CDE Services

Consulting
Development
Education

Creating a GUI with Swing

Jaroslav Porubän, Peter Václavík

CDE Services

User Interface

- **User interface** is the part of a computer and its software that people can see, hear, touch, talk to, or otherwise understand or direct
- A **user interface** is a collection of techniques and mechanisms to interact with something
- Proper interface design will provide a mix of well-designed **input and output mechanisms**
- User interface design is a subset of a field of study called **human-computer interaction** (HCI)
- **Graphical User Interface** – primary interaction mechanism is a **pointing device** of some kind

Wilbert O. Galitz: The Essential Guide to User Interface Design: An Introduction to GUI Design Principles and Techniques, 2nd Edition, John Wiley & Sons, 2002, 736 pp.

CDE Services

Swing and AWT

- Collection of APIs for developing interactive GUI applications
- **AWT** (`java.awt`) – Abstract Window Toolkit
  - Some AWT components use native code
  - Platform-dependent look
- **Swing toolkit** (`javax.swing`)
  - Written entirely using the Java programming language
  - Platform-independent
  - Ensures applications deployed across different platforms have the same appearance – look and feel

CDE Services

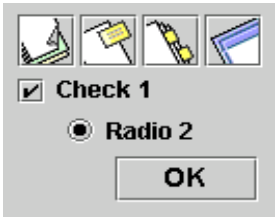
GUI Components

- A **component** is an object having a **graphical representation** that can be displayed on the screen and that **can interact with the user**
- Parent of GUI Swing components classes is class `JComponent`

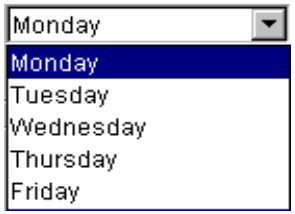
Sun Microsystems: Java Look and Feel Design Guidelines. 2nd Edition, Addison Wesley Professional, 2001, 416 pp.
Sun Microsystems: Java Look and Feel Design Guidelines: Advanced Topics. Addison Wesley Professional, 2002, 200 pp.
Mauro Marinilli: Professional Java User Interfaces. Wiley, 2006, 668 pp.

CDE Services

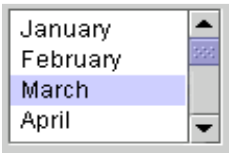
GUI Components



JButton, JToggleButton,
JCheckBox, JRadioButton

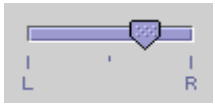


JComboBox



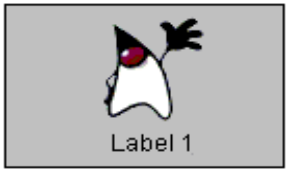
JList

JMenu



JSlider

JTextField
JPasswordField

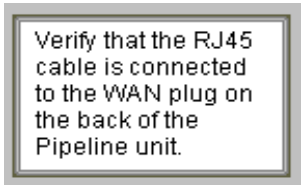


JLabel

JProgressBar

JToolTip

JTable

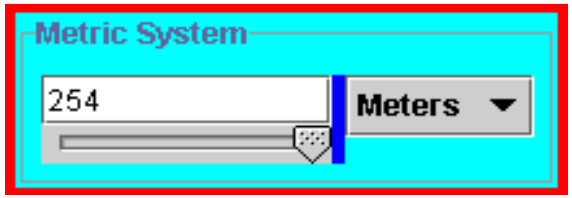


JTextArea
JTextPane, JEditorPane

JTree

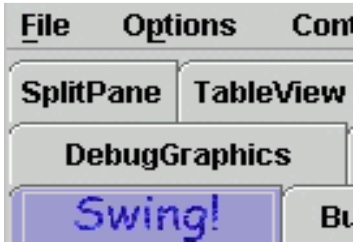
GUI Container

- **Container** is a component that can contain other components



JPanel

JScrollPane



JTabbedPane

JToolBar

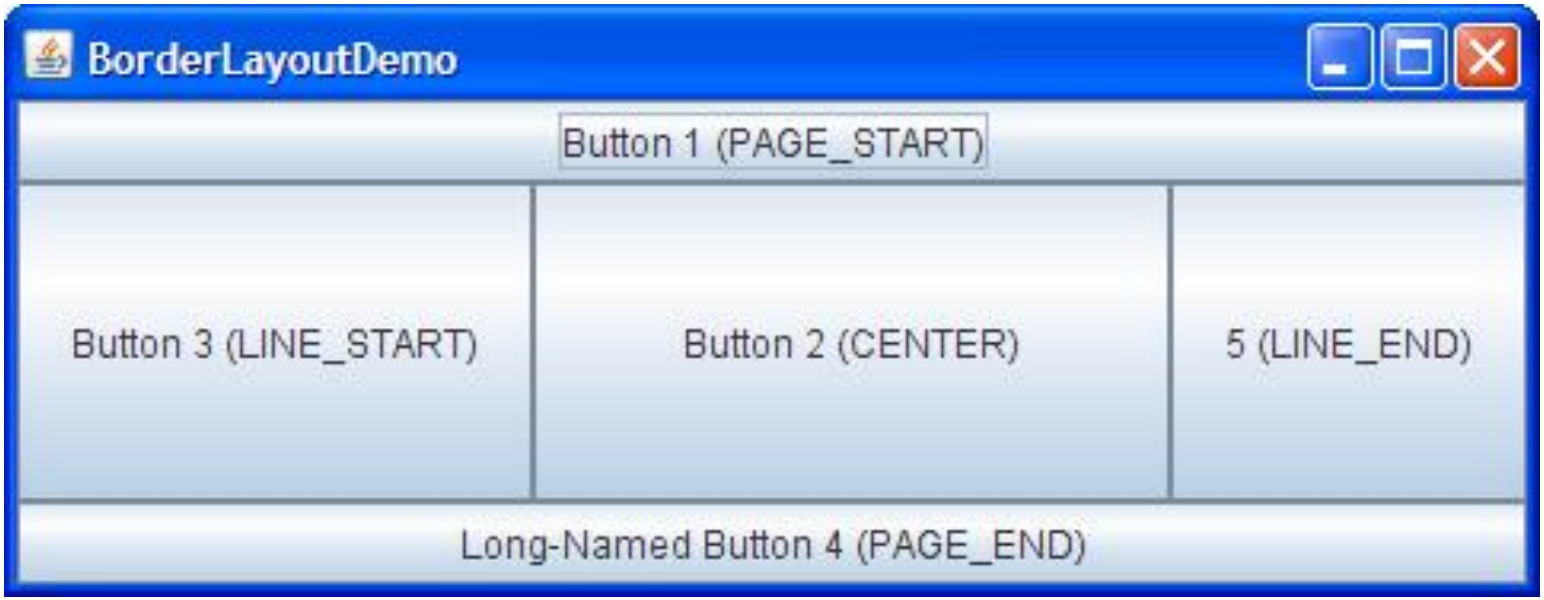
JSplitPane

CDE Services

Layout Manager

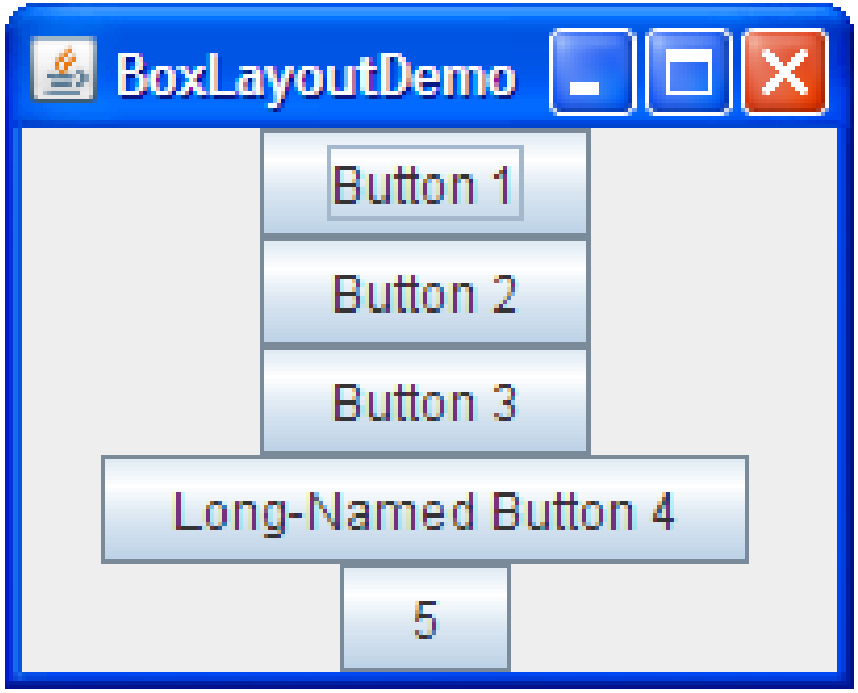
- A **layout manager** is an object that **determines the size and position of the components** within a **container**
- Layouts in Swing
  - `BorderLayout`
  - `BoxLayout`
  - `FlowLayout`
  - `GridBagLayout`
  - `GridLayout`
  - `GroupLayout`

BorderLayout



BorderLayoutDemo.java

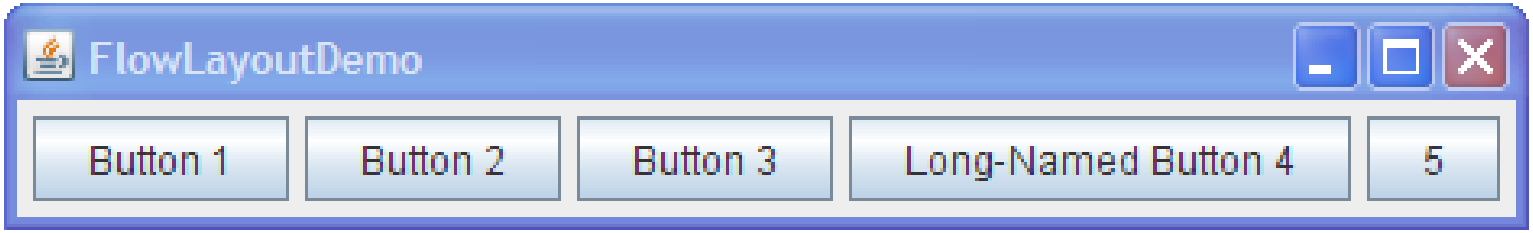
BoxLayout



BoxLayoutDemo.java

CDE Services

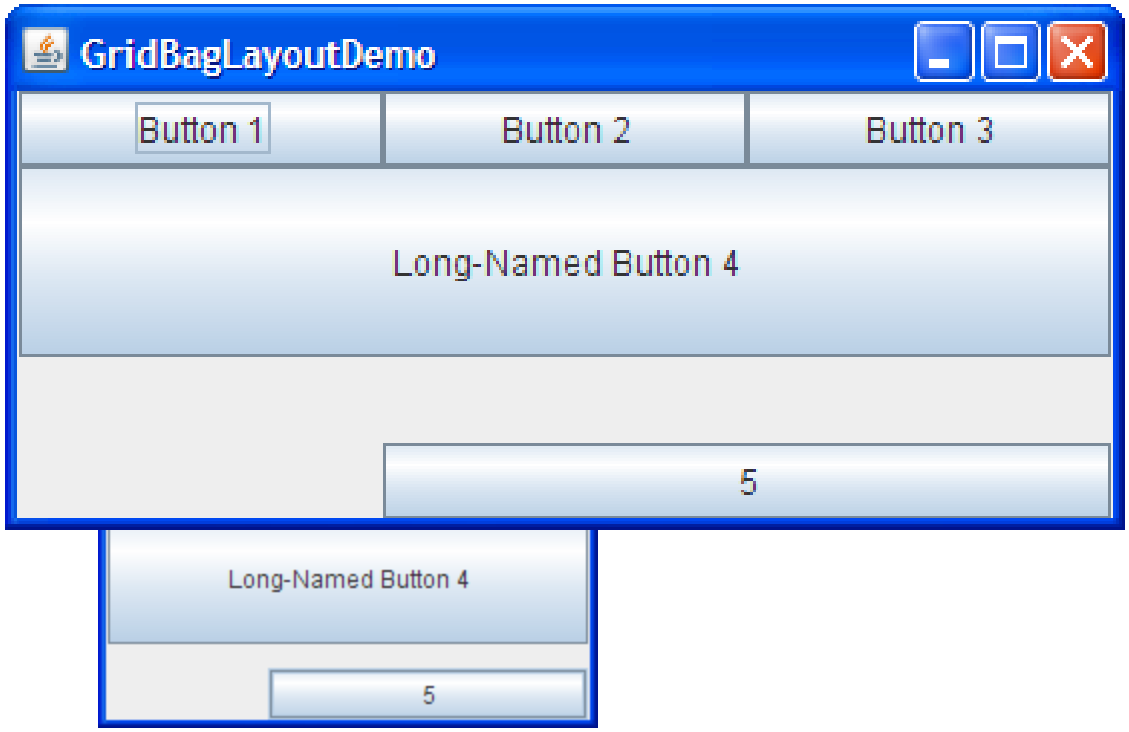
FlowLayout



FlowLayoutDemo.java

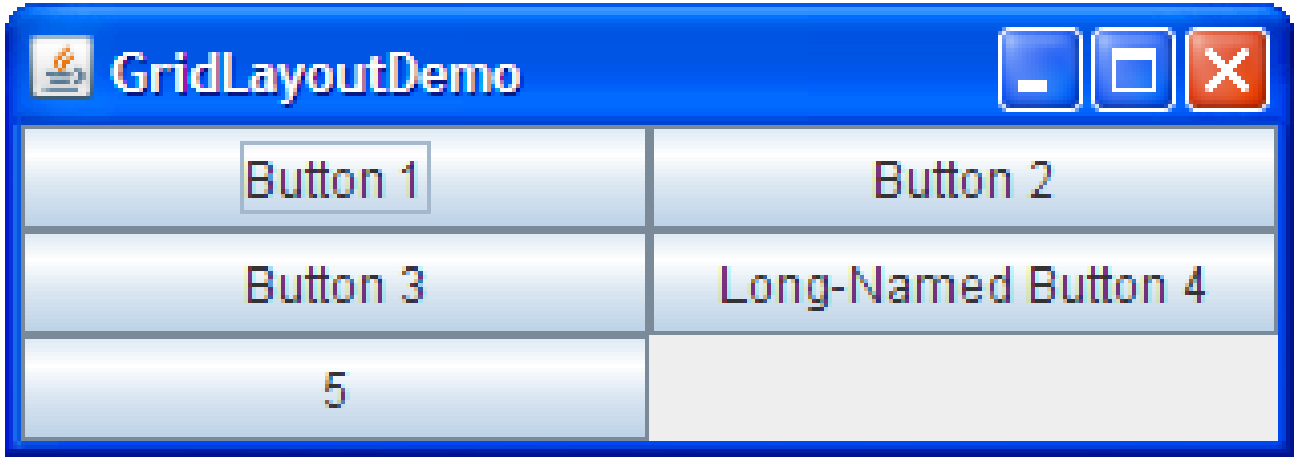
CDE Services

GridBagLayout



GridBagLayoutDemo.java

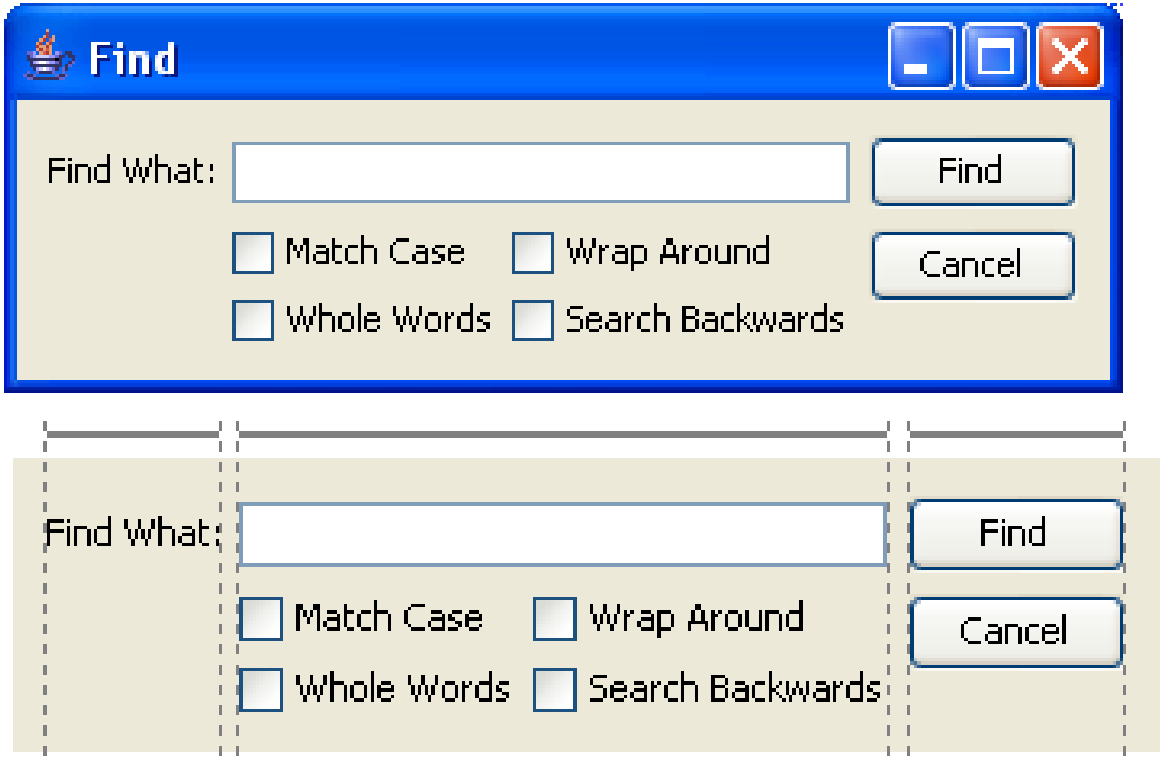
GridLayout



GridLayoutDemo.java

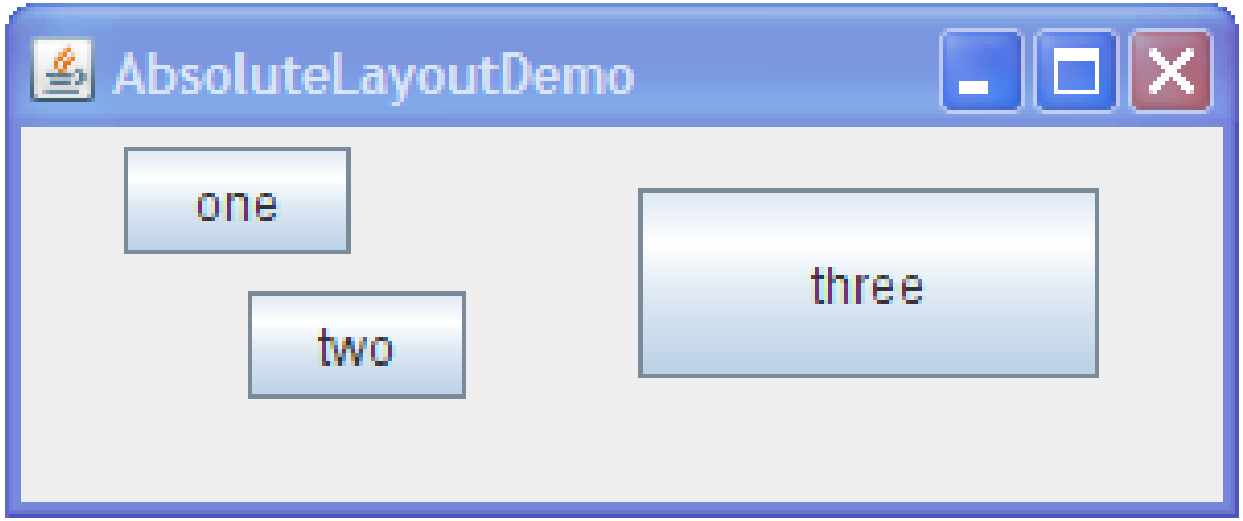
GroupLayout

- Basic layout in Netbeans GUI Builder
- Since JDK 1.6



Find.java

Without a Layout Manager (Absolute Positioning)

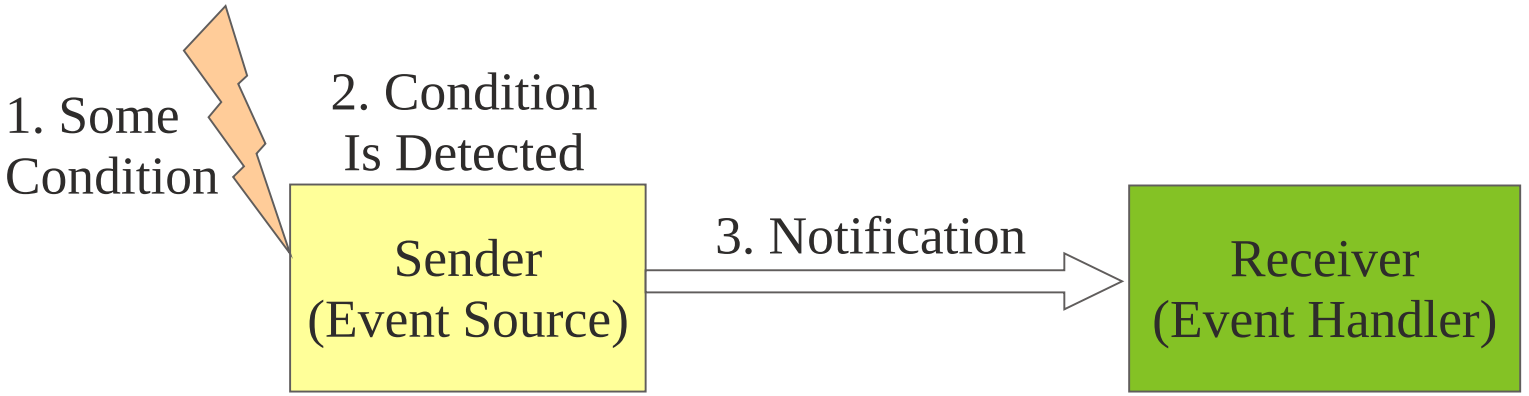


AbsoluteLayoutDemo.java

CDE Services

Event Driven Programming

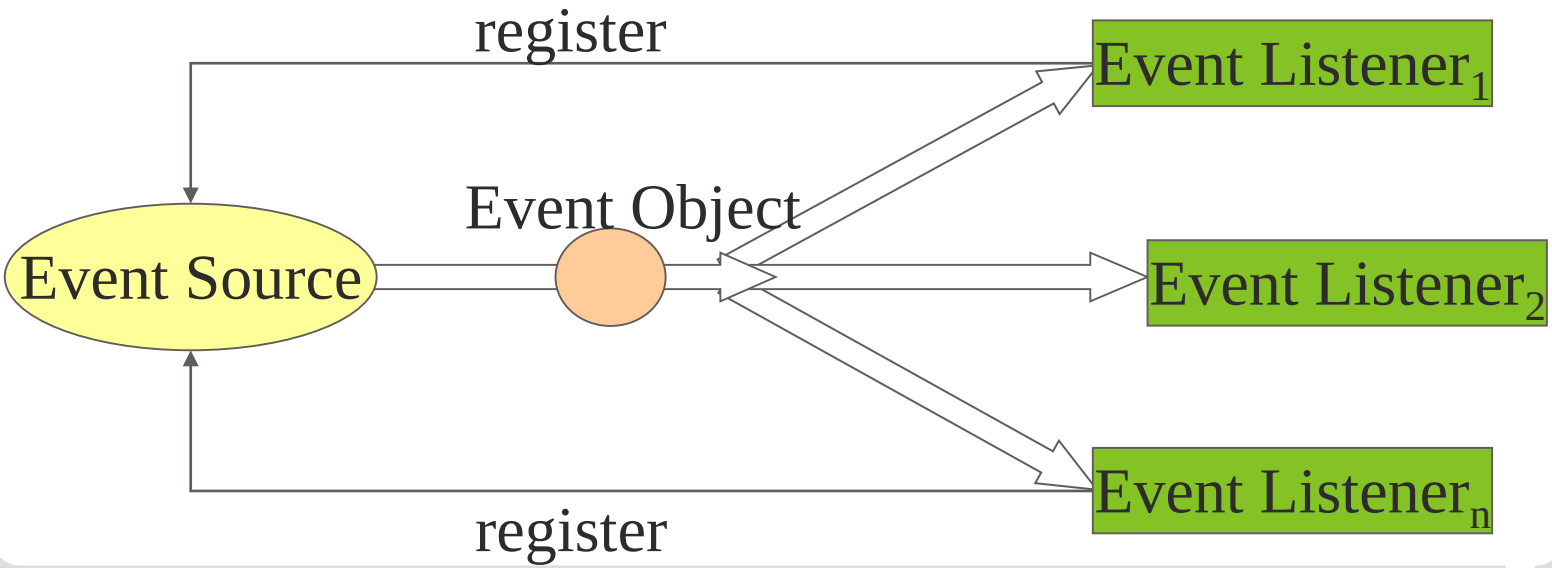
- The flow of the program is controlled/determined by **events**
- An **event** is a detectable condition that can trigger a notification
- A **notification** is an event-triggered signal sent to a run-time–defined recipient



Ted Faison: Event-Based Programming: Taking Events to the Limit. Apress, 2006, 700 pp.

Events in Java

- **Events** are objects sent from **event sources** to **event listeners**
- **Listeners** are **added** to event **sources**
- **Listeners handle events** by providing methods that are invoked when the **event source fires the event**



CDE Services

Event Objects and Listeners

Event Type	Event Object	Event Listener
component-defined action occurred	`ActionEvent`	`ActionListener`
window has changed its status	`WindowEvent`	`WindowListener`
mouse action occurred in a component	`MouseEvent`	`MouseListener` `MouseMotionListener`

CDE Services

Event Objects and Listeners

Event Type	Event Object	Event Listener
gained or lost the input focus	`FocusEvent`	`FocusListener`
keystroke occurred in a component	`KeyEvent`	`KeyListener`
component moved, changed size, or changed visibility	`ComponentEvent`	`ComponentListener`

CDE Services

Event Handling Examples (ActionListener)

```
package java.awt.event;

public interface ActionListener extends
  EventListener {

  /**
   * Invoked when an action occurs.
   */
  public void actionPerformed(ActionEvent e);
}
```

CDE Services

Event Handling I

```
public class EventDemo1 extends JFrame
  implements ActionListener {

  public EventDemo1() {
    JButton button = new JButton("Click Me!");
    button.addActionListener(this);
    //...
  }

  public void actionPerformed(ActionEvent e) {
    JOptionPane.showMessageDialog(this,
      "Clicked!");
  }
}
```

EventDemo1.java

CDE Services

Event Handling II (Inner class)

```
public class EventDemo2 extends JFrame {

    public EventDemo2() {
        JButton button = new JButton("Click Me!");
        button.addActionListener(new Action());
        //...
    }

    private class Action implements ActionListener {
        public void actionPerformed(ActionEvent e) {
                JOptionPane.showMessageDialog(
                        EventDemo2.this, "Clicked!");
        }
    }
}
```

EventDemo2.java

Event Handling III (Anonymous class)

```
public class EventDemo3 extends JFrame {

 public EventDemo3() {
    JButton button = new JButton("Click Me!");
    button.addActionListener(
       new ActionListener(){
      public void actionPerformed(ActionEvent e){
        JOptionPane.showMessageDialog(
                    EventDemo3.this, "Clicked!");
       }
    });
    //...
  }
}
```

EventDemo3.java