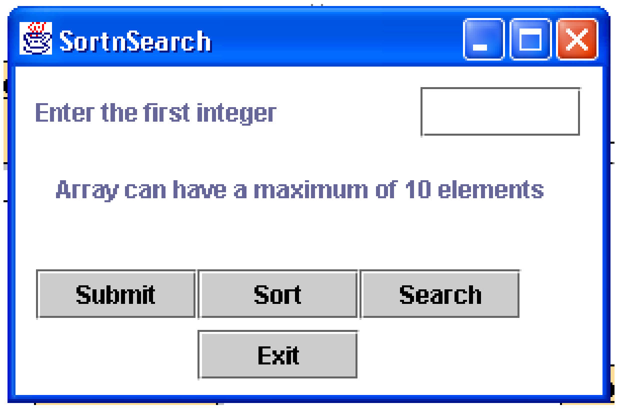
Lab Assignment 11 (week 11)

In this lab, you are going to learn to create a GUI project which involves

1. usage of Arrays (Array of integers is used).
2. Sorting an array and searching for an element.
3. usage of multiple windows in the GUI.

Specification:

Your GUI should look something like this.



Enter the first integer

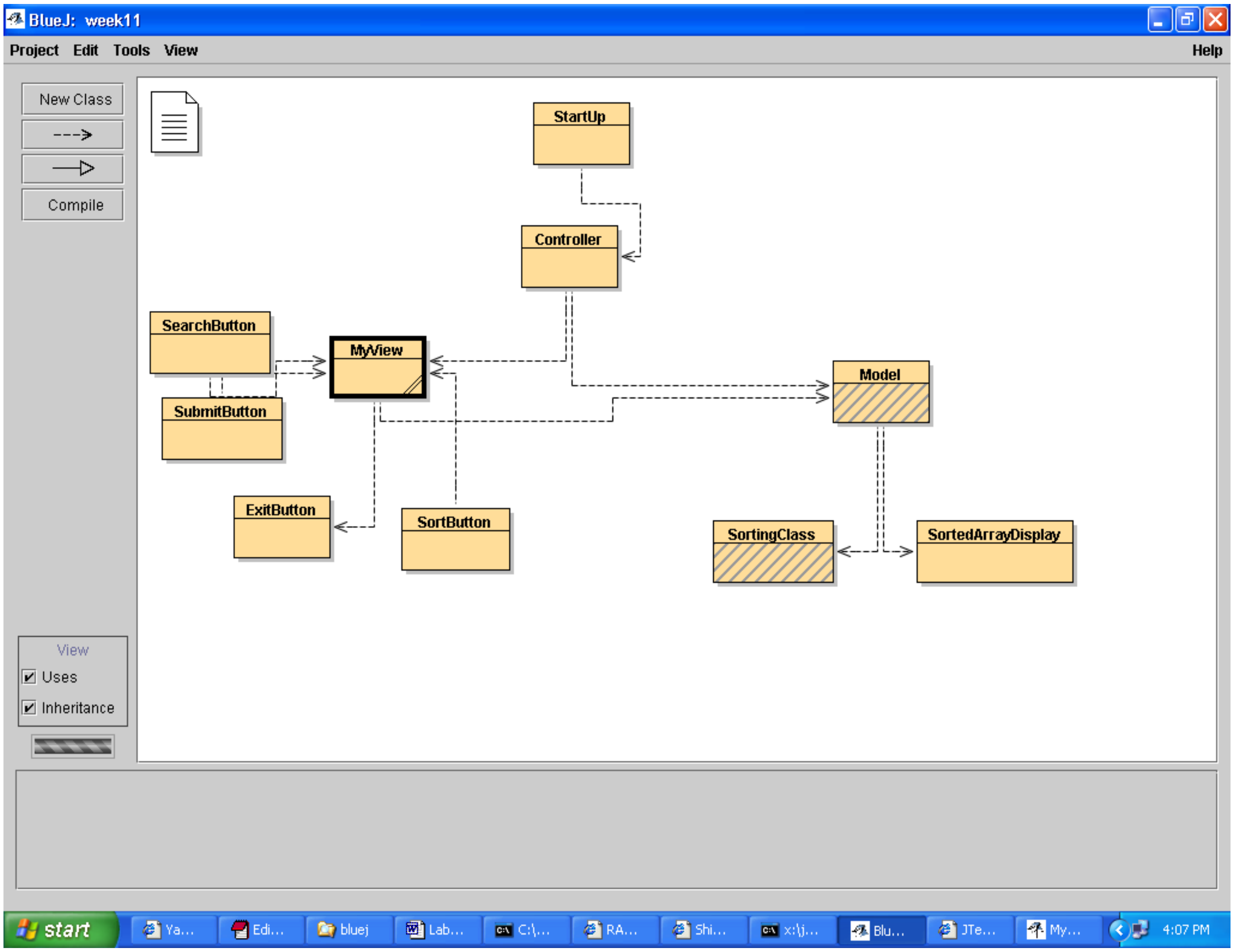
Array can have a maximum of 10 elements

If you look at the above figure, you see the following components in the JFrame.

1. Two Labels.
2. A TextField.
3. Four Buttons.

You are coding a project in which the user can fill an integer array of size 10 by entering an integer into the JTextField and hitting Submit JButton. If the user wants to fill the aray completely, he needs to do this action 10 times. The user can then use the Sort JButton to sort the array in ascending order. The sorted array will be displayed in a separate JFrame. To search for a particular element, the user should enter a search key (an integer) in the JTextField and hit Search. The result is shown as the JLabel gets updated.

At the end of the assignment, your Bluej window will look something like this.



Step1:

Open Bluej and create a new Project named Lab11.

Step2:

Create a new main class named *StartUp*.
Paste the following code.

```
class StartUp
{
        public static void main(String[] args)
        {
                new Controller();
        }
}
```

Step 3:

Create a class called *Controller* and put the following code in it.

```
class Controller
  {
        public Controller()
        {
                Model m = new Model();
                MyView view = new MyView(m);
        }
  }
```

Step 4:

The main component in this project is the GUI. Look at the above figure and think what components are needed and how they are arranged in the JFrame. The following is the code for the *MyView* class that you need to create.

```
import java.awt.*;
import javax.swing.*;
import java.awt.event.*;

class MyView extends JFrame
{
        private Container c;
        private JLabel lblPrompt = new JLabel("Enter the first integer ");
        private JLabel lblSearchResult = new JLabel("Array can have a maximum of 10
elements");
        private JTextField txtField = new JTextField();
        private SubmitButton submitButton = new SubmitButton("Submit", this);
        private SortButton sortButton = new SortButton("Sort", this);
        private SearchButton searchButton = new SearchButton("Search", this);
        private ExitButton exitButton = new ExitButton("Exit");
private Model model;
private int i;

public MyView(Model m)
{
            super("SortnSearch");
            model = m;

            c = getContentPane();
            setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            c.setBackground(Color.white);
            c.setLayout(null);
            setSize(320,200);

            lblPrompt.setBounds(10, 10, 220, 25);
            lblPrompt.setBackground(Color.white);

            txtField.setBounds(200,10,80,25);
            txtField.setBackground(Color.white);

            lblSearchResult.setBounds(20, 50,300,20);

            submitButton.setBounds(30, 100,80,25);
            sortButton.setBounds(110,100,80            ,25);
            searchButton.setBounds(190,100,80,25);
            exitButton.setBounds(110,130,80,25);

            c.add(lblPrompt);
            c.add(txtField);
            c.add(submitButton);
            c.add(sortButton);
            c.add(searchButton);
            c.add(exitButton);
            c.add(lblSearchResult);
            setVisible(true);
}

public void fillArray()  // This gets called when Submit button is clicked.
{
            i = Integer.parseInt(txtField.getText());
            lblPrompt.setText(model.fillArray(i));
            txtField.setText("");
}

public void sortArray()
{
            model.sortArray();
}

public void searchElement()  //This gets called when Search button is clicked
{
```

```
                int key = Integer.parseInt(txtField.getText());
                if(model.searchElement(key))
                        lblSearchResult.setText("The Element " + key +
"  is found");
                else
                        lblSearchResult.setText("The Element " + key +
" is NOT found");
        }//end of searchelement()
    }//end of class
```

Step 5:

Note that there is ActionListener for each of the four JButtons. Instead of putting the action listeners for each button in the *MyView,* consider this new way in which there is a separate class for each of the four JButtons which has the ActionListener for that JButton.Create four classes :*SubmitButton, SearchButton, ExitButton and SortButton*. Here's how the classes look like

//This is *SubmitButton* class

```
import javax.swing.*;
import java.awt.event.*;
// When the Submit JButton is pressed, fillArray() method of the MyView will be called. (look at
the actionPerformed method.)
class SubmitButton extends JButton implements ActionListener
{
    private MyView theView;

    public SubmitButton(String label, MyView v)
    {
        super(label);
        theView = v;
        addActionListener(this);
    }

    public void actionPerformed(ActionEvent ae)
    {
        theView.fillArray();
    }
}
```

//This is *SearchButton* class

```
import javax.swing.*;
import java.awt.event.*;

class SearchButton extends JButton implements ActionListener
{
    private MyView theView;

    public SearchButton(String label, MyView v)
    {
        super(label);
        theView = v;
        addActionListener(this);
    }

    public void actionPerformed(ActionEvent ae)
    {
        theView.searchElement();
    }
}
```

Similarly, write the code for the classes *ExitButton* and *SortButton*.

Step 6: Create a class called *Model* . Paste the following code into it.

```
class Model
{
    private final int MAX = 10; //used to specify the size of the array.
    private int array[] = new int[MAX];  //Look how an array of integers is created
    private int index = 0;
    private SortingClass SortingObject  = new SortingClass(); //This class has a method for
sorting.(NOT provided)
    SortedArrayDisplay sortedarraydisplay;

    public String fillArray(int element) //Called by MyView when Submit button is clicked
    {
        array[index] = element;
        if (index < (MAX - 1))
        {
                index++;
                return "Enter the next Element";
        }
        else
            return "Array is full, use Search or Sort";
    }
    public void sortArray()    // Called by MyView when Sort button is clicked
    {
        array = SortingObject.sort(array);
        sortedarraydisplay = new SortedArrayDisplay(array);
    }

    public boolean searchElement(int key) //Called by MyView when Submit button is
clicked
    {
        // This method searches the array for a given key and returns true or false
        depending on whether the element was found or not respectively.
        //WRITE THE REQUIRD CODE YOURSELF
    }//end of method.
}//end of class
```

Step 7:

When the Sort Button is clicked, the sortArray() of Model is called, which creates a new JFrame containing a JTextArea and fills in the sorted array elements.
The code to create a new window(JFrame) is given below.

```
//This is SortedArrayDisplay class
import javax.swing.*;
import java.awt.*;

class SortedArrayDisplay extends JFrame
{
                public SortedArrayDisplay(int array[]) {
                        super("Sorted Array Display");
                        setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
                        setSize(200, 300);
                        Container cp = getContentPane();
                        JTextArea text = new JTextArea();
                        cp.add(new JScrollPane(text));
                        for(int index = 0; index < array.length; index++)
                        {
                                text.append("Element " + index + " is " + array[index]);
                                text.append("\n");
                        }
                        setVisible(true);
        }
}
```

The code for the *SortingClass* is not provided. This has to be coded by you.
This class is used in the *Model* class. NOTE: Lines of code using *SortingClass* are colored dark red.
[HINT: The class *SortingClass* has got one method named sort which takes as input an integer array and returns an integer array.]

Show your lab instructor, the complete execution of the project including the functionality of Sorting and Searching. Try to implement a similar project in which you have an array of objects (instead of an integer array) and you need to sort the element **object.element**. Ask your instructor more about this.

Note: This assignment has been designed in lines with the HomeWork assignment due next week.