Lecture 28 – Swing Layout Programming

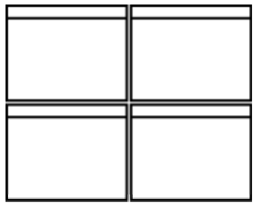
Lecturer: Gerald Weber

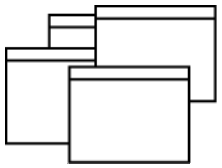
Overview

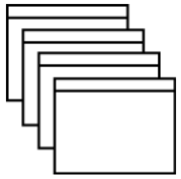
- With a toolkit like Swing, it’s more like you give the window manager ideas (or at best requirements) for your interface will look like, rather than specifying it exactly
  - Good in some ways: it can resize intelligently, and potentially the look-and-feel can evolve over time
  - But it is a paradigm that takes getting used to: working with layout managers
- Also, there’s the perennial problem of lack of “screen real estate”
  - How do we fit everything in?!

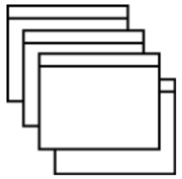
Windows

- Can managed real estate issues with window placements


Tiled windows


Overlapping windows


Cascading windows

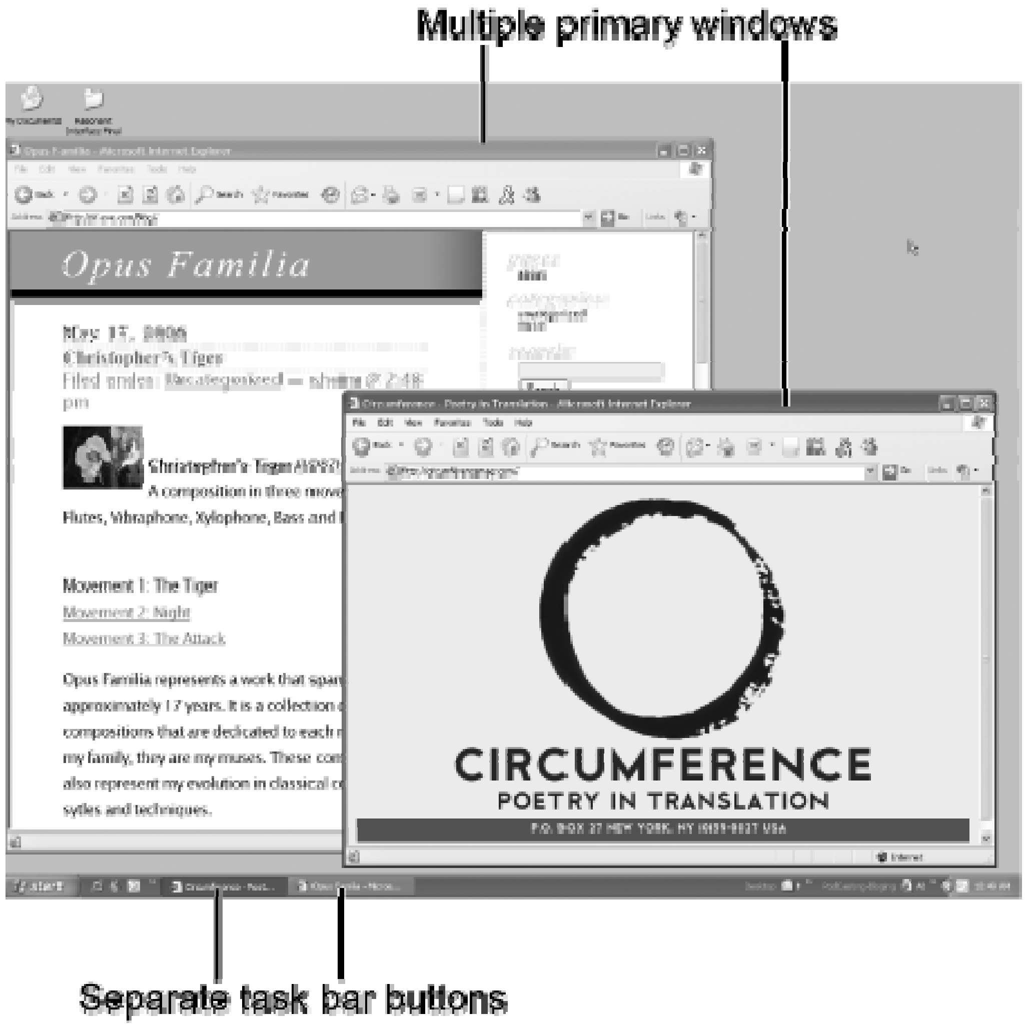

Interrupted cascade

Window Interfaces – *SDI*

Single

Document

Interface



Single document interface—Microsoft Internet Explorer®

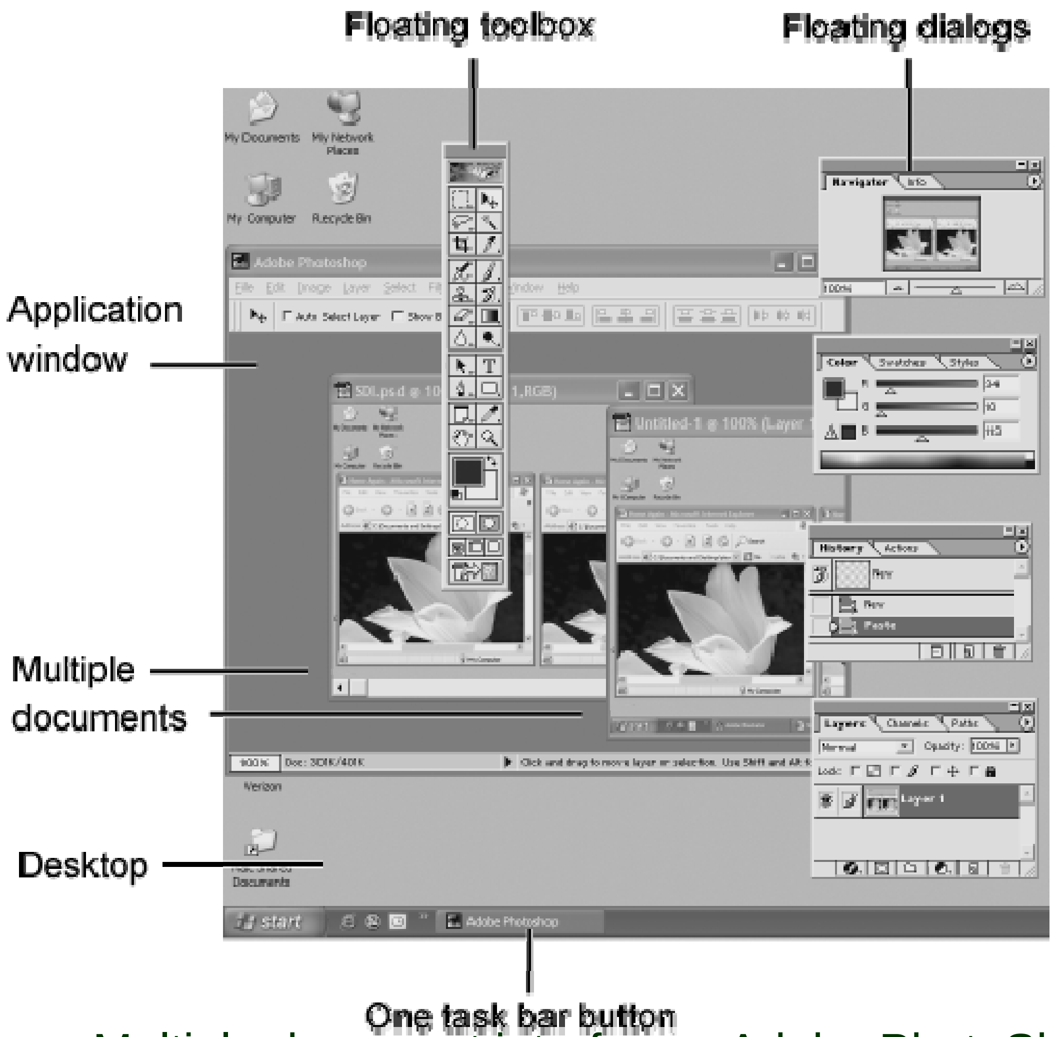
Window Interfaces – *MDI*

Multiple

Document

Interface

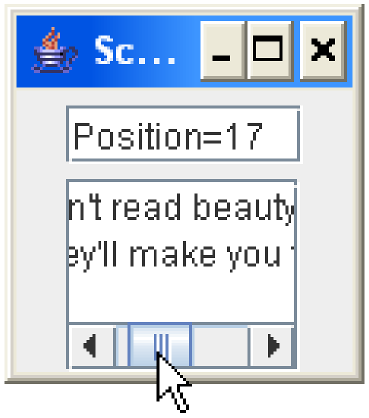
(more powerful,
    but also more
    complex for
    user)



Multiple document interface—Adobe PhotoShop® applicatio

Scrollbar Example Part 1

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

public class ScrollBarExample {
  JFrame frame;
  JPanel panel;
  JTextArea area;
  JTextField field;
  JScrollPane scrollpane;

  public static void main(String[] args) {
    ScrollBarExample v = new ScrollBarExample();
  }

  public ScrollBarExample() {
    // see next slide…
  }
}
```



Position=17

n't read beauty
ey'll make you

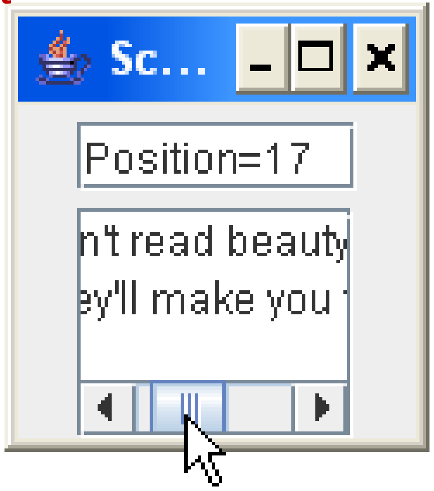
Simple technique to manage screen real estate

Scrollbar Example Part 2

```
public ScrollBarExample() {
    frame = new JFrame("ScrollBarExample");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(100, 130);

    field = new JTextField(7);
    area = new JTextArea("…", 3, 7);
    scrollpane = new JScrollPane(area);
    scrollpane.getHorizontalScrollBar()
      .addAdjustmentListener(new AdjustmentListener(){
          public void adjustmentValueChanged(
            AdjustmentEvent e){
            field.setText("Position=" + e.getValue());
          }
       });

    panel = new JPanel();
    panel.add(field);
    panel.add(scrollpane);
    frame.add(panel);
    frame.setVisible(true);
}
```

Yes, it's that easy to make something be in a scroll pane

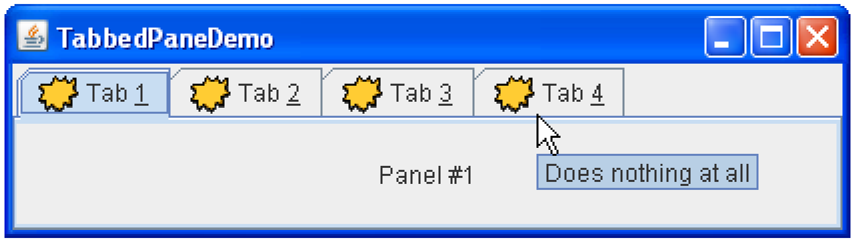
Anonymous class used to create event handler



```
Position=17
n't read beauty
ey'll make you
```

Tabbed pane

Good for managing space



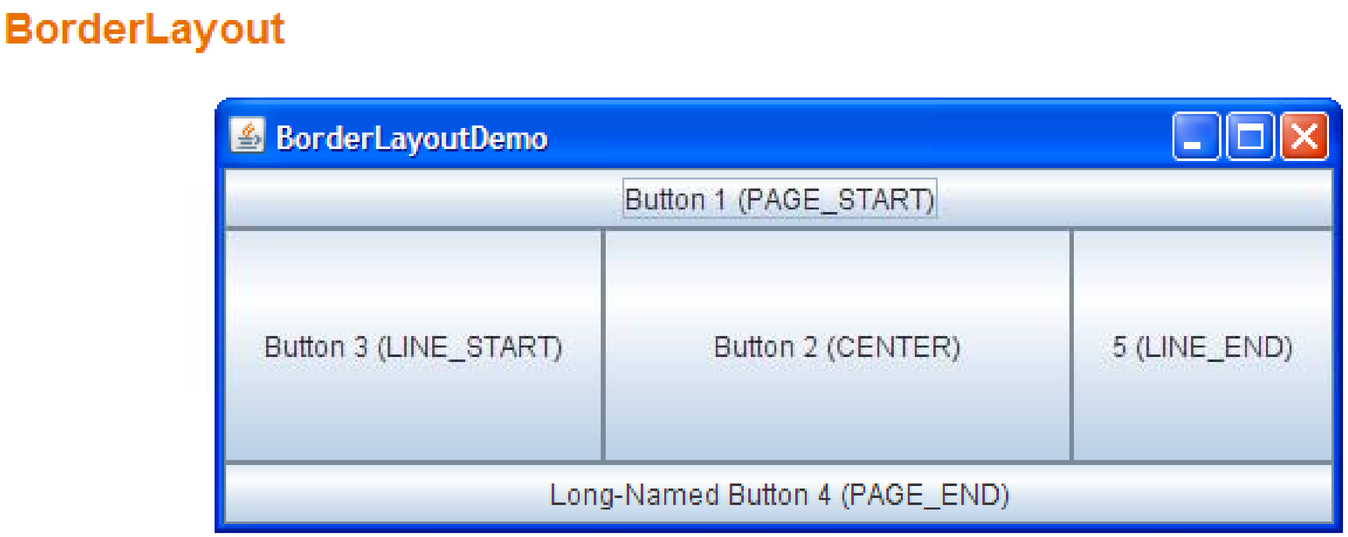
```
JTabbedPane tabbedPane = new JTabbedPane();
ImageIcon icon = createImageIcon("images/middle.gif");

JComponent panel1 = makeTextPanel("Panel #1");
tabbedPane.addTab("Tab 1", icon, panel1,
        "Does nothing");
tabbedPane.setMnemonicAt(0, KeyEvent.VK_1);

...

protected JComponent makeTextPanel(String text) {
JPanel panel = new JPanel(false);
JLabel filler = new JLabel(text);
filler.setHorizontalAlignment(JLabel.CENTER);
panel.setLayout(new GridLayout(1, 1));
panel.add(filler);
return panel;
```

Tab heuristics

- Be sure to create good names for your tabs
  - “Details 1” and “Details 2” is a total failure!
- The end user should have a good intuition for what is under each tab based on its name
  - Similar reasoning applies when naming the entries on a hierarchical menu – user should be able to guess fairly reliably which submenu has their desired option
- It’s OK if some tabs are less packed with fields than others if that results in better tab names
- It’s not great to have so many tabs that it takes multiple rows
  - Consider what else you can do as an alternative (e.g., put rarer options under an “Options...” button; and do you *really* need all these details? – watch for ‘feature creep’)

Layout managers in Swing

- Everything goes through a layout manager in Swing
  - Learning to control them well let’s you get the UI you really want

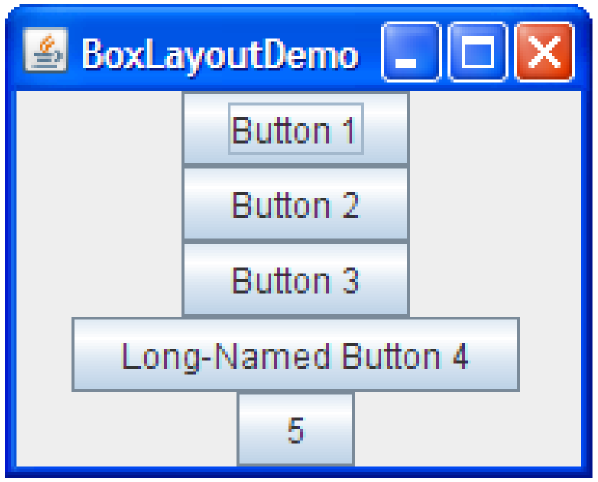


- BorderLayout is default for content pane
  - All extra space is given to ‘CENTER’

From http://java.sun.com/docs/books/tutorial/uiswing/layout/visual.html

Box Layout

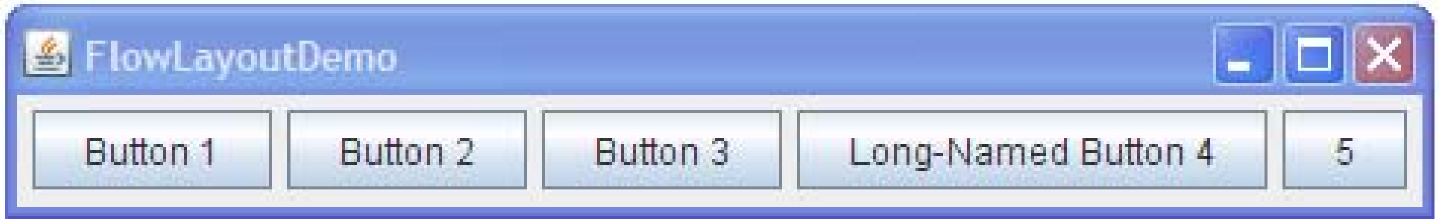
BoxLayout



- Respects components requested max sizes
- Allows you to set alignments

Flow layout

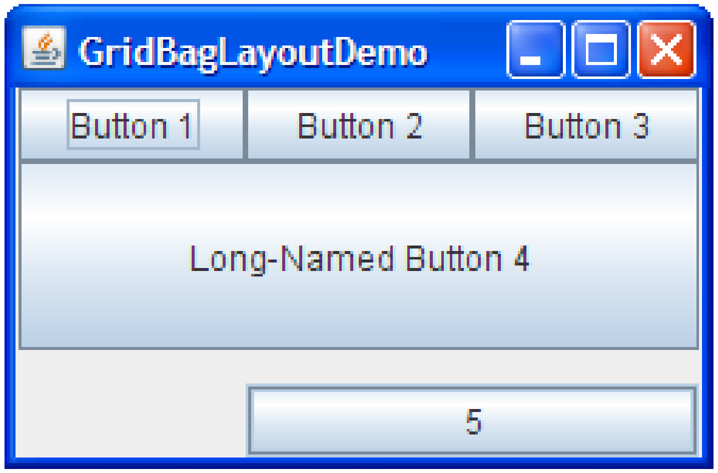
FlowLayout



- Default layout manager for JPanel
- Simply lays out components left to right in order they are added
- Adds another row if its not wide enough to hold all its contents

Grid Layouts

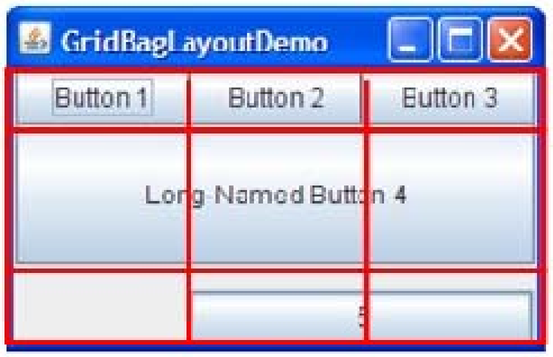
GridBagLayout



- Flexible GridBagLayout lets you specify rows of cells
  - Cells can span columns
  - Rows can have different heights
- Using GridLayout is simpler if you want a uniform table of rows and columns

GridBagLayout code

```
JButton button;
pane.setLayout(new GridBagLayout());
GridBagConstraints c = new GridBagConstraints();

button = new JButton("Button 1");
c.weightx = 0.5;
c.fill = GridBagConstraints.HORIZONTAL;
c.gridx = 0;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 2");
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.5;
c.gridx = 1;
c.gridy = 0;
pane.add(button, c);

button = new JButton("Button 3");
c.fill = GridBagConstraints.HORIZONTAL;
c.weightx = 0.5;
c.gridx = 2;
c.gridy = 0;
pane.add(button, c);
```



Set the layout of a container with a new layout manager – in this case ‘pane’ is the result of frame.getContentPane()

Attributes of the GridBagConstraints object determine position/behaviour of objects once added to the pane

.fill indicates button fills available space; .weightx determines how space is allocated among columns

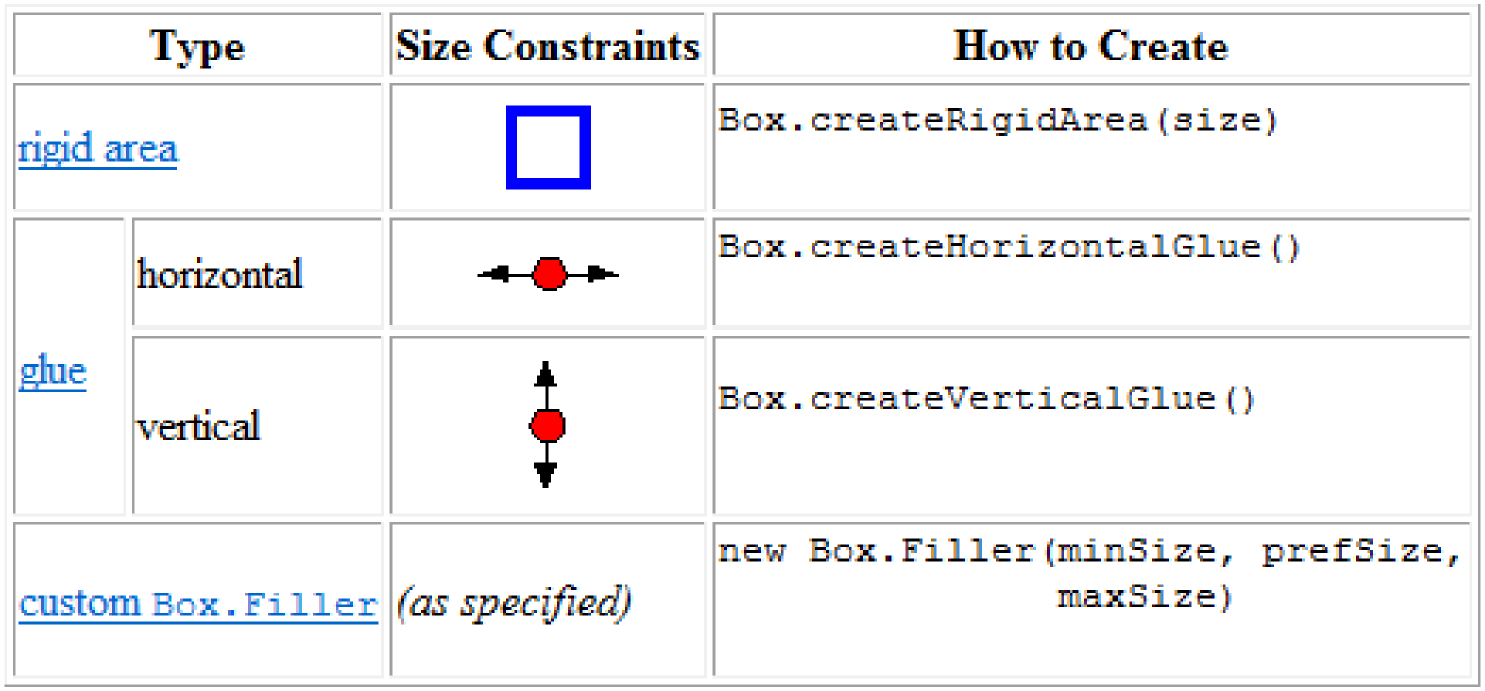
```
button = new JButton("Long-Named Button 4");
c.fill = GridBagConstraints.HORIZONTAL;
c.ipady = 40;      //make this component tall
c.weightx = 0.0;
c.gridwidth = 3;
c.gridx = 0;
c.gridy = 1;
pane.add(button, c);

button = new JButton("5");
c.fill = GridBagConstraints.HORIZONTAL;
c.ipady = 0;       //reset to default
c.weighty = 1.0;   //request any extra vertical space
c.anchor = GridBagConstraints.PAGE_END; //bottom of space
c.insets = new Insets(10,0,0,0);  //top padding
c.gridx = 1;       //aligned with button 2
c.gridwidth = 2;   //2 columns wide
c.gridy = 2;       //third row
pane.add(button, c);
```

Padding makes the object use more space internally
.gridwidth makes the object span columns

Insets create space between object and edges of its cell

Getting it just right: Glue and Rigid Areas

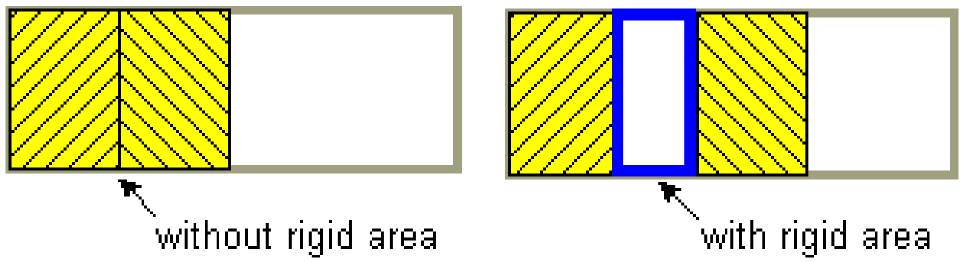
- Don't let the layout managers push you around!
  - Get the alignment your design calls for



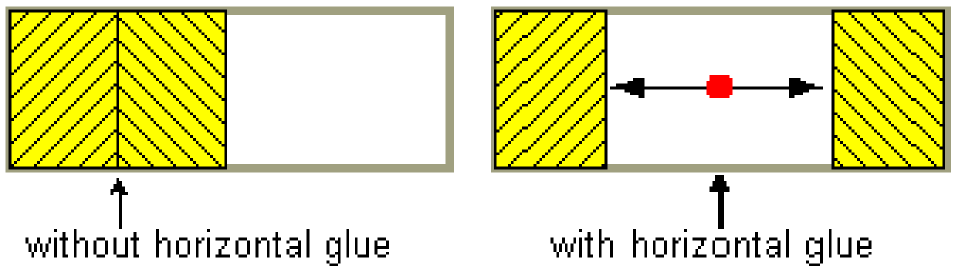
Type		Size Constraints	How to Create
rigid area		□	`Box.createRigidArea(size)`
glue	horizontal	↔	`Box.createHorizontalGlue()`
	vertical	↕	`Box.createVerticalGlue()`
custom `Box.Filler`		*(as specified)*	`new Box.Filler(minSize, prefSize, maxSize)`

Glue and Rigid Areas

```
container.add(firstComponent);
container.add(Box.createRigidArea(new Dimension(5,0)));
container.add(secondComponent);
```

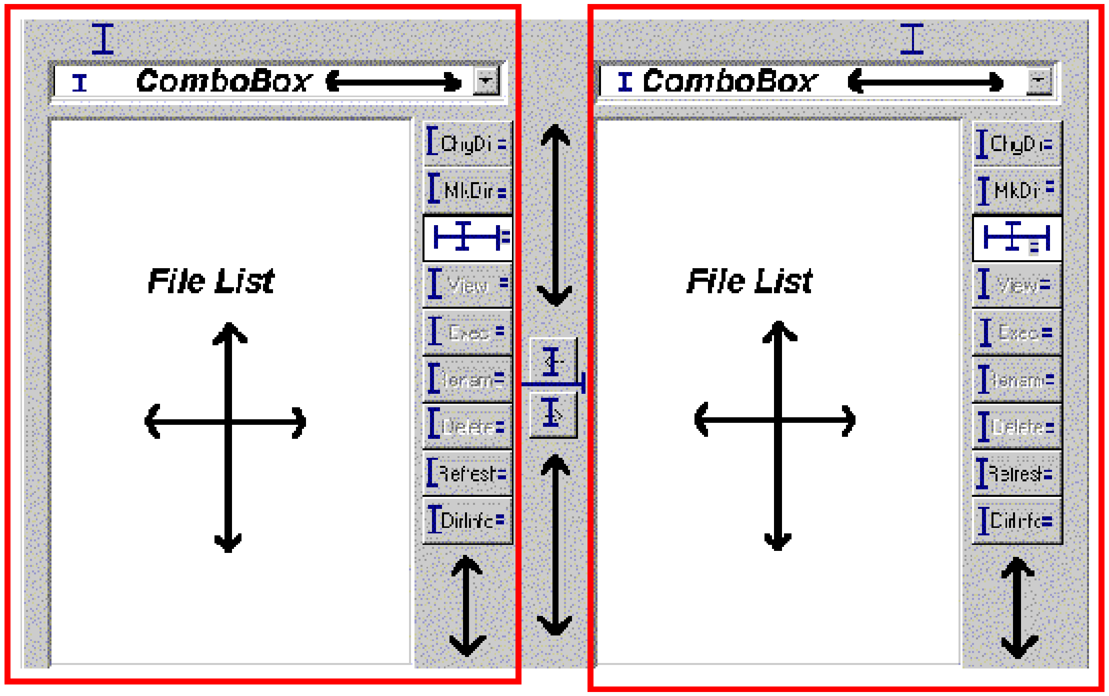


```
container.add(firstComponent);
container.add(Box.createHorizontalGlue());
container.add(secondComponent);
```



- “Glue” expands to fill all the available space between other components

Getting it right



- For sophisticated dialogs you'll often need to nest layout managers (e.g., JPanels with vertical BoxLayouts within a BorderLayout)
  - See "Nesting Layout Managers to Achieve Nirvana" at http://java.sun.com/developer/onlineTraining/GUI/AWTLayoutMgr/shortcourse.html#nesting (it's AWT, but easily transfers to Swing)

Summary

- Swing gives you mechanisms to make ‘conventional’ GUIs
- With effort, you can override default behaviours and make custom layouts and controls, too
- Results from layout managers and event handlers can end up surprising you
- Keep sight of your design
  - Don’t be a slave to your toolkit
  - Then again, know when to compromise rather than fail