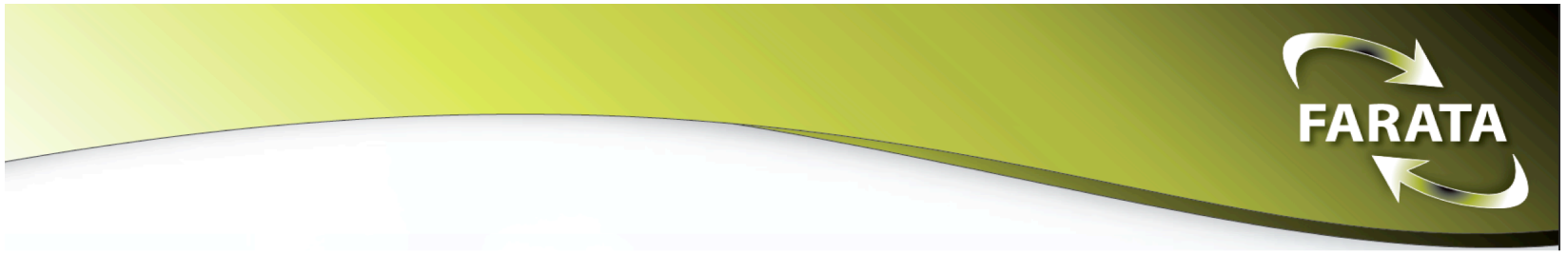


Java Programming
Unit 5

Intro to GUI with Swing. Listeners

AWT => Swing => JavaFX

- First, there was Abstract Windowing Toolkit (AWT)
- Swing library of GUI components replaced AWT.
- JavaFX 8 framework may replace Swing.

HelloWorld with Swing

```
import javax.swing.JFrame;

public class HelloWorld extends JFrame {

    public HelloWorld(){
    setSize(200,300);
    setTitle("Hello World");

    setVisible(true);
    }
    public static void main(String[] args) {
        HelloWorld myHello = new HelloWorld();
    }
}
```

After creating JFrame (one of the containers) add UI controls to it, for example:

```
JButton myButton = new JButton ("Click me");
add(myButton);
```

Layout Managers: Arranging UI Components Inside a Container

Use case: add controls to a JPanel and then a panel to a JFrame.

1. Create an instance of JPanel

2. Assign a layout manager to it

3. Instantiate some Swing controls and add them to the panel.

4. Add the panel to the top-level container - JFrame - by calling setContentPane() method.

5. Set the frame's size and make it visible.

Three Main Tasks of GUI Programming

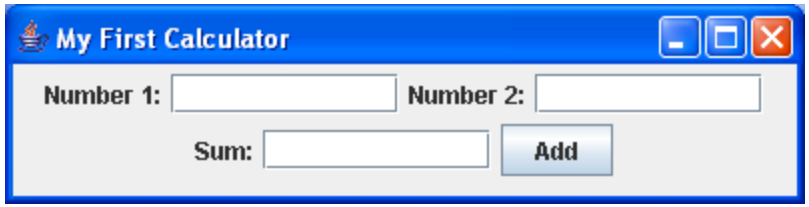
1. Create a nice looking layout of your GUI components.

2. Write the code to react on user-generated and system events.

3. Populate GUI components with the data.

Calculator With `FlowLayout`

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

public class SimpleCalculator {
 public static void main(String[] args) {
  // 1. Create a panel
     JPanel windowContent= new JPanel();

  // 2. Set a layout manager for this panel
     FlowLayout fl = new FlowLayout();
     windowContent.setLayout(fl);

  // 3. Create controls in memory
     JLabel label1 = new JLabel("Number 1:");
     JTextField field1 = new JTextField(10);
     JLabel label2 = new JLabel("Number 2:");
     JTextField field2 = new JTextField(10);
     JLabel label3 = new JLabel("Sum:");
     JTextField result = new JTextField(10);
     JButton go = new JButton("Add");

  // 4. Add controls to the panel
     windowContent.add(label1);
     windowContent.add(field1);
     windowContent.add(label2);
     windowContent.add(field2);
     windowContent.add(label3);
     windowContent.add(result);
     windowContent.add(go);
```




```
//5. Create the frame and add the panel to it
  JFrame frame = new JFrame(
                          "My First Calculator");

// 6. Add the panel to top-level container
   frame.setContentPane(windowContent);

// 7. set the size and make the window visible
   frame.setSize(400,100);
   frame.setVisible(true);
  }
 }
}
```

Swing Layout Managers

- `FlowLayout`
- `GridLayout`
- `BoxLayout`
- `BorderLayout`
- `CardLayout`
- `GridBagLayout`

First instantiate the layout manager, and then assign its instance to a container by calling `setLayout()`.

GridLayout

Say, your container needs to allocate 8 elements of the same size. You may do it in 4 columns and 2 rows: 4x2=8 cells.

```
JPanel windowContent= new JPanel();

// Set the layout manager for the panel
GridLayout gl = new GridLayout(4,2);
windowContent.setLayout(gl);

// Code to add components to the panel goes here
```


```
// To disable window resizing

frame.setResizable(false);
```

Walkthrough 1

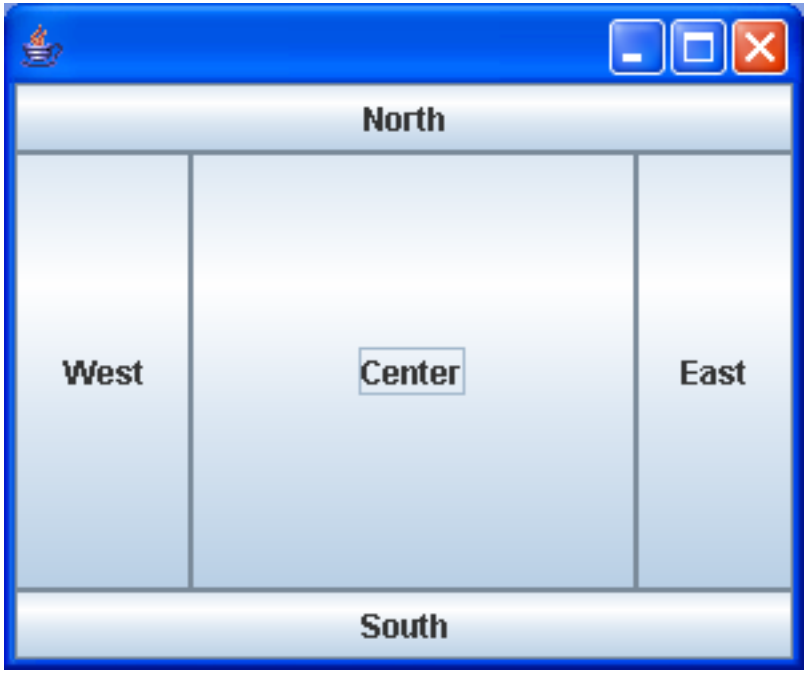
1. Download and import into Eclipse the source code of the Lesson8

2. Run SimpleCalculator. Stretch the window and observe the changes in the window layout.

3. Run SimpleCalculatorGrid. Stretch the window and observe the changes in the window layout.

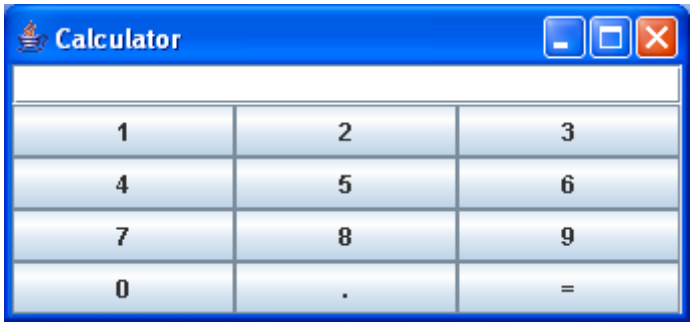
BorderLayout

BorderLayout divides a UI container into 5 imaginary areas: South, West, North, East, and Center. Add components to all or some of these areas in your container.



The calculator below uses only the North and Center.

The Center area uses GridLayout for allocating buttons.



`CardLayout`

- In a deck of cards only the top card is visible.
- Use `CardLayout` if you need to display several panels one at a time.
- See a `CardLayout` demo at http://bit.ly/NbmfRs

Absolute Layout

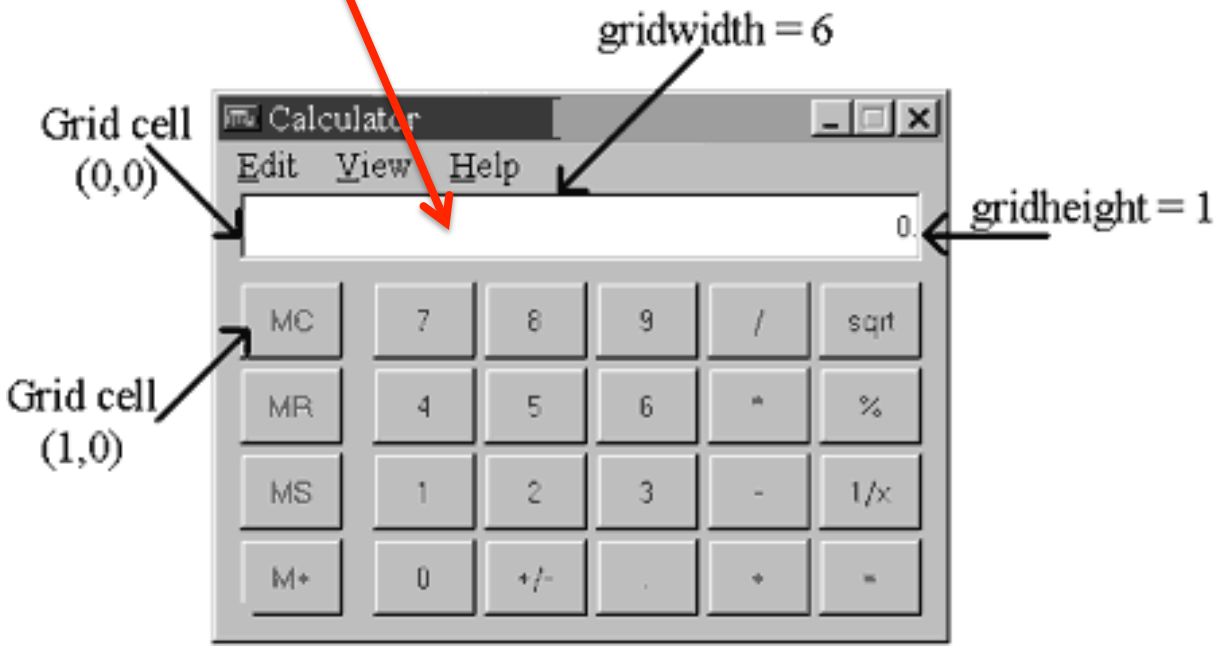
It's like not having any automatic layout.

```
windowContent.setLayout(null);

JButton myButton = new Button("New Game");

//Specify X and Y coordinates of each component
myButton.setBounds(100,200,40,20);
```

GridBagLayout

Allows laying out components of different sizes by assigning constraints to each grid element.

For example this cell will be 6 times wider than other cells in the grid:



Using GridBagConstraints

```
// Set the GridBagLayout for the window's content pane
 GridBagLayout gb = new GridBagLayout();
 this.setLayout(gb);

// Create an instance of the GridBagConstraints
// You'll have to repeat these lines for each component
// that you'd like to add to the grid cell
 GridBagConstraints constr = new GridBagConstraints();

//set constraints for the Calculator's displayField:

constr.gridx=0; // x coordinate of the cell
constr.gridy=0; // y coordinate of the cell

// this cell has the same height as others
constr.gridheight =1;

// this cell is as wide as 6 others
constr.gridwidth= 6;

// fill all space in the cell
constr.fill= constr.BOTH;
```

```
// proportion of horizontal space  taken by  this
// component
 constr.weightx = 1.0;

// proportion of  vertical space taken by  this
component
 constr.weighty = 1.0;

// position of the component within the cell
 constr.anchor=constr.CENTER;

 displayField = new JTextField();

// set constrains for this field
 gb.setConstraints(displayField,constr);

// add the text field to the window
 windowContent.add(displayField);
```

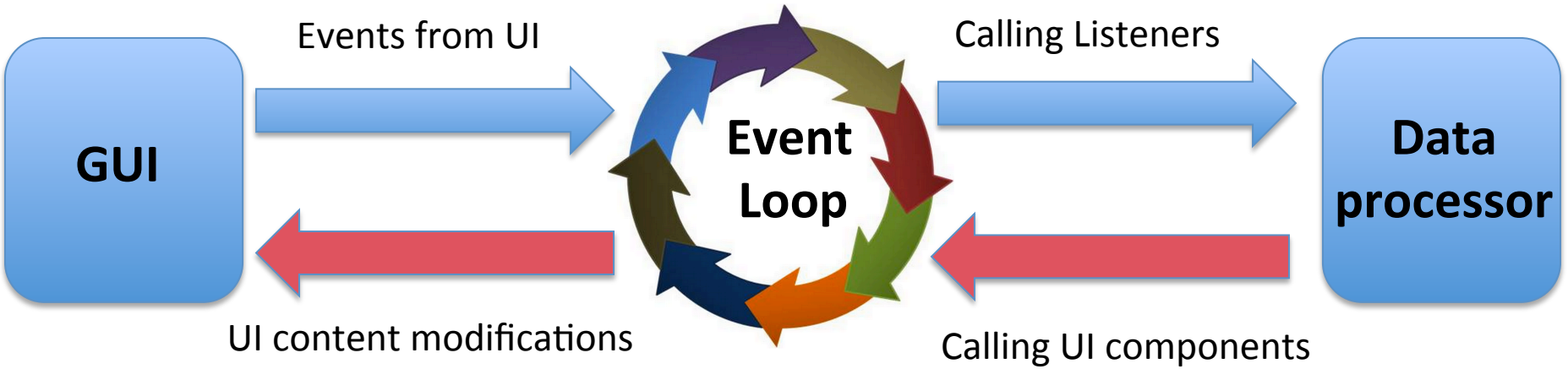
Events and Listeners

- There are two types of events: user-generated (clicks, mouse moves etc.) and system generated (paint, resize etc.).

- A click on the button fdispatches an event, *and if you want to process it*, create an `ActionListener` for this button.

- To process mouse moves, create one of the following: `MouseListener`, `MouseMotionListener`, or `MouseWheelListener`

The GUI Event Loop



Every request from/to UI is placed to an event queue.

When the event loop becomes available, it'll process the event.

Event handling code runs on the event dispatch thread.

Long running Processes and GUI updates

- Do not start long running processes on the event dispatch thread.

- If an application starts a separate thread and needs to update GUI, it should be done via event dispatching thread using `SwingUtilities.invokeLater()`.

- Swing includes the `SwingWorker` class that simplifies executing long-running processes in a separate thread. N

The `ActionListener` Interface

This interface declares just one **callback** method actionPerformed():

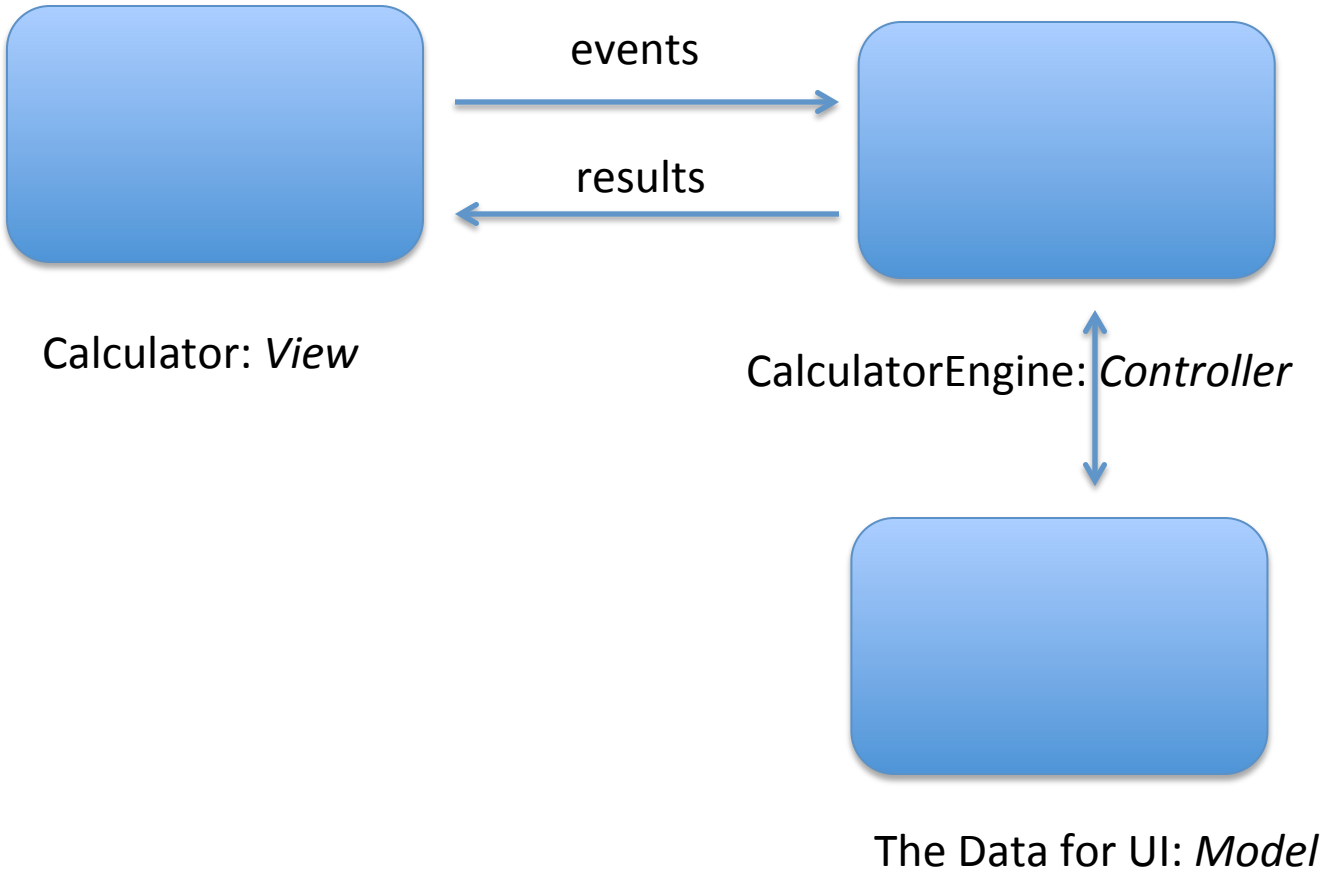
```
 public interface ActionListener extends EventListener

     void actionPerformed(ActionEvent e);
}
```

To process button events in your Calculator, there should be a class that implements the ActionListener. It can be the same class or another one, e.g. CalculatorEngine

```
public class CalculatorEngine implements ActionListener {

   public void actionPerformed(ActionEvent e){

       // Place the click-processing  code here
   }
}
```

In Unit 19 you'll see how to implement `ActionListener` using a lambda expression.

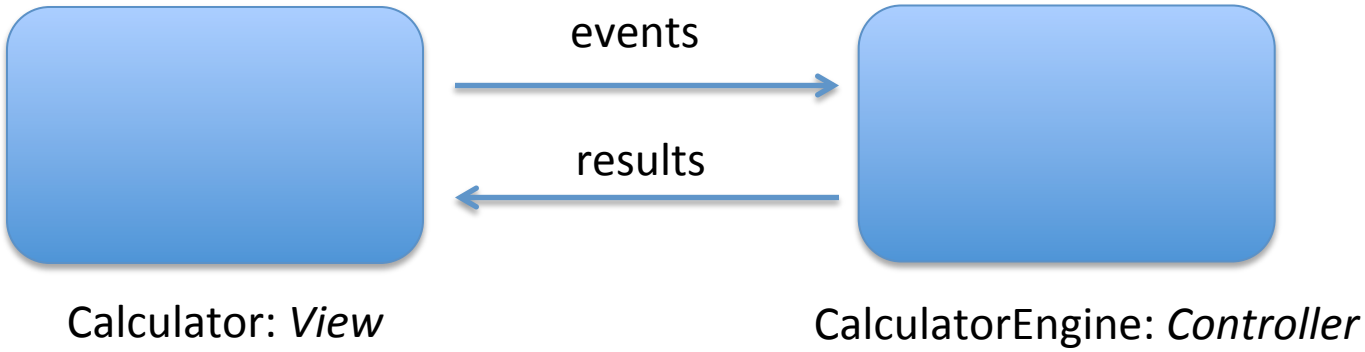
MVC: Model-View-Controller



Register components with listeners

```
CalculatorEngine calcEngine = new CalculatorEngine(this);

button0.addActionListener(calcEngine);
button1.addActionListener(calcEngine);
button2.addActionListener(calcEngine);
```

The code above can be located inside the class Calculator.
We pass the reference to the Calculator object using `this`.



You can register more than one listener with a component.

What triggered the event?

```
public class CalculatorEngine implements ActionListener {

   public void actionPerformed(ActionEvent e){

        // Get the source of this action
     JButton clickedButton=(JButton) e.getSource();

      // Get the button's label
       String clickedButtonLabel =  clickedButton.getText();

        // Concatenate the button's label
       // to the text of the message box
       JOptionPane.showConfirmDialog(null,
               "You pressed " + clickedButtonLabel, "Just a test",
               JOptionPane.PLAIN_MESSAGE);
   }
}
```

Passing Data Between Objects

Say, you need to reach a field in the `Calculator` from the `CalculatorEngine`.

The `Calculator` object passes the reference to itself to the `CalculatorEngine`:

```
CalculatorEngine calcEngine = new CalculatorEngine(this);
```

The engine's constructor stores reference to `Caclulator` in its own variable, say **parent**, and uses it in the method `actionPerformed()` to access the calculator's display field.

Bad practice: `parent.displayField.getText();`

Never try to access children of another object directly. Add to Calculator public getter and setter methods, for example:

```
getDisplayValue();
setDisplayValue(String value);
```

Adding Public API to Calculator

```
public class Calculator{
    private JTextField displayField;

    public void setDisplayValue(String val){
        displayField.setText(val);
    }

    public String getDisplayValue() {
        return displayText.getText();
    }

    // The rest of the code goes here
}
```

Do not allow direct access to UI components from other classes.

Walkthrough 2

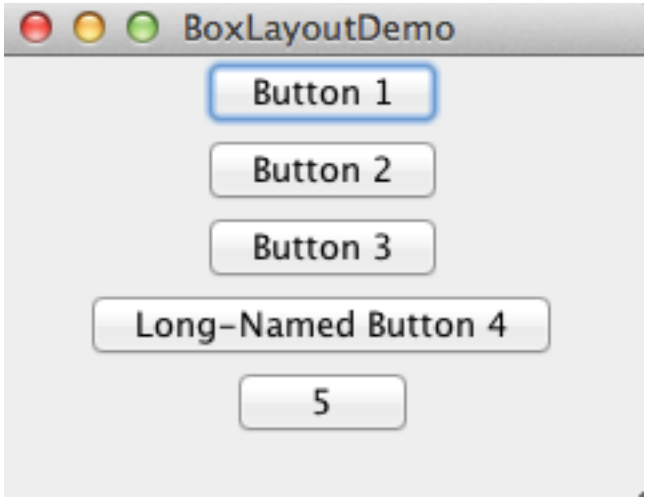
1. Download and import the code from Lesson 9 and review it with the instructor.

2. Run the Calculator program and see if the buttons react to clicks.

BoxLayout

Arrange GUI components either vertically or horizontally.

```
JFrame frame = new JFrame("BoxLayoutDemo");
       frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

 //Set up the content pane.
addComponentsToPane(frame.getContentPane());

public static void addComponentsToPane(Container pane) {
       pane.setLayout(new BoxLayout(pane, BoxLayout.Y_AXIS));

       addAButton("Button 1", pane);
       addAButton("Button 2", pane);
       addAButton("Button 3", pane);
       addAButton("Long-Named Button 4", pane);
       addAButton("5", pane);
}
```



Read about `BoxLayout` and test the above example at http://bit.ly/NbncJz

Additional Reading

Observer Design Pattern Tutorial: http://bit.ly/1crDR1I

Using `invokeLater()`: http://bit.ly/1GusA1C

Using `SwingWorker` class: http://bit.ly/1E90eG9

Homework

1. Get familiar with the layout manager `GridBagLayout`.

2. Do the assignment from the Try It section from Lesson 8 and 9 from the textbook.

3. Go over the Java Swing tutorial at http://bit.ly/1hHLUKZ .

4. Modify Calculator.java to use BoxLayout.