CMSC 341

Building Java GUIs

Why Java GUI Development?

- Course is about Data Structures, not GUIs.
- We are giving you the opportunity to do extra credit and have some fun on the project.
- GUIs are a good example of Object Oriented Programming.
- GUIs are another example of a container.

Java and GUIs

- There are two packages that generate GUI components in Java.
  - *java.awt*
  - *javax.swing*
- The AWT (Abstract Windows Toolkit)
  - Came first
  - No platform independence
- Swing
  - Part of Java Foundation Classes (released with Java 2)
  - Built on top of the AWT
  - Offers platform independence

Containers

- In Java, all GUI objects go into a Container.
- A top level container can stand alone in a web browser or in an operating system.
  - JFrame
  - JApplet
- Some containers may only be added to other containers.
  - JPanel

JFrame Methods

- ***add(Object)*** - adds objects to the frame.
- ***setVisible(boolean)*** - makes the frame visible
- ***setLocation(int x, int y)*** – aligns top left corner of frame with coordinates on screen
- ***setSize(int width, int height)*** – sets size of frame in pixels
- ***setDefaultCloseOperation(Windows.constants.EXIT_ON_CLOSE);***

JFrame Code

```
import javax.swing.*;
import java.awt.*;
public class UpperCaseConverter extends JFrame
{
    public UpperCaseConverter(String name){
            super(name);
            setLocation(300, 100);
            setSize (400,300);
            setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
    }

    public static void main(String args[]){
            UpperCaseConverter ucc =
                        new UpperCaseConverter("Convert to Upper Case");
            ucc.setVisible(true);
    }
}
```

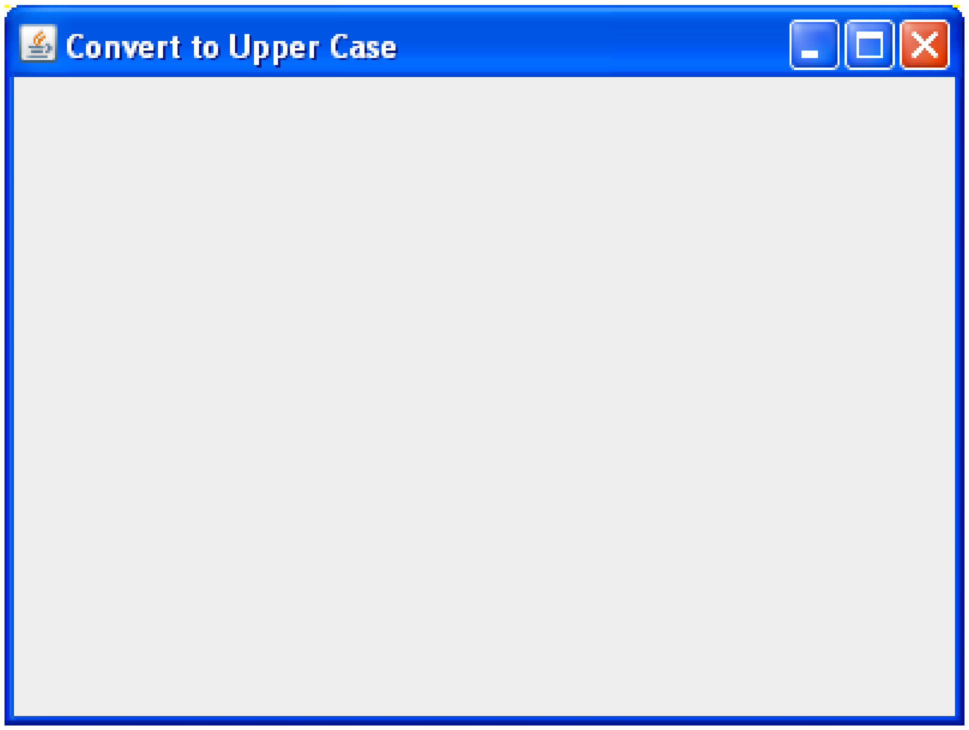
Constructor sets the title, size and location of the JFrame

Makes program end when window is closed

Instantiates JFrame and makes it visible

JFrame Example

- The code on the previous page renders the following:



LayoutManagers

- Every container has an underlying default LayoutManager.
- The LayoutManager determines
  - the size of the objects in the container and
  - how the objects will be laid out.
- The default LayoutManager for a JFrame is a BorderLayout.

BorderLayout

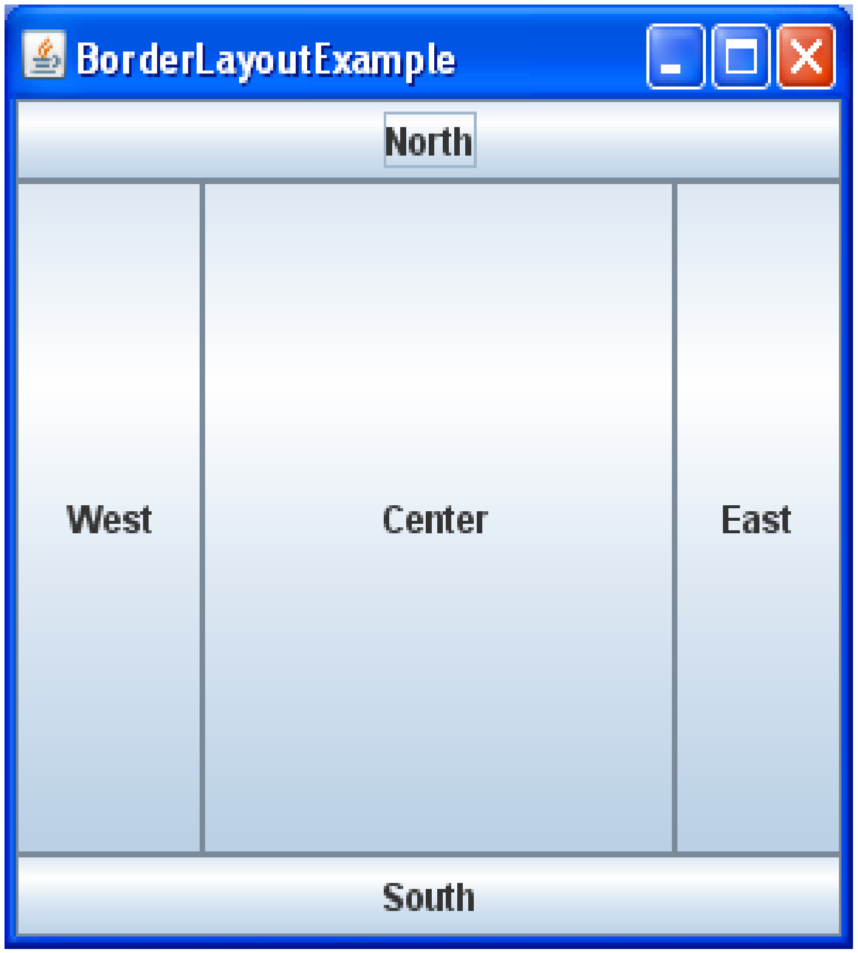
- Divides container into five regions
  - BorderLayout.NORTH
  - BorderLayout.SOUTH
  - BorderLayout.CENTER
  - BorderLayout.EAST
  - BorderLayout.WEST
- One component per region
- Component takes size of region
- Center region is greedy
- Components are added to center by default

BorderLayout Code

```
import java.awt.*;
import javax.swing.*;
public class BorderLayoutExample extends JFrame {
  public BorderLayoutExample(String name) {
      super(name);
      setSize(300,300);
      add(new JButton("North"), BorderLayout.NORTH);
      add(new JButton("South"), BorderLayout.SOUTH);
      add(new JButton("East"), BorderLayout.EAST);
      add(new JButton("West"), BorderLayout.WEST);
      add(new JButton("Center"), BorderLayout.CENTER);
  }
  public static void main(String args[]) {
      BorderLayoutExample b = new
          BorderLayoutExample("BorderLayoutExample");
      b.setVisible(true);
  }
}
```

Specialized add method for adding to regions

BorderLayoutExample



JPanel

- However, we want to put several buttons in the North region of the GUI, but BorderLayout only allows one component per region…
- Add a second level container like a JPanel.
- JPanels have a FlowLayout manager by default.

FlowLayout

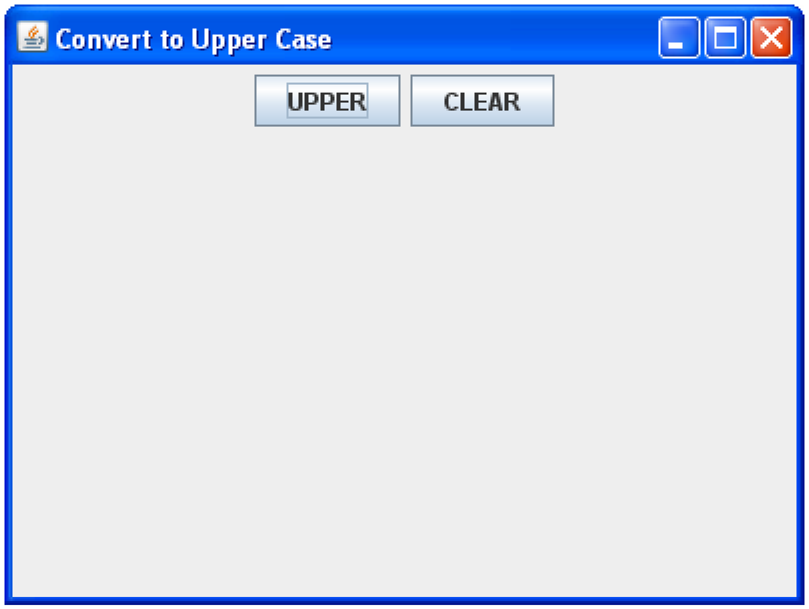
- Lays components in a fluid direction as determined by its orientation.
- By default, orientation is L -> R, T -> B.
- Possible to set the horizontal and vertical width between components.
- Components take preferred size.
  - For buttons, preferred size is the size of the text within them.

JPanel and FlowLayout Code

```
//omitting code here from previous example
public class UpperCaseConverter extends JFrame
{
      //Since we are expecting to make these components to
      //react to user interaction we make them object data
      JButton upper;
      JButton clear;
      public UpperCaseConverter(String name){
                //omitting code here from previous example
                JPanel top;
                top = new JPanel();
                upper = new JButton("UPPER");
                clear = new JButton("CLEAR");
                top.add(upper);
                top.add(clear);
                add(top, BorderLayout.NORTH);
      }
      //omitting code here from previous example
}
```

JPanel and FlowLayout Example

- Code on previous page renders as follows:



- But, we also need a text field to enter text.

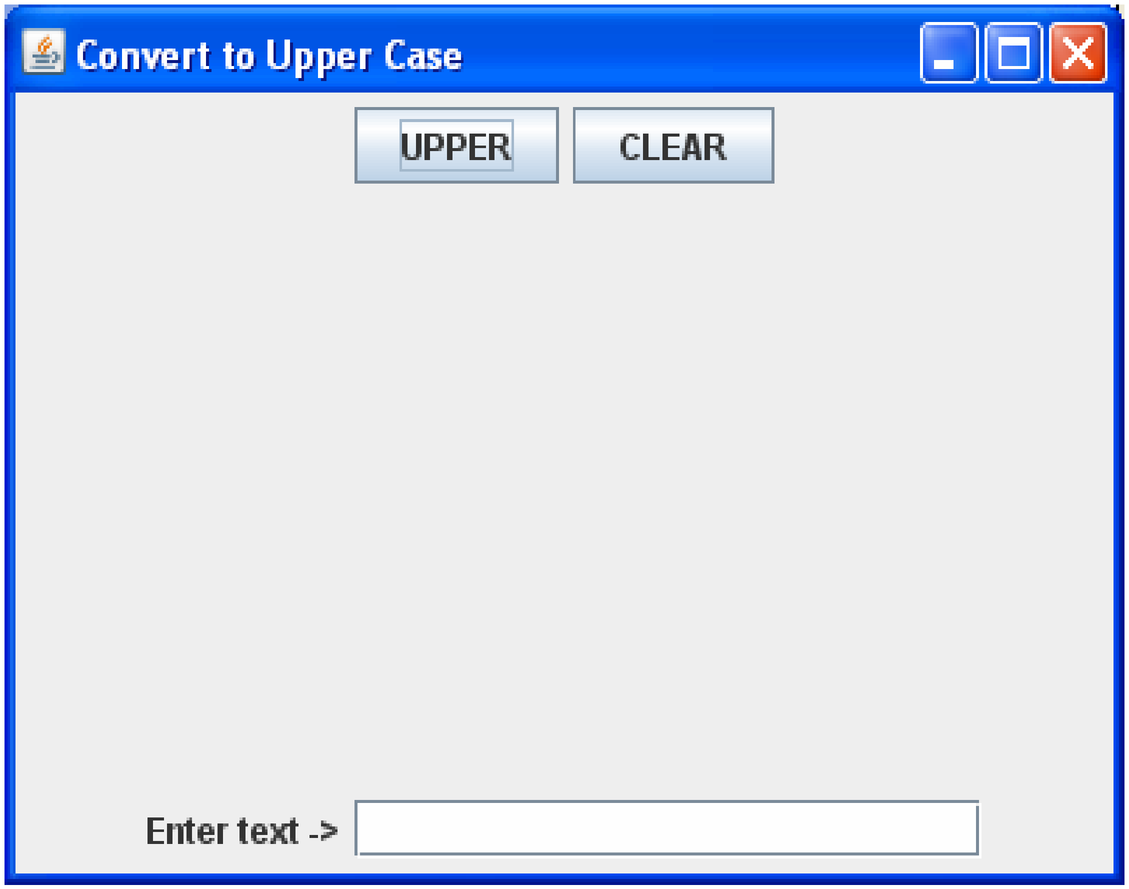
Second JPanel

```
public class UpperCaseConverter extends JFrame
{
    //code omitted from previous example
    JTextField input;

    public UpperCaseConverter(String name){
          //code omitted from previous example
          JPanel bottom = new JPanel();
          JLabel label = new JLabel("Enter text ->");
          input = new JTextField(20);
          bottom.add(label);
          bottom.add(input);
          add(bottom, BorderLayout.SOUTH);

    }
     //code omitted from previous example
}
```

JLabel **may also take an *Icon* or both a *String* and *Icon* in its constructor**

JTextField **takes an int which indicates the number of characters to be displayed**

Second JPanel Example



How would we add a *JTextArea* to the center of our frame?

JTextArea

- Add *JTextArea* reference to object data so that it can be referenced by all member methods.
- Instantiate *JTextArea* reference in constructor method and add reference to the center of the *JFrame*.

```
JTextArea output;
```

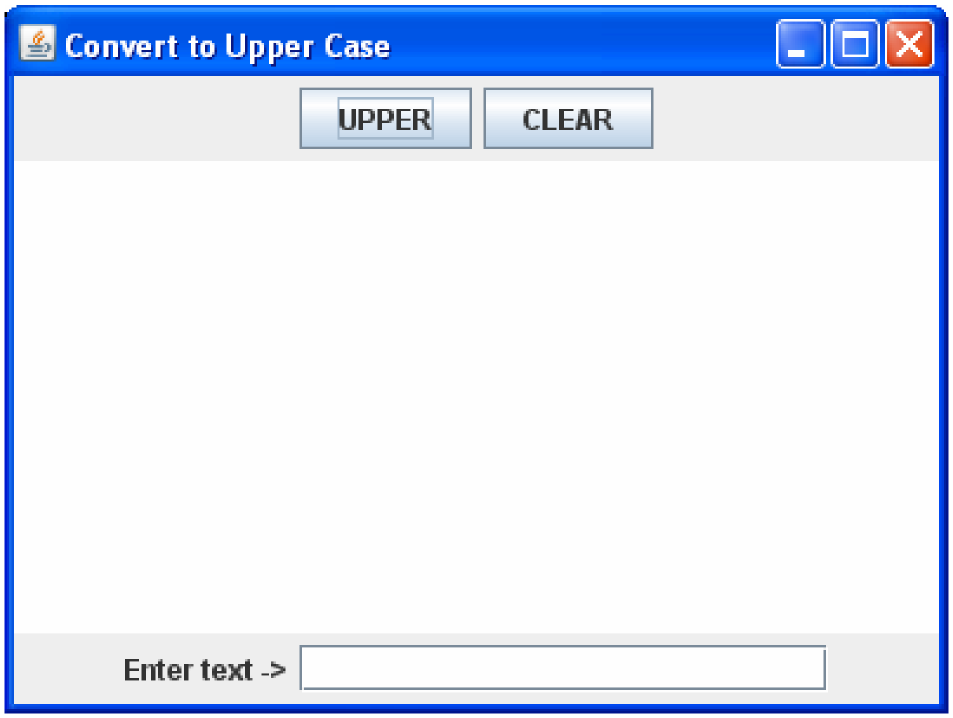
Declare outside of methods so object data

```
output = new JTextArea(10, 20);
add(output);
```

Constructor for JTextArea takes number of rows and columns

JTextArea Example

Next time, we will make this GUI functional.



JComponent Methods

- There exists several JComponent methods that allow you to change the look of a component
  - setBackground(Color)
  - setForeground(Color)
  - setFont(Font)
  - setPreferredSize(Dimension)
  - setAlignmentX(float)
  - setAlignmentY(float)

Values for all the arguments of these methods are already defined in Java.

More LayoutManagers

- Seven Basic Layout Managers in Java
  - BorderLayout
  - BoxLayout
  - CardLayout
  - FlowLayout
  - GridLayout
  - GridBagLayout
  - OverlayLayout
- We will only focus on two more of these.
  - GridLayout
  - BoxLayout

GridLayout

- Creates a grid with number of rows and columns given in the constructor
- One component per cell
- Cells of equal size
- Component take the size of the cell

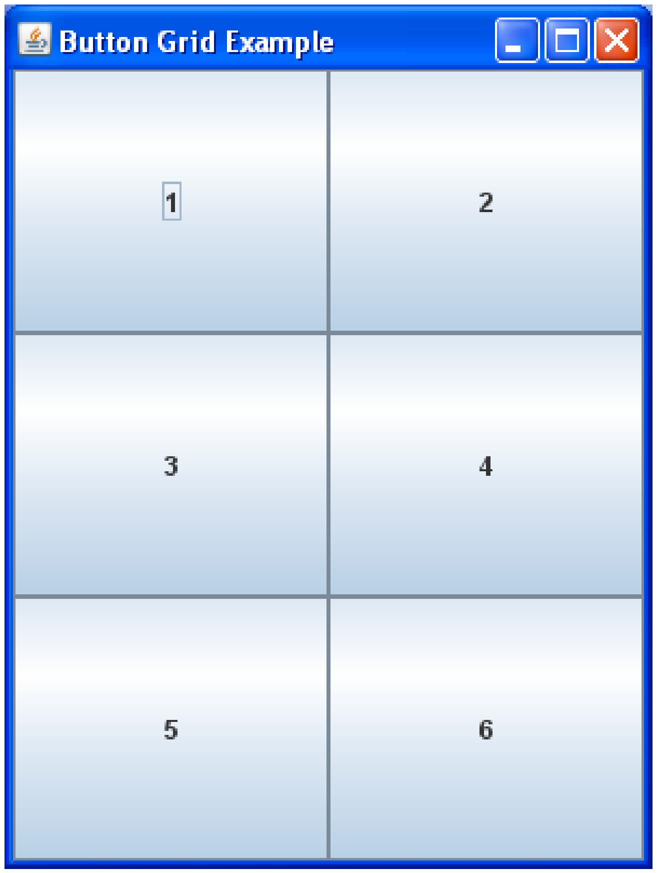
GridLayout Code

```
import java.awt.*;
import javax.swing.*;
public class ButtonGrid extends JFrame {
    public ButtonGrid() {
            super("Button Grid Example");
            setLayout(new GridLayout(3,2));
            setSize(300,400);
            add(new JButton("1"));
            add(new JButton("2"));
            add(new JButton("3"));
            add(new JButton("4"));
            add(new JButton("5"));
            add(new JButton("6"));
    }
    public static void main(String arg[]){
            ButtonGrid bg = new ButtonGrid();
            bg.setVisible(true);
    }
}
```

The *setLayout* method changes a container's LayoutManager.

Compare the order in which the buttons are added to the GUI on the next page.

GridLayout Example



BoyLayout

- Components are arranged either vertically or horizontally depending on parameter
  - BoxLayout.X_AXIS
  - BoxLayout.Y_AXIS
  - BoxLayout.LINE_AXIS
  - BoxLayout.PAGE_AXIS
- Components will not wrap even if container is resized
- Allows for filler (“glue”) between components to make them space evenly within container
- Part of javax.swing package

BoxLayout Code

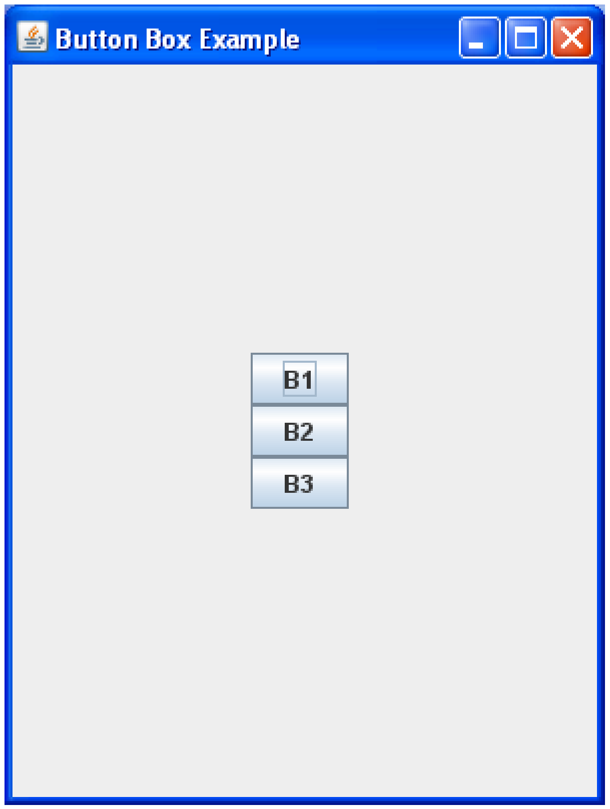
```
import java.awt.*;
import javax.swing.*;
public class ButtonBox extends JFrame {
    public ButtonBox() {
            super("Button Box Example");
            JPanel p = new JPanel();
            JButton b1 = new JButton("B1");
            JButton b2 = new JButton("B2");
            JButton b3 = new JButton("B3");
            b1.setAlignmentX(Component.CENTER_ALIGNMENT);
            b2.setAlignmentX(Component.CENTER_ALIGNMENT);
            b3.setAlignmentX(Component.CENTER_ALIGNMENT);
            p.setLayout(new BoxLayout(p,BoxLayout.Y_AXIS));
           setSize(300,400);
           p.add(Box.createGlue());
            p.add(b1);p.add(b2)p.add(b3);
           p.add(Box.createGlue());
           add(p);
    }
    //main goes here
}
```

Component method to align the button horizontally

Constructor for a *BoxLayout* takes an instance of its container

Here is the "glue"

BoxLayout Example



Calculator

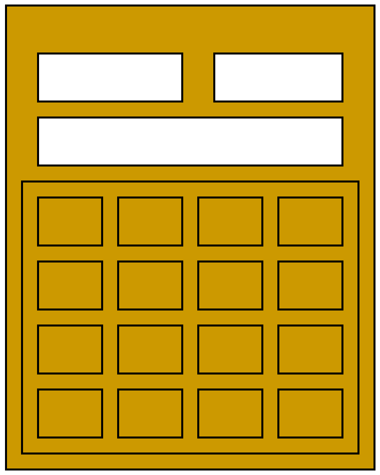
- What do we need for a Calculator GUI?
  - 16 JButtons
    - Numbers 0-9
    - Operators + - x / = .
  - 3 JTextFields
    - 2 operands
    - 1output
- Which need to respond to events?

Declare Object Data

```
import java.awt.*;
import javax.swing.*;
public class Calculator extends JFrame
{
      JButton [] numbers = new JButton[10];
      JButton plus;
      JButton minus;
      JButton mult;
      JButton div;
      JButton equals;
      JButton dot;
      JTextField output;
      JTextField operand1;
      JTextField operand2;
}
```

Constructor

- Constructor is where everything will be created.
- Before beginning decide
  - how to break up your frame into panels,
  - which LayoutManager goes where,
  - what components will go where.



Instantiate Object Data

```
public Calculator()
{
     super("My Calculator");
     numbers = new JButton[10];
     for(int i = 0; i < 10; i++)
             numbers[i] = new JButton("" + i);
     plus = new JButton("+");
     minus = new JButton("+");
     mult = new JButton("x");
     div = new JButton("/");
     equals = new JButton("=");
     dot = new JButton(".");
     operand1 = new JTextField(10);
     operand2 = new JTextField(10);
     output = new JTextField(21);

     setSize(300,400);
     setDefaultCloseOperation(WindowConstants.EXIT_ON_CLOSE);
}
```

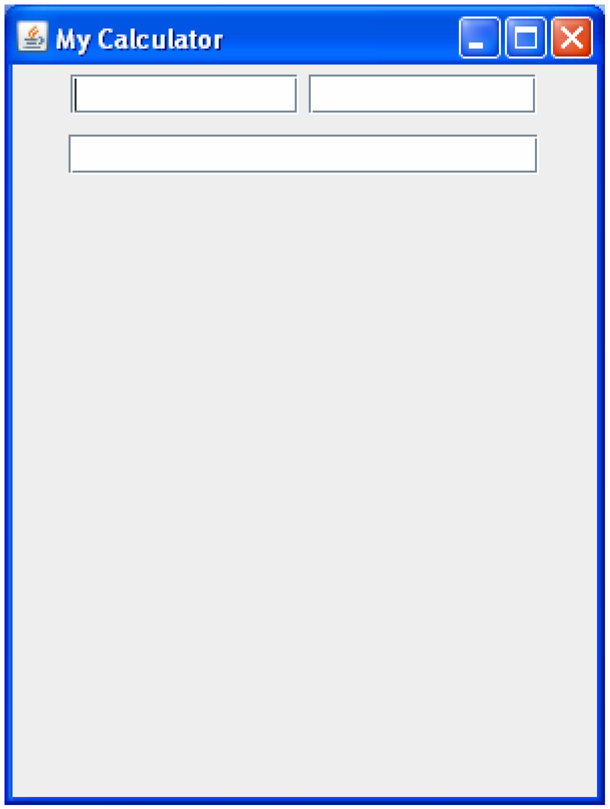
Setting properties for the frame, too

Top Panel

- Need to split the top panel into a grid with two panels. Why?

```
JPanel top = new JPanel();
top.setLayout(new GridLayout(2,1));
add(top, BorderLayout.NORTH);

JPanel input = new JPanel();
input.add(operand1);
input.add(operand2);
top.add(input);

JPanel results = new JPanel();
results.add(output);
top.add(results);
```

Rendering of Previous Code



The Center Panel

- The center will also consist of a grid with four rows and four columns.
- What happens if we add buttons directly to grid?
- What can we do to get our desired effect?
- What do we want the calculator to do when we resize?

Panels of Panels

- **Often GUI programmers create methods to create Panels.**

```
private JPanel
getRow(JButton b1, JButton b2, JButton b3, JButton b4)
{
   JPanel row = new JPanel();
   row.setLayout(new
       BoxLayout(row,BoxLayout.X_AXIS));
   row.add(b1);
   row.add(b2);
   row.add(b3);
   row.add(b4);
   return row;
}
```

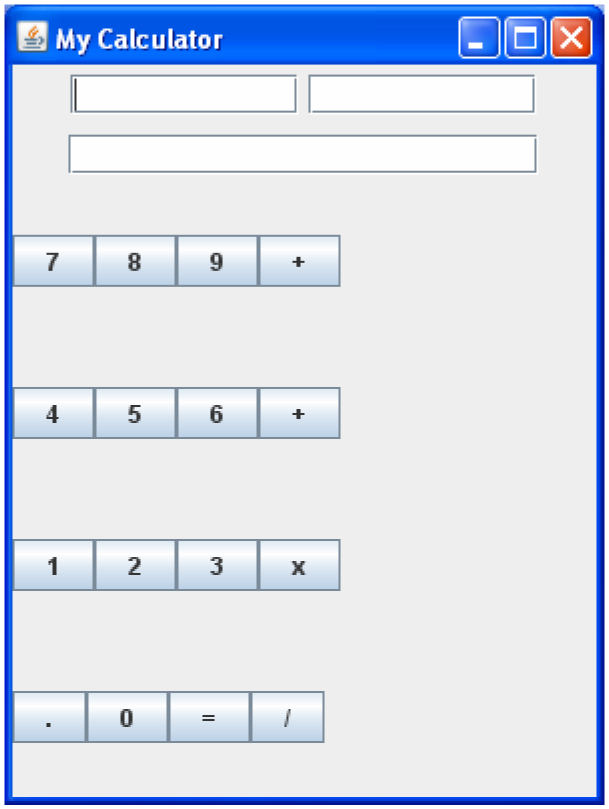
Panels of Panels (cont.)

- Several calls to the method are made from the constructor.

```
JPanel center = new JPanel();
center.setLayout(new GridLayout(4,1));
center.add(getRow(numbers[7], numbers[8], numbers[9], plus));
center.add(getRow(numbers[4], numbers[5], numbers[6], minus));
center.add(getRow(numbers[1], numbers[2], numbers[3], mult));
center.add(getRow(dot, numbers[0], equals, div));
add(center);
```

Calculator

- Adding the previous code, the calculator now renders like so.



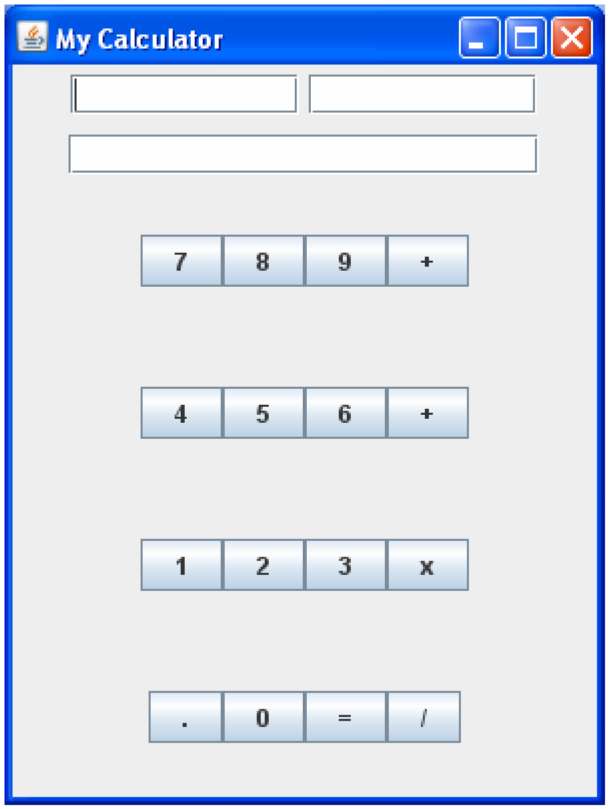
Adding Glue

- Adjust the Panel method to incorporate some glue.

```
private JPanel
getRow(JButton b1, JButton b2, JButton b3, JButton b4)
{
   JPanel row = new JPanel();
   row.setLayout(new
                BoxLayout(row,BoxLayout.X_AXIS));
   row.add(Box.createHorizontalGlue());
   row.add(b1);row.add(b2);row.add(b3);row.add(b4);
   row.add(Box.createHorizontalGlue());
   return row;
}
```

Almost there.

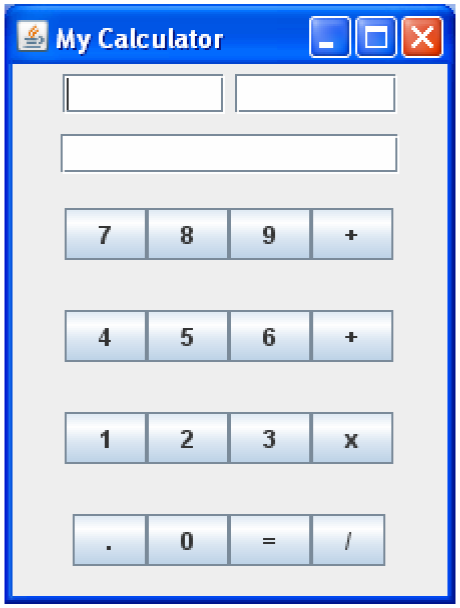
- Now it looks like so.



Small Changes and Viola

```
setSize(225,300)
operand1 = new JTextField(7);
operand2 = new JTextField(7);
output = new JTextField(15);
```

Next class we will make it functional using Java's Event Delegation Model.


My Calculator
7
8
9
+
4
5
6
+
1
2
3
x
.
0
=
/