

Introduction to Programming Languages and Techniques



Swing II

11/6/09

Office Hours

- My office hours have changed (again):

Monday 1:30-3 (as always)

Friday 4:30-5:30 (new)

WINDOWLISTENERS

Recall...

... our little demo program from last time

Problem

- When window is closed, program does not terminate.
- How to fix it?

The Easy Way

- `frame.setDefaultCloseOperation (EXIT_ON_CLOSE) ;`

More flexible: WindowListener

- Closing windows creates a **WindowEvent**
 - React to that event by terminating program
- A *WindowListener* handles this event and others

```
void windowActivated(WindowEvent e)
void windowClosing(WindowEvent e)
void windowClosed(WindowEvent e)
void windowDeActivated(WindowEvent e)
void windowDeiconified(WindowEvent e)
void windowIconified(WindowEvent e)
void windowOpened(WindowEvent e)
```

Terminator class

```
class Terminator implements WindowListener {

    public void windowClosing(WindowEvent e) {
        Window w = e.getWindow();
        // delete top-level frame, release OS resources,
        // generate WindowClosed event
        w.dispose();
    }

    public void windowClosed(WindowEvent e) {
        // terminate the application
        System.exit(0);
    }

    // other five methods do nothing
    void windowActivated(WindowEvent e) {}
    void windowDeActivated(WindowEvent e) {}
    void windowDeiconified(WindowEvent e) {}
    void windowIconified(WindowEvent e) {}
    void windowOpened(WindowEvent e) {}
}
```

Register WindowListener

```
public OnOffSwitch () {
    super("On/Off Switch"); // frame title

    this.addWindowListener(new Terminator());

    // create button and set its colors
    JButton button = new JButton("On/Off");
    button.setForeground(Color.black);
    button.setBackground(Color.white);

    // create and register button's listener:
    button.addActionListener(new Switcher());

    // add button to JFrame's content pane:
    this.getContentPane().add(button);
}
```

Problem

- A window listener must implement all 7 methods of **WindowListener** interface
- We only care about 2 of those methods
→ lots of annoying boilerplate
- Solution: **WindowAdapter**

Adapter classes:

- Swing provides a collection of abstract *event adapter classes*
- These adapter classes implement listener interfaces with empty, do-nothing methods
- To implement a listener class, we *extend* an adapter class and override just the methods we need

Terminator class (II)

```
class Terminator extends WindowAdapter {  
  
    public void windowClosing(WindowEvent e) {  
        Window w = e.getWindow();  
        w.dispose();  
    }  
  
    public void windowClosed(WindowEvent e) {  
        System.exit(0);  
    }  
  
}
```

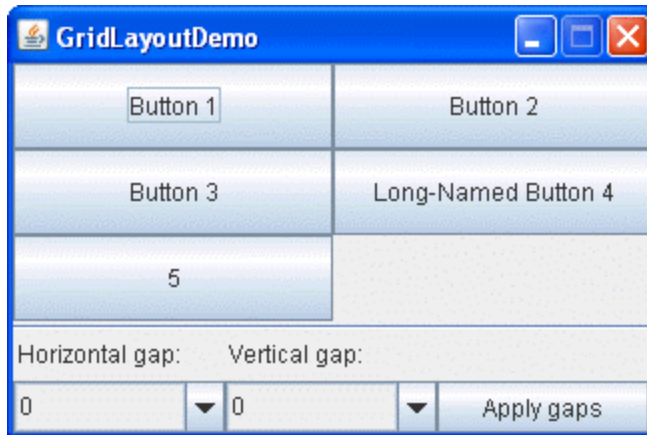
LAYOUTS

Flow Layout



- Items just “flow”
- Layout manager places them one after another
- (These and following examples taken from Java’s “The Swing Tutorial”)

Grid Layout

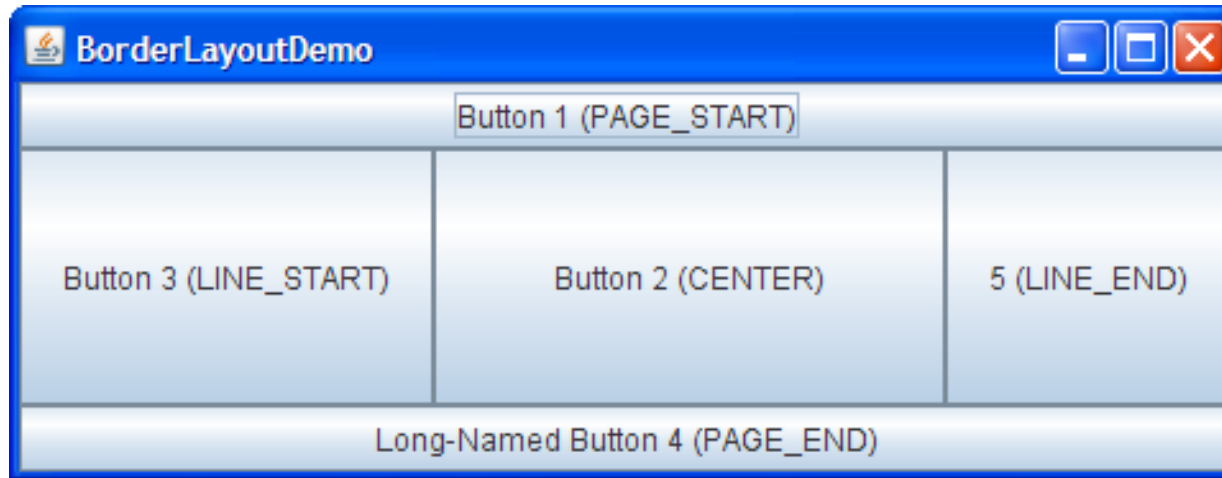


This JPanel contains a Grid (2x3)

This JPanel contains another Grid (3x2)

- Places components in a grid of cells.
- Each component takes all the available space within its cell,
- Each cell is exactly the same size.

Border Layout



- New “Position Relative” Names as of Java 1.4
 - Corresponds to NORTH, SOUTH, EAST, WEST, CENTER
 - Old names still work

More Sophisticated Managers

- Include:
 - [BoxLayout](#)
 - [CardLayout](#)
 - [GridBagLayout](#)
 - [GroupLayout](#)
 - [SpringLayout](#)
- These are harder to use when coding by hand
- Layout tools such as NetBeans provide GUIs that write code for GUIs!

Time for some more demos...

