

Lecture 19: GUI Programming in Java

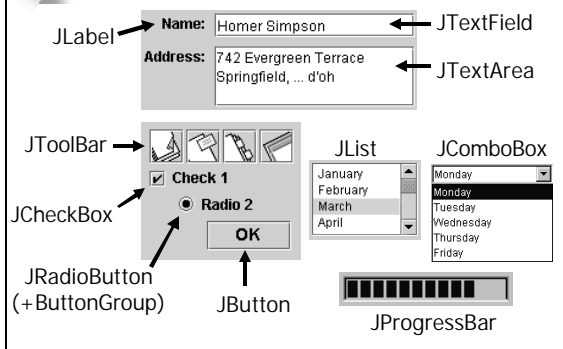
6.170 Lab in Software Engineering
October 23, 2002

Java UI Toolkits: Some History

- AWT (Abstract Windowing Toolkit)
 - Java adapters for Win/MacOS/Motif widgets
- Swing, aka JFC (Java Foundation Classes)
 - Widgets are reimplemented in pure Java
 - Still uses AWT for layout, events, & drawing
- You need both AWT and Swing:


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

Basic Widgets



Attach Listeners to Widgets



```
JButton b = new JButton ("Beep");
b.addActionListener (new BeepAction ());

class BeepAction implements ActionListener
{
    public void actionPerformed(ActionEvent e)
    {
        Toolkit.getDefaultToolkit ().beep ();
    }
}
```

Anonymous Classes

```
JButton b = new JButton ("Beep");
b.addActionListener (new ActionListener () {
    public void actionPerformed(ActionEvent e)
    {
        Toolkit.getDefaultToolkit ().beep ();
    }
});
```

required!
↓

- Java compiler generates a name for the anonymous class, e.g. MyWindow\$7

Various Kinds of Listeners

- ActionListener


```
void actionPerformed (ActionEvent e);
```

 - fired by all buttons, menu items, JComboBox, JTextField (on Enter key)
- ItemListener

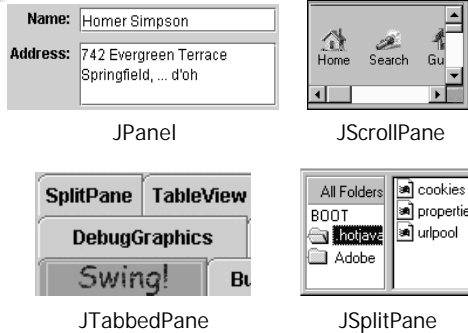

```
void itemStateChanged (ItemEvent e);
```

 - fired by JCheckBox, JRadioButton, JComboBox
- ListSelectionListener


```
void valueChanged (ListSelectionEvent e);
```

 - fired by JList, JTable

Basic Containers



Using Containers

- For structured containers, just add components

```
JScrollPane scroller = new JScrollPane ();
scroller.add (new JTextArea ());
```

```
JSplitPane splitter = new JSplitPane
(JSplitPane.HORIZONTAL_SPLIT);
splitter.setLeftComponent (new JTree ());
splitter.setRightComponent (scroller);
```

```
JTabbedPane tabs = new JTabbedPane ();
tabs.add ("File Browser", splitter);
tabs.add ("Spreadsheet", new JTable ());
```

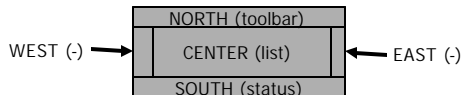
- Unstructured containers (JPanel) need **layout manager**

Layout Managers

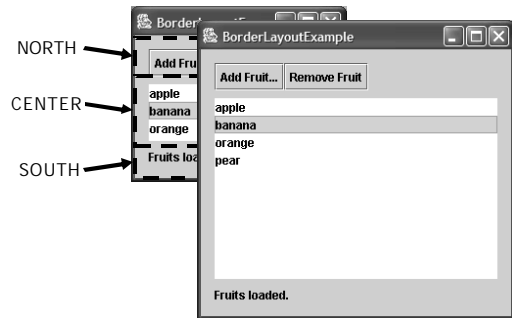
- Layout manager sets the sizes and positions of components in a container

- Example: BorderLayout

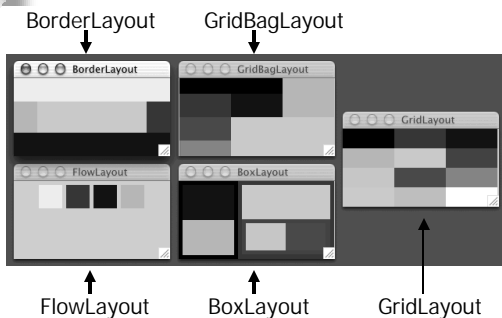
```
JComponent toolbar, list, status = ...;
JPanel panel = new JPanel ();
panel.setLayout (new BorderLayout ());
panel.add (toolbar, BorderLayout.NORTH);
panel.add (list, BorderLayout.CENTER);
panel.add (status, BorderLayout.SOUTH);
```



Layout Adjusts to Space Available



Visual Guide to Layout Managers



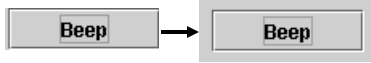
Choosing A Layout Manager

- BorderLayout:** good for main windows
 - e.g., toolbar N, main pane CENTER, status bar S
- BoxLayout:** multi-purpose
 - Use nested panels to lay out arrays of widgets
 - Look at Box (JPanel with BoxLayout included)
- FlowLayout:** mostly useless
 - Horizontal BoxLayout is better for most purposes
- GridLayout:** good for uniform button arrays
 - But stick to 1D (1xN or Mx1), not 2D (MxN)
- GridBagLayout:** the most powerful
 - It's the only way to align a nonuniform array of labels and widgets both vertically and horizontally
 - Painful to use; see Swing tutorial for examples

Loosen Up Your Layout

- Any Swing component may have a border


```
JButton b = new JButton ("Beep");
      b.setBorder (new EmptyBorder (5,5,5,5));
```



- Even better:


```
b.setBorder(BorderFactory.createEmptyBorder
              (5,5,5,5));
```
- Other Border classes exist too (e.g. Etched, Beveled, Titled), but less is more

Windows: Top-level Containers



- Ignore JWindow; use JFrame & JDialog
- Can't put window in any other container
- Can't add widgets directly to window
 - Call `getContentPane()` to get a container

Standard Window Idioms

- Create window


```
JFrame f = new JFrame ("My Program");
```
- Add widgets to its content pane

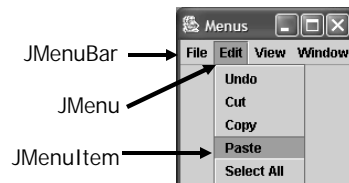

```
Container content = f.getContentPane ();
      content.setLayout (...);
      content.add (...);
```
- Set its behavior when user closes it


```
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      // calls System.exit. Use only for main window!
```
- Run layout and make window appear


```
f.pack ();
      f.setVisible (true);
```
- Destroy window when you're done with it


```
f.dispose (); // not needed with EXIT_ON_CLOSE
```

Frames Can Have Menu Bars



- JMenuBar and JMenu are containers
- Attach ActionListeners to JMenuItem
- Menu bar is separate from content pane

Dialog Boxes

- JDialogs can be modal or modeless
 - Modal = user can't click on main window
- JOptionPane provides simple modal dialogs
 - `showMessageDialog` for error & info messages:


```
JOptionPane.showMessageDialog
      (mainWindowFrame,
       "Can't open file " + filename,
       "Error Opening File",
       JOptionPane.ERROR_MESSAGE);
```
 - `showConfirmDialog` for yes/no/cancel or OK/cancel questions
 - `showInputDialog` for one-line input with OK/cancel

Periodic Tasks

- Use a timer to run a periodic task

```
import javax.swing.Timer;
// not java.util.Timer!

Timer t = new Timer (1000 /* milliseconds */,
new ActionListener () {
public void actionPerformed(ActionEvent e) {
    autoSave ();
}
});
t.start (); // repeats every 1s until stopped
```

Background Tasks

- Timer task must be brief, because the user interface is frozen while it runs
- Long-running tasks (both automatic and user-invoked) belong in a separate **thread**
- Programming with multiple threads is tricky because of **race conditions**
 - While one thread is mutating an object (rep invariant is temporarily invalid), another thread tries to look at the object (boom! broken rep)
 - Generally solved with synchronization, but this is beyond the scope of 6.170. However...

Background Tasks in Swing

- Create a **SwingWorker** to run the task
 - `construct()` runs in background thread
 - Can't access UI objects; mutable data shared with main thread must be protected by synchronization
 - `finished()` runs in main thread
 - Applies the results of `construct()` to the UI
 - Not in Java library, but in Swing tutorial
- Warnings
 - Multiple threads can have nasty bugs
 - `QuoteServer` is not thread-safe

Other Stuff

- Icons & images
 - Don't bother – labels often work better
 - It's hard to draw effective icons
 - Look on web for public-domain collections (or Sun's icon repository in references)
- Painting
 - Don't bother yet (but it will be necessary for final project)
 - Encapsulate painting in a new widget

Hints on Code Organization

- Class for every window or dialog box

```
class PokerGame extends JFrame {...}
```
- Classes for reusable panels

```
class HandDisplay extends JPanel {...}
```
- Factory methods for parts of a window

```
JComponent makeBookmarkPane () {...}
```
- Private fields for anonymous listeners

```
private final ActionListener saveAction
= new ActionListener () {...};
```

Further Reading

- Swing Tutorial
 - <http://java.sun.com/docs/books/tutorial/uiswing/>
- Java Look & Feel Graphics Repository
 - <http://developer.java.sun.com/developer/techDocs/hi/repository/>
 - These icons are not in the public domain – see the site for licensing details

Other Java UI Toolkits

- IBM SWT (Standard Widget Toolkit)
 - <http://www.eclipse.org>
 - Platform-specific widgets where available, pure Java emulation where necessary
- subArctic
 - http://www.cc.gatech.edu/gvu/ui/sub_arctic/
 - Constraint propagation, animation, output effects
- Jazz
 - <http://www.cs.umd.edu/hcil/jazz/>
 - Zoomable interfaces