**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**

**Introduction to GUIs and 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.
5. **Login to computer (1 minute)**
6. **Build a GUI (50 minutes)**
   1. So far you have created very simple GUI. During this lab, you will be working on a full GUI and make it fully responsive by using MouseListeners and KeyListeners.
   2. First, create a new project in Eclipse and name it NumericalKeyPadGUI
   3. 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.
   4. Add the following code to the WindowMainPanel class:

**public** **class** WindowMainPanel **extends** JPanel {

**public** WindowMainPanel() {

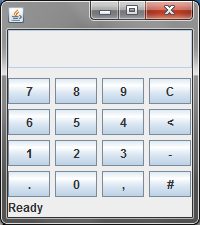
}

**private** **void** drawGUI() {

}

}

* 1. 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. Below is the GUI we intend to create:



* 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.
  2. Let’s create these components to our GUI. Add declarations of variables for the buttons, the display and the label. They’ll be static instance variables.

**private** **static** JTextField *display*;

**...**

**...**

* + - Notice that the buttons are actually a grid. You are not going to declare each button. We will produce an array of buttons for this. As for the those buttons’ labels, use this following instance variable:

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

* + - Initialize all the variables you just created for the components inside the constructor method. Notice that the labels for the buttons are provided in an array of String. 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)**
  1. Now, let’s draw the GUI. As your instructor must have explained, there are different types of JPanel’s. Using a combination of panels is the way to build a GUI. Do all this code inside the drawGUI method. For the moment being, we expect to have the following panels, in the same hierarchical order shown below in the figure.

Display Panel

Buttons Grid Panel

Central Panel

BorderLayout.Center

BorderLayout.South

* + - Notice the legend: Blue outline is for a BorderLayout, green outline is for a BoxLayout, red outline is for a GridLayout.
    - **Further notice**: Howe many panels are in total?
    - **Further notice: The class WindowMainPanel is a panel itself!!! This is the outmost panel in the figure above… so that means, its layout is…?**
    - 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!

JPanel keypadPanel = **new** JPanel();

keypadPanel.setLayout(**new** GridLayout(...));

...

...

* + - **Show the code to your instructor (20 points)**
    - Proceed to add features to each component. Write the following code inside your drawGUI method.

*display*.setFont(**new** Font(*display*.getFont().getFontName(), Font.*PLAIN*, 20));

*display*.setPreferredSize(**new** Dimension(180, 40));

*display*.setHorizontalAlignment(JTextField.*RIGHT*);

*display*.setEditable(**false**);

* + - Set the preferred size of the label to a width of 180 and a height of 20.
    - Add all components to their respective panels. The buttons should go in the Buttons Grid Panel. The display, in the Display Panel, and the label should go at the SOUTH location of the WindowMainPanel (the outmost panel).
    - Now, add panels to their own corresponding panels. Follow the hierarchy shown in the figure above.
  1. When you are done, go to the WindowViewer class and write the following code on it.

**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**);

}

}

* 1. Now run it and see what happens!
  2. **Show your work to your instructor (20 points)**

1. **Making a responsive GUI with listeners (50 minutes)**
   1. Now, we want to make our GUI responsive. We not only want our GUI to respond to the mouse clicks, but also that the users can type on their keyboards corresponding letters and they should work. For this purpose we are using Mouse and Key Listeners.
   2. From our GUI, the display and the status bar (the one implemented with a label) are two things that can 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 (**notice that**, they still are static, but not public anymore).
      * 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.**
  1. Add two new classes to your project:
     + One should be named PanelMouseListener, and the other PanelKeyListener. None of them should have the main stub.
  2. 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 name’s look s like this:

**public** **class** MousePanelListener **implements** MouseListener { ...

* 1. Eclipse should immediately complain. Put your mouse over the name MousePanelListener, and choose the option “add unimplemented methods”. Why is this?
  2. 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 is actually 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, correspondingly.
    - 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 else statement. Individual buttons **will not** be accepted.
    - 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.
  1. 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 in the past code for the mouseClicked, but make sure you remove the other options that are not “C” and “<” because these are not needed.
  2. Now, we want back the text in the status bar of “Ready” when the mouse is not in these buttons. Work this code on your mouseExited method. Base it in the code for the mouseEntered method.
  3. Run your program again. **Show your work to the instructor (20 points).**
  4. Once the MouseListener has been successfully implemented, it’s time to work in 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 this actions with “delete” and “backspace”, correspondingly. Add the following code to your method:

**if** (e.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... but! Look at it clearly. It should look very similar to your mouseClicked method.

// Missing stuff here!

**if** (e.getKeyChar() == WindowMainPanel.*KEYPAD\_GRID*[i].charAt(0)) {

... // Your code here!

}

// Missing stuff here!

* 1. Once your work is done, add the key listeners to all the buttons, the display and to the status bar.
  2. Run your program and **show your work to the instructor (20 points).**

### 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 overall followed the specified timeline for each step |  |  |  |  |  |
| You found the lab today overall Great (helpful, fruitful, interesting, etc.). |  |  |  |  |  |

**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 overall followed the specified timeline for each step |  |  |  |  |  |
| You found the lab today overall Great (helpful, fruitful, interesting, etc.). |  |  |  |  |  |