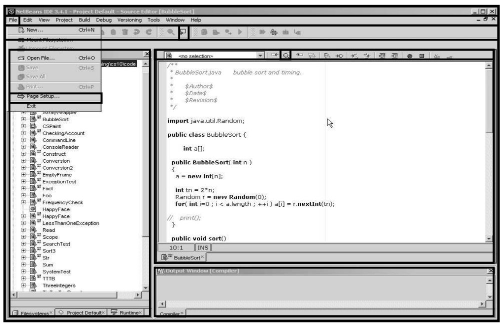
CS202 Java Object Oriented Programming

GUI Programming – Introduction

Chengyu Sun
California State University, Los Angeles

GUI



```
/**
 * BubbleSort.java      bubble sort and timing.
 *
 *     $Author$
 *     $Date$
 *     $Revision$
 */

import java.util.Random;

public class BubbleSort {

    int a[];

  public BubbleSort( int n )
  {
   a = new int[n];

   int tn = 2*n;
   Random r = new Random(0);
   for( int i=0 ; i < a.length ; ++i ) a[i] = r.nextInt(tn);

//   print();
  }

  public void sort()
```

GUI Components

- Widgets
  - Windows, menus, toolbars, buttons, label, text components, lists, …
- Widgets libraries (APIs)
  - Windows – MFC
  - XWindow – XT, GTK, QT
  - Java AWT and Swing
    - Cross platform
    - Slower

Swing GUI Components

- A Visual Index to the Swing Component
  - http://java.sun.com/docs/books/tutorial/uiswing/components/components.html

GUI Programming

- Create a container
- Create components
- Add components to the container
- Handle events

What You Should Know

- General knowledge about what GUI components Swing provide
- Look up documentations for details about specific components
  - SUN's Java Tutorial
    - Using Swing Components
  - Java API documentation
  - Textbook
- Assemble and layout components together
- Event handling

Display An Empty Window

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

public class EmptyFrame extends JFrame {

    public EmptyFrame()
    {
        super( "An Empty Window" );

        setSize( 400, 400 );
        setLocationRelativeTo( null );
        setDefaultCloseOperation( EXIT_ON_CLOSE );
    }

    public static void main( String args[] )
    {
        EmptyFrame f = new EmptyFrame();
        f.setVisible( true );
    }
}
```

Panel

- A general-purposed container
- Common usage
  - A place holder for layout purposes
  - Drawing area for customized paints
- Important methods
  - JPanel()
  - **add(Component c)**
  - remove(Component c), removeAll()
  - paint(Graphics g)

Buttons and Labels

- JButton
  - JButton(String s)
  - String getText()
  - void setText(String s)
  - void setEnabled(boolean b)
- JLabel
  - JLabel(String s)
  - String getText()
  - void setText(String s)

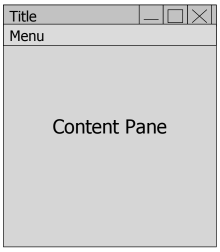
Text Field and Text Area

- JTextField is for single-line text input
- JTextArea is for multi-line text input
- Both inherits from JTextComponent
  - getText(), setText(String s), getSelectedText()
  - setEditable(boolean b), isEditable()
- JTextField(), JTextField(int cols), JTextField(String s)
- JTextArea(), JTextArea(String s)

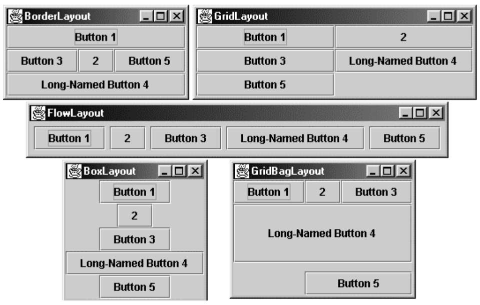
Adding Component

- All components must be added to the content pane, except the menu bar
- getContentPane()

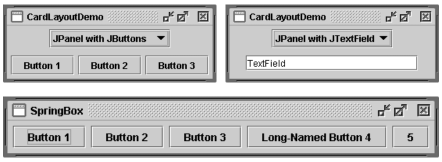
Container cp = getContentPane();
cp.add(*some_component*);

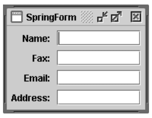


Layouts



More Layouts





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

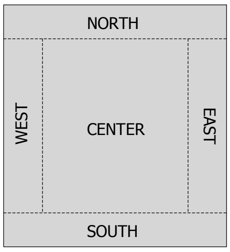
Border Layout

- ◈ Default layout for applet, content pane, and dialog box

```
cp = getContentPane();
cp.setLayout( new BorderLayout() );

JButton b1 = new JButton(“Button1”);
JButton b2 = new JButton(“Button2”);

cp.add( b1, BorderLayout.NORTH );
cp.add( b2, BorderLayout.CENTER );
```


NORTH

WEST

CENTER

EAST

SOUTH

Flow Layout

- ◈ Adds components from left to right
- ◈ Starts a new row if necessary

```
cp = getContentPane();
cp.setLayout( new FlowLayout() );

JButton b1 = new JButton(“Button1”);
JButton b2 = new JButton(“Button2”);

cp.add( b1 );
cp.add( b2 );
```

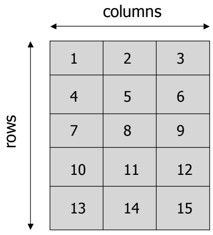
Grid Layout

- ◈ Either rows or cols can be zero
- ◈ All components are of equal size

```
cp = getContentPane();
cp.setLayout( new GridLayout(2,3) );

JButton b1 = new JButton(“Button1”);
JButton b2 = new JButton(“Button2”);

cp.add( b1 );
cp.add( b2 );
```


columns

rows

1	2	3
4	5	6
7	8	9
10	11	12
13	14	15

Box Layout

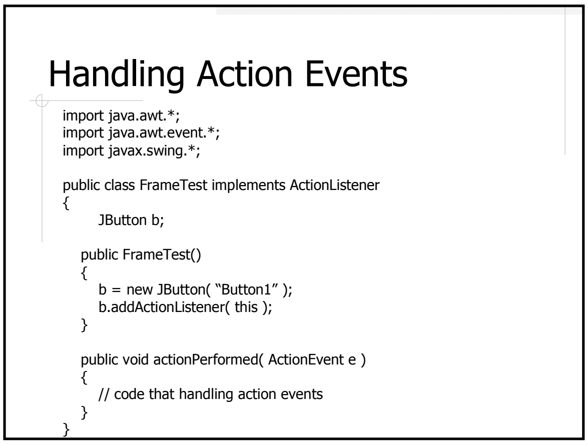
- ◈ Lay out components in one row or one column
- ◈ Can do vertical layout (vs. FlowLayout)
- ◈ Do not force components to be equal size (vs. GridLayout)
- ◈ Constructor
  - BoxLayout(*Container c*, int axis)

```
cp = getContentPane();
cp.setLayout( new BoxLayout(cp, BoxLayout.X_AXIS) );

JButton b1 = new JButton(“Button1”);
cp.add( b1 );
```

Gridbag Layout

- ◈ The most flexible layout, and
- ◈ The most difficult to use
- ◈ Often can be “simulated” with a combination of simpler layouts



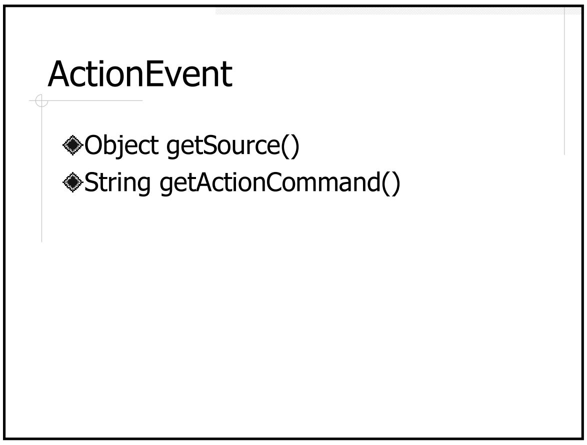
Handling Action Events

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

public class FrameTest implements ActionListener
{
        JButton b;

    public FrameTest()
    {
        b = new JButton( "Button1" );
        b.addActionListener( this );
    }

    public void actionPerformed( ActionEvent e )
    {
        // code that handling action events
    }
}
```



ActionEvent

- Object getSource()
- String getActionCommand()