ICOM 4015-Advanced Programming
Spring 2014
Instructor: Dr. Amir H. Chinaei
TAs: Hector Franqui, Jose Garcia, and Antonio Tapia

Reference: **Big Java**
By Hortsmann, Ed 4

Lab 4
GUIs, Panels, as well as Mouse and Key Listeners

Department
of
Electrical and Computer Engineering
University of Puerto Rico at Mayagüez

Before laboratory:

1. Review JPanels
2. Review MouseListeners
3. Review KeyListeners
4. Print (at least) the evaluation sheet in the last page.

0 - Login to computer (1 minute)

1 - PART A: Build a GUI (50 minutes)

- So far you have created simple GUIs. During this lab, you will be working on a more complete GUI and make it fully responsive by using MouseListeners and KeyListeners.

- Let's focus on the GUI in this part. In next part, we focus on listeners.

- First, create a new project in Eclipse and name it NumericalKeyPadGUI

- Add two new classes.
  - One should be WindowViewer, which will be your main class, make sure you add the main stub to it.

  - The other one will be WindowMainPanel. This class will describe a JPanel. Your instructor will explain the usage of panels and the existing types.

- Add the following code to the WindowMainPanel class:

```
public class WindowMainPanel extends JPanel {

        public WindowMainPanel() {

                display =new JTextField("");

                …
        }

        private void drawGUI() {
                // WindowMainPanel is a panel as it extends JPanel
                // set its layout here.
                …

                // As Figure 2 shows, the second layer of the panels
                // has a CentralPanel
                // Create central panel here and set its layout.
                …

                // As Figure 2 shows, the third layer of the panels include a
                // diplay panel and a grid panel
                // define the display panel here, set its layout,
                // and add component display (that is a textbox) to it.
                …

                // define the keypad panel here, set its layout; note that
                // the GridLayout must have 4 rows and 4 columns
                // and add component buttons (which is an array of buttons) to it
                …
```

```
                // so far, you have 4 panels. BUT, the real hierarchy between
                // them has not defined yet.
                // to define the hierarchy according to Figure 2, add the central
                // panel to your main window panel
                // and add the display panel as well as the keypad panel to
                // your central panel
                …

                // add the component status bar to the south of your main
                // window panel
                this.add(statusBar, BorderLayout.SOUTH);

                //polishing the GUI a little bit
                …

        }

        …

        // define instance and static variables here
        private static JTextField display;
        …


        // When you are in Part B of the Lab, add two listener variables here:
        // one from PanelMouseListener type
        // the other from  PanelKeyListener type
        PanelMouseListener mListener = new PanelMouseListener();
        …

        // then go to WindowMainPanel constructor and add
        // these two listener variables to the buttons.
}
```

- Notice that drawGUI is just a private method that can be called only from class `WindowMainPanel`. The rational to use such a method is just to make our code more readable (by modularizing different tasks in different methods).

- Now, let's create the GUI. We are going to create a numerical keypad very similar to a calculator, but it will not have the complexity of a calculator. Figure 1 illustrates the GUI we intend to create:
-

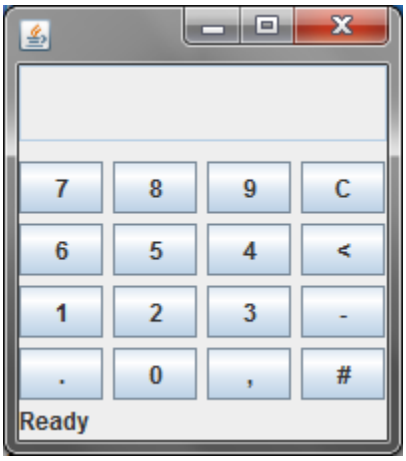

Figure 1

- There are three basic GUI components here: buttons, textbox, and a label. The textbox is being used as a display, and the label as the status bar in the lower part of the window.

- Let's create these components in our GUI (WindowMainPanel). Add declarations of variables for the display, buttons, and the label to the class. They'll be static instance variables. Well, following shows you the declaration of the display.

```
private static JTextField display;
...
...
```

  - Notice that the buttons are actually a grid. You are not going to declare each button in a separate statement. Instead, we will produce an array of buttons. As each button is better to have a label (caption), we are going to use the following variable for all captions of all the buttons:

```
public static final String[] KEYPAD_CAPTION = {"7", "8", "9", "C",
                                               "6", "5", "4", "<",
                                               "1", "2", "3", "-",
                                               ".", "0", ",", "#"};
```

  - What is remaining for you is to discover how the … above must be written. In other words, you still need to define the buttons and the label as static variables (similar to the line up there for the display).

  - Then, inside the constructor method, initialize all the variables you created. Notice that the captions of the buttons are provided in the array above. Think of a smart way of initializing your buttons. An individual initialization **will not** be accepted.

  - Your label should be initialized with the text "Ready".

  - **Show the code to your instructor (20 points)**

- Now, let's think about drawing the GUI.

- **For the next 20 minutes,** the lab instructor is going to explain two concepts:
  1) How JPanels are important in creating a GUI
  2) Different types of panel layouts, including `GridLayout, BoxLayout, BorderLayout`

- As you should have noticed by your lab instructor's explanations, there are more than one way to use panels and make the same GUI.

- Inside the drawGUI method, use JPanles and layouts complying the following guidelines: We expect to have the following panels, in the same hierarchical order shown in Figure 2.

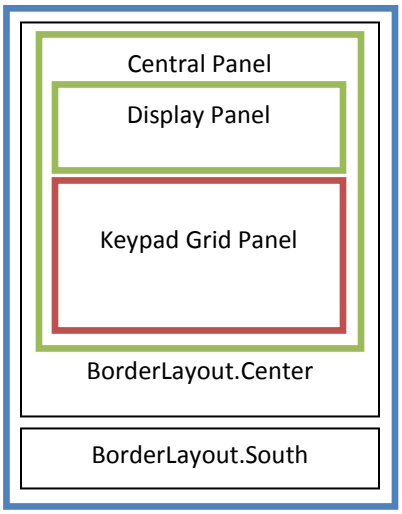

Figure 2

- Legend: Blue outline is for a BorderLayout, green outline is for a BoxLayout, and red outline is for a GridLayout. The black outlines only show the Center and the South of the Border Layout. In other words, they are not different panels.

- So, how many panels are in total?

- **Ntice: The class WindowMainPanel is a panel itself!!! This is the outmost panel in the figure above. That means, its layout is …?**

- For the rest of this step, complete the code of drawGUI method, by following the comments provided there and Figure 2 as your reference. In particular, declare as many panels as you need, and set their appropriate layouts. **Remember to set the layout of the WindowMainPanel class**. This class extends JPanel, so it's a panel itself!

- Just as a brief hint, we are showing you the lines of creating the keypadPanel as follows.

```
…
JPanel keypadPanel = new JPanel();
keypadPanel.setLayout(new GridLayout(...));
…
```

- To be able to run your code and see your progress, add the following statements to your *main* method.

```
public class WindowViewer {

        public static void main(String[] args) {

                JFrame frame = new JFrame();
                frame.setSize(200,225);
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
                frame.add(new WindowMainPanel());

                frame.setVisible(true);
            }
        }
```

- You should see a GUI like the one in Figure 3.

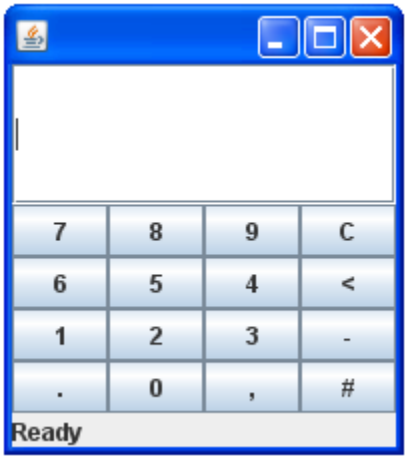

Figure 3

- **Show the result to your instructor (20 points)**

- As you see, you have created the GUI; But, Figure 3 is not as appealing as Figure 1. We need to polish our GUI now.

- So copy the following code to your drawGUI method in the area that is commented for polishing the code.

```
display.setFont(new Font(display.getFont().getFontName(), Font.PLAIN, 20));
display.setPreferredSize(new Dimension(180, 40));
display.setHorizontalAlignment(JTextField.RIGHT);
display.setEditable(false);
```

- Also, set the preferred size of the status bar label to a width of 180 and a height of 20.

- Moreover, as you see in Figure 3, there is no gap between the buttons that are in the keypad panel. To make it look better (like Figure 1), go to the definition of your keypad panel and set the layout as follows:

```
keypadPanel.setLayout(new GridLayout(4, 4, 5, 5));
```

- Finally, to add a nice space between your display and keypad panels, go to the lines that you added these two panels to your central panel and insert the following statement:

```
centralPanel.add(Box.createVerticalStrut(10)); // Adds a separator
```

- Now run your project again. You should exactly se Figure 1.

- **Show the result to the lab instructor (20 points)**

2 - PART B: Making a responsive GUI with listeners (40 minutes)

- Now, we want to make our GUI responsive. We not only want our GUI to respond to the mouse clicks also to the key pressed (typed). For this purpose, we are using Mouse and Key Listeners.

- From our GUI, the display and the status bar (the one implemented with a label) are two objects that would change. We want to only be able to edit these objects by using methods of the WindowMainPanel class that we define. This way, we are not allowing anyone from outside the class direct access to the instance variable.

    - Add the following code to the WindowMainPanel class:

```
public static String getDisplayText() {
	return display.getText();
}

public static void setDisplayText(String s) {
	display.setText(s);
}

public static String getStatusBarText() {
	return statusBar.getText();
}

public static void setStatusBarText(String s) {
	statusBar.setText(s);
}
```

    - Make sure to change the statusBar variable for the name of your corresponding variable.

- Now, add two new classes to your project:
    - One should be named PanelMouseListener, and the other PanelKeyListener. None of them needs the main stub.

- Let's begin with the MouseListener as you have already worked with something similar, but we will follow a different approach. In your PanelMouseListener class, make sure your class definition looks like this:

```
public class MousePanelListener implements MouseListener { ...
```

- Eclipse should immediately complain. Put your mouse over the name PanelMaouseListener, and choose the option "add unimplemented methods". Why?

- Now, five methods should have appeared. We are going to work with mouseClicked, mouseEntered, and mouseExited only.

    - Inside the mouseClicked method, write the following code:
      ```
      JButton temp = (JButton) arg0.getSource();
      ```

    - What is the use of this? Why do this? By doing this, we are copying into the temp variable the button that raised the event, in this case, the clicked button. This is to identify which button has been clicked. By inspecting the button's text, we can identify what action we have to perform.

    - Now, when we click any button that **is not** the "C" or "<" (which we use for backspace), we want to type into the display characters in the same fashion as a calculator.

    - When we click the "C" or "<", we want to delete all text in the display or remove the last typed character, respectively.

    - To do this, base your work in this code:
      ```
      if (temp.getText().equals("C")) {
            ... // Your code here
      }
      else if (temp.getText().equals("<")) {
            ... // Your code here
      }
      else {

      }
      ```

    - Think **very** carefully what should go inside the last else statement. While, in a naïve way of programming one may use individual buttons or hardcode text, we **DO NOT** accept it here as you are becoming professional developers now.

    - Add this listener to your buttons, after you have initialized them in the WindowMainPanel class. Run it and test it. Make sure it works before you proceed to the next part.

- Now, proceed to work with the mouseEntered method. We want to display in the status bar that the "C" button is for clearing the screen, and the "<" is for deleting the last input. Base your code similar to the one for the mouseClicked, but make sure you remove the other options that are not "C" and "<" because these are not needed.

- Now, we want back the text in the status bar of "Ready" when the mouse is not on buttons "C" or "<". Work this code on your mouseExited method. Base it in the code for the mouseEntered method.

- Run your program again.

  Show your work to the instructor (20 points).

- Once the MouseListener has been successfully implemented, it's time to work on the KeyListener.

  - Go to your PanelKeyListener class, and make it implement the KeyListener interface.

  - As before, you should add the unimplemented methods when Eclipse complains.

  - You should see three methods. We are only going to work with the keyTyped and keyPressed methods.

  - In the keyPressed method, we will only handle the "C" and "<" corresponding actions. In our keyboard, we can map these actions with "delete" and "backspace", correspondingly. Add the following code to your method and complete it:

```
if (arg0.getKeyCode() == KeyEvent.VK_DELETE) {
	... // Your code here
}
else if (...) {
	... // Your code here
}
```

  - Now, in the keyTyped method, handle the typing of the rest of the keys. They can be mapped directly to our keyboard. Base your code in the following way. But! Look at it clearly. It should look very similar to your mouseClicked method.

```
// Missing stuff here!
if (arg0.getKeyChar() == WindowMainPanel.KEYPAD_CAPTION[i].charAt(0)) {
	... // Your code here!
}
// Missing stuff here!
```

- Once your work is done, add the key listeners to all the buttons, the display and to the status bar.

- Run your program and **show your work to the instructor (20 points).**

After laboratory:

1. Investigate more types of layouts. You may find the following tutorial useful: http://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html#card

2. Use the KeyListener and Mouse Listener in a more fancy GUI or drawing. Try to implement (override) other methods of these classes too.

Evaluating Lab 4

Write Your Section# here: ……..

Please evaluate the quality of the lab and performance of the instructors by filling up the following table and give it to your lab representative. (Choose 5 as the highest and 1 as the lowest grade).

Items	5	4	3	2	1
The lab started **on time SHARP**.					
The instructor covered adequately the GUI material and answered the group's questions thoroughly.					
The instructor covered adequately the Listeners material and answered the group's questions thoroughly.					
The instructor explained the panels and different layouts clearly and adequately; furthermore, he managed his 20 minutes time properly as well.					
The instructor **overall** followed the specified **timeline** for each step					
You found the lab today **overall** Great (helpful, fruitful, interesting, etc.).					