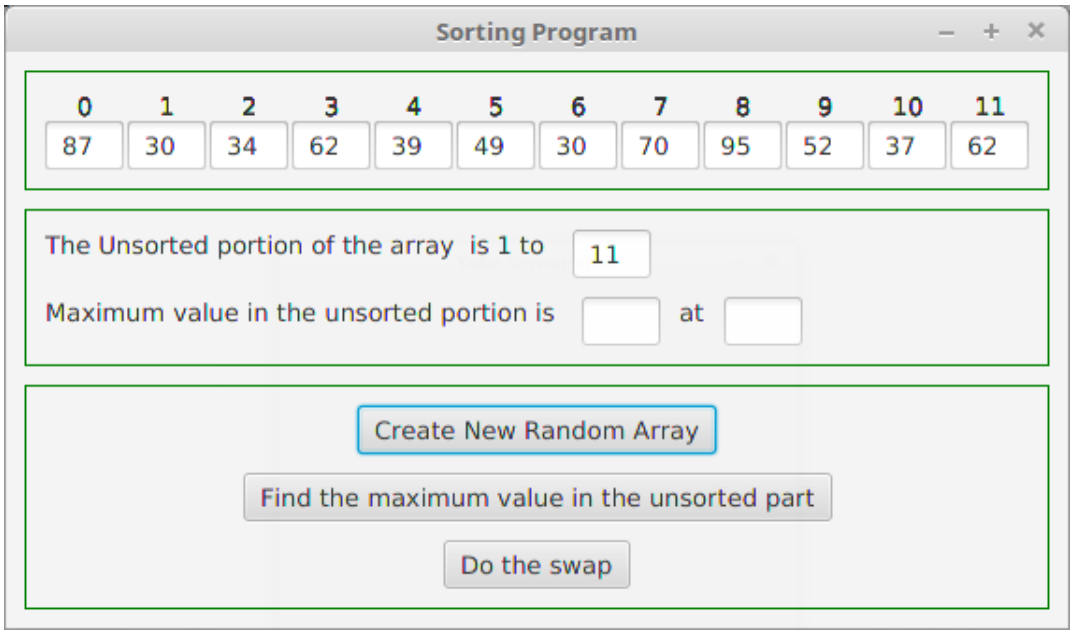


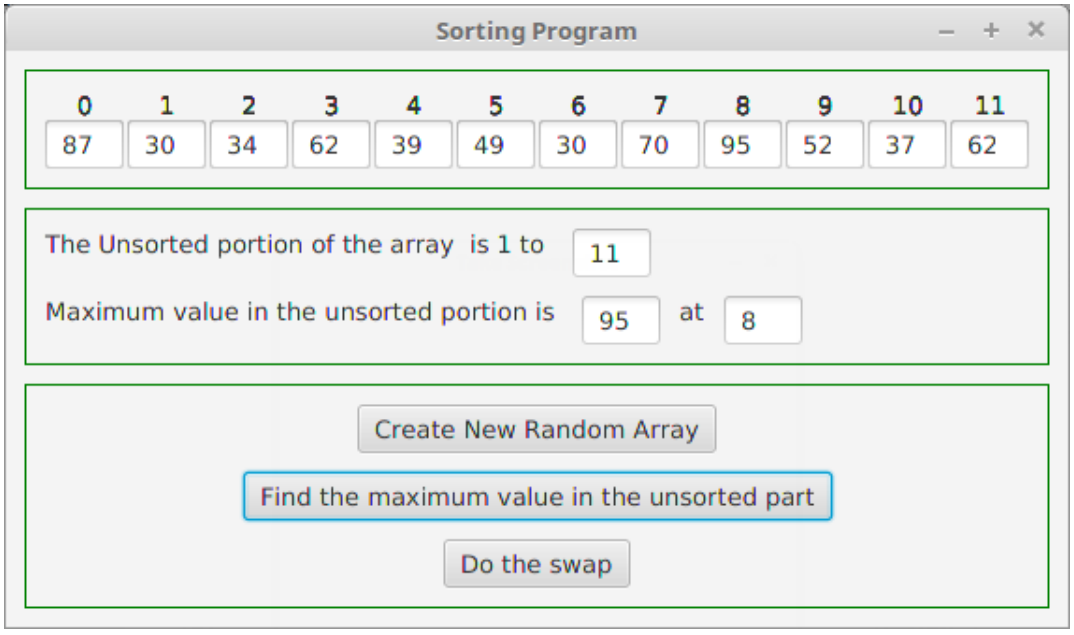
7. Add an Event Handler: The Swap Operation

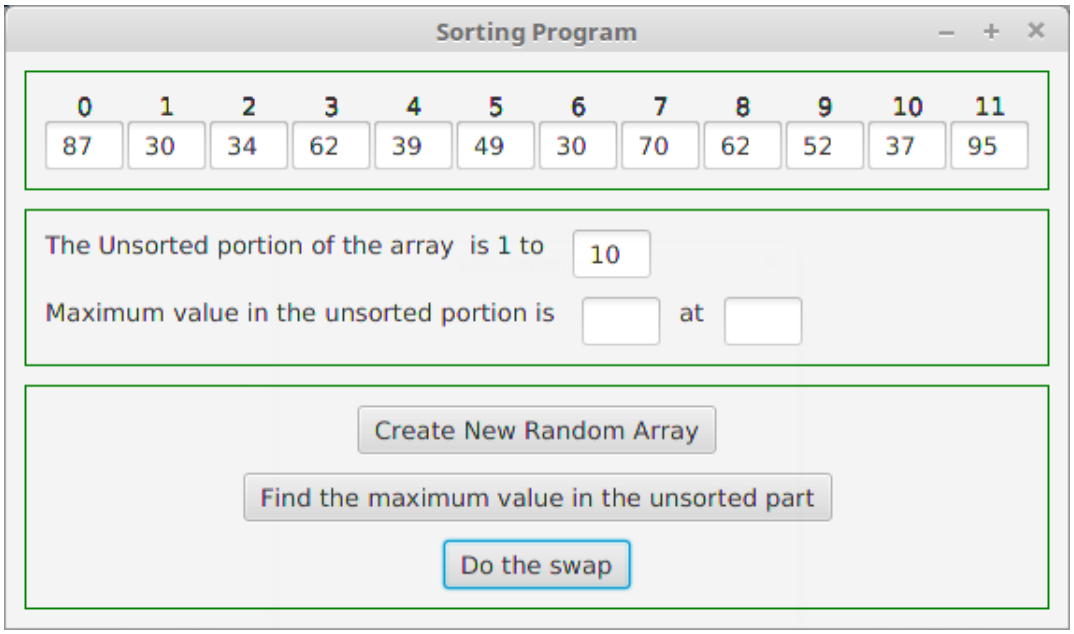
The event handler for the swap button will swap the maximum value in the sorted part of the array with the value at the end of the unsorted part of the array. It will decrement the *upper bound* and then update state variables and the user interface to reflect these changes. In particular, it will update the *upper bound* text field and blank out the *max found* text field and the *max found position* text fields.

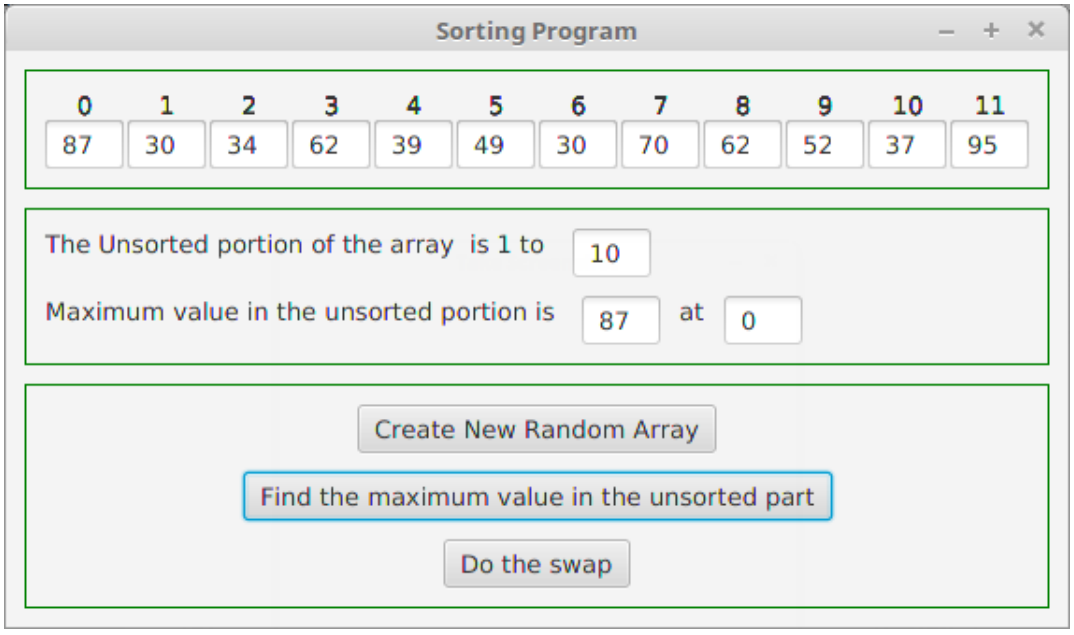
8. The program in operation

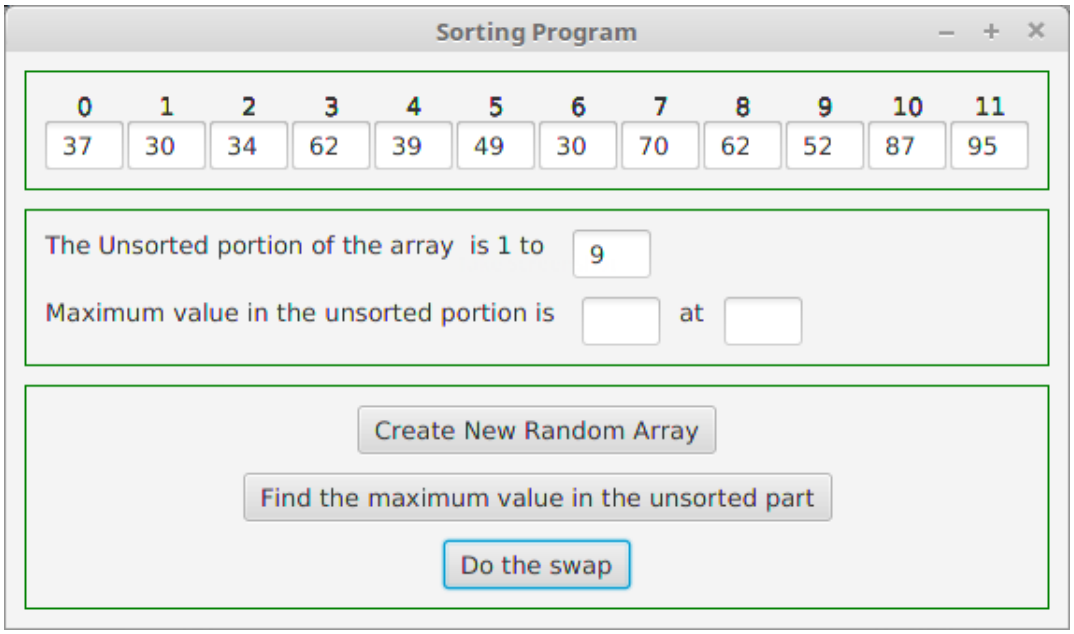
Here is a sequence of screen shots of the program in operation.

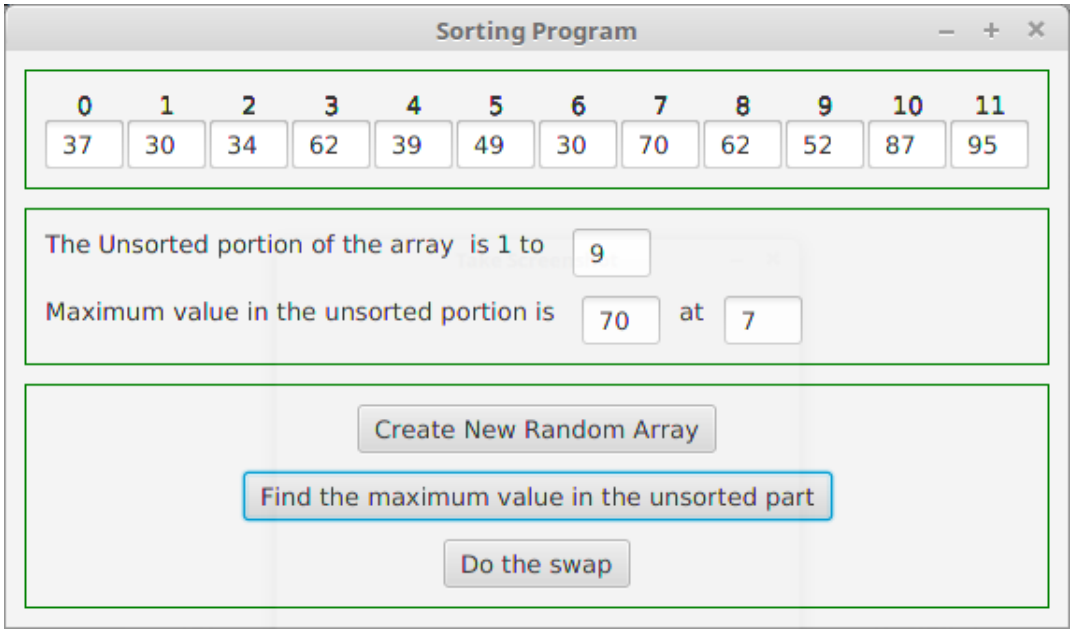

Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
87 30 34 62 39 49 30 70 95 52 37 62
The Unsorted portion of the array is 1 to
11
Maximum value in the unsorted portion is
at
Create New Random Array
Find the maximum value in the unsorted part
Do the swap

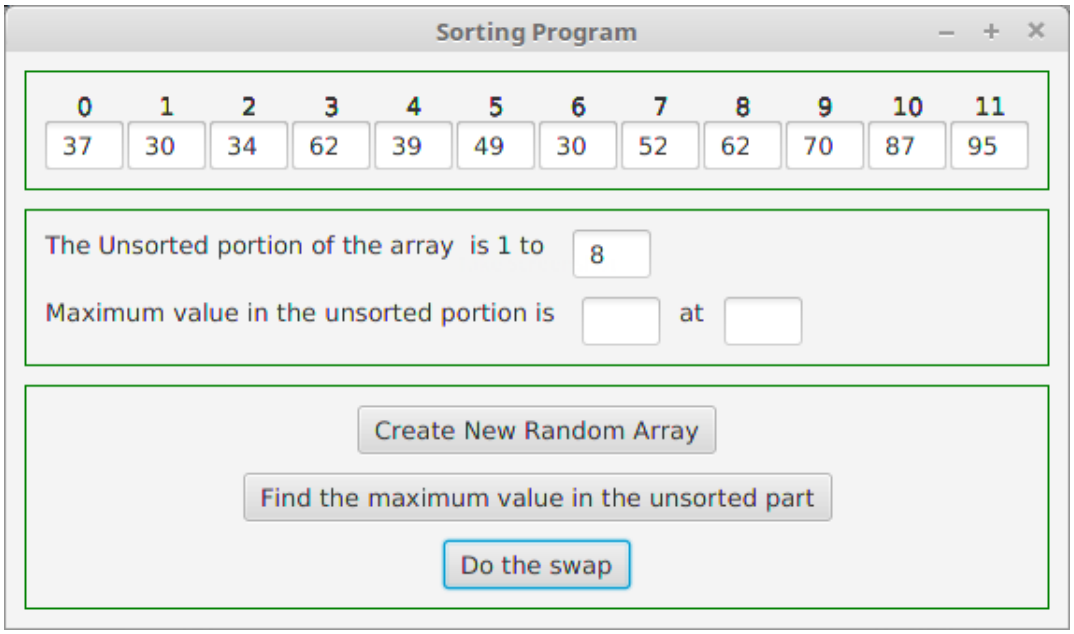

Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
87 30 34 62 39 49 30 70 95 52 37 62
The Unsorted portion of the array is 1 to
11
Maximum value in the unsorted portion is
95
at
8
Create New Random Array
Find the maximum value in the unsorted part
Do the swap


Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
87 30 34 62 39 49 30 70 62 52 37 95
The Unsorted portion of the array is 1 to
10
Maximum value in the unsorted portion is
at
Create New Random Array
Find the maximum value in the unsorted part
Do the swap


Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
87 30 34 62 39 49 30 70 62 52 37 95
The Unsorted portion of the array is 1 to
10
Maximum value in the unsorted portion is
87
at
0
Create New Random Array
Find the maximum value in the unsorted part
Do the swap


Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
37 30 34 62 39 49 30 70 62 52 87 95
The Unsorted portion of the array is 1 to
9
Maximum value in the unsorted portion is
at
Create New Random Array
Find the maximum value in the unsorted part
Do the swap


Sorting Program
0 1 2 3 4 5 6 7 8 9 10 11
37 30 34 62 39 49 30 70 62 52 87 95
The Unsorted portion of the array is 1 to
9
Maximum value in the unsorted portion is
70
at
7
Create New Random Array
Find the maximum value in the unsorted part
Do the swap



9. Due Date

Due date is Wednesday of Week 6.