Programming Mobile Devices

J2SE GUI

University of Innsbruck

WS 2009/2010

thomas.strang@sti2.at

STI · INNSBRUCK

Graphical User Interface (GUI)

- Why is there more than one Java GUI toolkit?
- AWT – “write once, test everywhere (WOTE)” instead of “write once, run everywhere (WORE)
- SWING – solve AWT shortcomings

High-Level: Abstract Windows Toolkit (AWT)

- The original Java GUI toolkit
- Comes standard with every version of Java SE/EE
- It is very stable → everywhere were you find a Java runtime environment, your program will work (in theory)
- Simple: limited GUI components, layout managers, and events (e.g. not supported: Tables, Trees, Progress Bars, etc.)

The basic AWT Class tree

All in java.awt package

```
Object
   CheckboxGroup
   *Component                          "*" indicates abstract
        Button
        Canvas
        CheckBox
        Choice
        Container
                Panel
                        Applet
                ScrollPane
                Window
                        Dialog
                        Frame
        Label
        List
        TextComponent
                TextArea
                TextField
   MenuComponent
        MenuItem
                CheckboxMenuItem
                Menu
                        PopupMenu
```

Steps to create any GUI application or applet

1. Compose a GUI by adding components to a `Container` object
2. Setup event handlers to respond to user interactions with the GUI
3. Display the GUI (automatically done for applets, you must explicitly do this for applications).

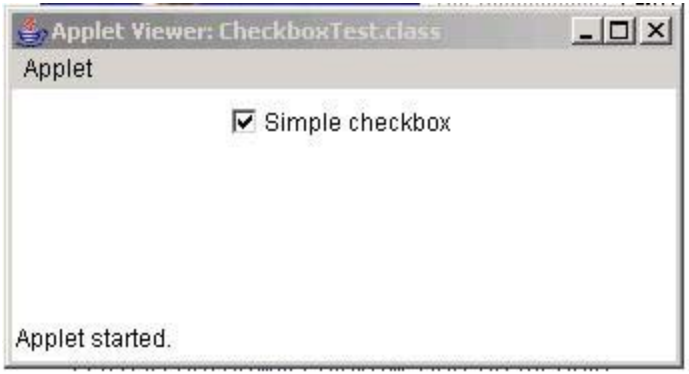
Simple AWT Application

```
import java.awt.*;
public class AButtonPanel extends Frame {
    public AButtonPanel() {
        // STEP 1: Compose the GUI
        super("BasicApplication Title");
        Button uselessButton = new Button("Useless Button");
        add(uselessButton);
        // STEP 2: Setup Event handlers
        addWindowListener(
          new WindowAdapter() {
                    public void windowClosing(WindowEvent e) {
                                        setVisible(false); dispose(); System.exit(0);
                                }
    });}
    public static void main(String[] args) {
          AButtonPanel panel = new AButtonPanel();

          // STEP 3: Display the GUI (automatic in applets)
          panel.setSize(100, 100);
          panel.setVisible(true);
}}
```

Selected AWT Components (1)

- Checkbox

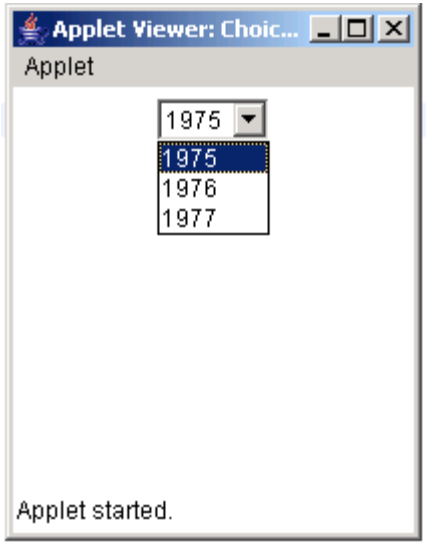
```
import java.applet.Applet;
import java.awt.Checkbox;

public class CheckboxTest extends Applet {
    public void init() {
          Checkbox checkbox = new Checkbox("Simple checkbox");
          checkbox.setState(true);
          add(checkbox);
    }
}
```



Selected AWT Components (2)

- Choice

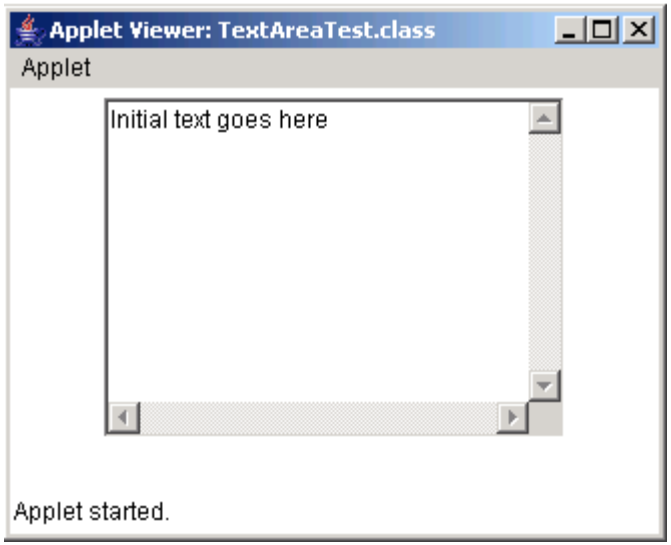
```
import java.applet.Applet;
import java.awt.Choice;

public class ChoiceTest  extends Applet {
    public void init() {
          Choice choice = new Choice();
          choice.add("1975");
          choice.add("1976");
          choice.add("1977");

          add(choice);
    }
}
```



Selected AWT Components (3)

- TextArea

```
import java.applet.Applet;
import java.awt.TextArea;

public class TextAreaTest extends Applet{

    public void init() {
          TextArea area = new TextArea("Initial text goes here", 10, 30);
          add(area);
    }
}
```



Common Component Methods

- getHeight();
- getWidth();
- getX();
- getY();
- getLocationOnScreen();
- setEnabled(boolean);
- setVisible(boolean);
- setBackground(Color);
- setForeground(Color);
- setFont(Font)

AWT Event Handling

- Objects register as listeners for events
- All listeners are always notified
- How it works:
  - Components generate subclasses of `AWTEvent` to notify listensers about state changes
  - Any class can be a listener using `addXXListener()`
  - You have to implement the listener type
  - Some listeners require to implement multiple methods. There are adapters that implement the listener interfaces and stub out all the methods

Low-level Events (basics)

ComponentEvent	Hiding, moving, resizing, showing
ContainerEvent	Adding/removing component
FocusEvent	Getting/losing focus
KeyEvent	Pressing, releasing, or typing (both) a key
MouseEvent	Clicking, dragging, entering, exiting, moving, pressing, or releasing
WindowEvent	Iconifying, deiconifying, opening, closing, really closed, activating, deactivating

AWT Event Hierarchy

Most in java.awt package

```
Object
    EventObject
        AWTEvent
            ActionEvent
            AdjustmentEvent
            ComponentEvent
                ContainerEvent
                FocusEvent
                InputEvent
                    KeyEvent
                    MouseEvent
                        MouseWheelEvent
                PaintEvent
                WindowEvent
            HierarchyEvent
            InputMethodEvent
            InvocationEvent
            ItemEvent
            TextEvent
```

AWT Event Listeners

- Example: MouseListener
  - mouseClicked(MouseEvent e)
  - mouseEntered(MouseEvent e)
  - mouseExited(MouseEvent e)
  - mousePressed(MouseEvent e)
  - mouseReleased(MouseEvent e)

```
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

public class MyMouseAdapter extends MouseAdapter {
        public void mouseClicked(MouseEvent e) {
                System.out.println("Mouse Clicked");
        }
}
```

Listeners – Mouse Example

```
import java.applet.Applet;
import java.awt.*;
import java.awt.event.*;

public class Draw extends Applet {
  public void init() {
    addMouseListener(
        new MouseAdapter() {  // inline
            int savedX, savedY;
            public void mousePressed(MouseEvent e) {
              savedX = e.getX();
              savedY = e.getY();
            }
            public void mouseReleased(MouseEvent e) {
              Graphics g = Draw.this.getGraphics();
              g.drawRect(savedX, savedY,
                         e.getX()-savedX,
                         e.getY()-savedY);
            }
        }
    );
  }
}
```

AWT Layout Manager

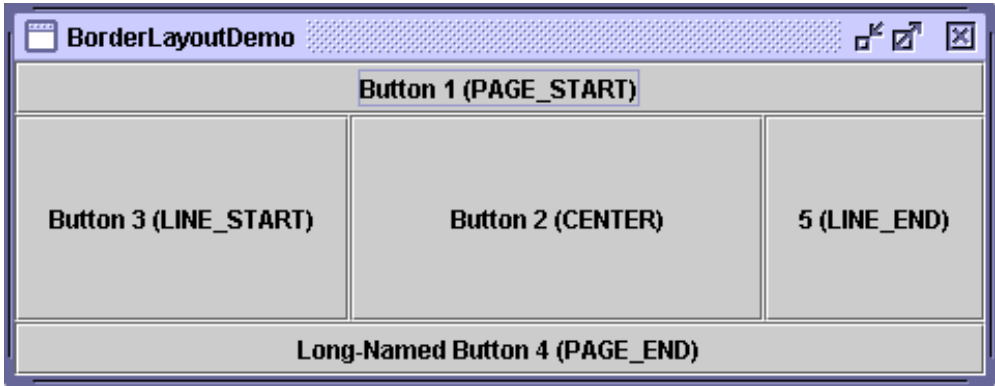
- A Layout Manager is an object that controls the size and position (layout) of components inside a container object

All in java.awt package, “*” indicates interface

```
*LayoutManager
    FlowLayout
    GridLayout
    *LayoutManager2
        BorderLayout
        CardLayout
        GridBagLayout
```

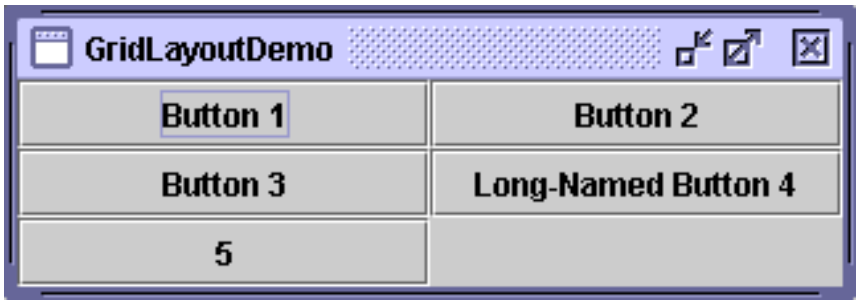
Layouts (1) – BorderLayout

```
JButton button = new JButton("Button 1 (PAGE_START)");
pane.add(button, BorderLayout.PAGE_START);

button = new JButton("Button 2 (CENTER)");
button.setPreferredSize(new Dimension(200, 100));
pane.add(button, BorderLayout.CENTER);

button = new JButton("Button 3 (LINE_START)");
pane.add(button, BorderLayout.LINE_START);

button = new JButton("Long-Named Button 4 (PAGE_END)");
pane.add(button, BorderLayout.PAGE_END);

button = new JButton("5 (LINE_END)");
pane.add(button, BorderLayout.LINE_END);
```



Layouts (2) – GridLayout

```
pane.setLayout(new GridLayout(0,2));


pane.add(new JButton("Button 1"));
pane.add(new JButton("Button 2"));
pane.add(new JButton("Button 3"));
pane.add(new JButton("Long-Named Button 4"));
pane.add(new JButton("5"));
```



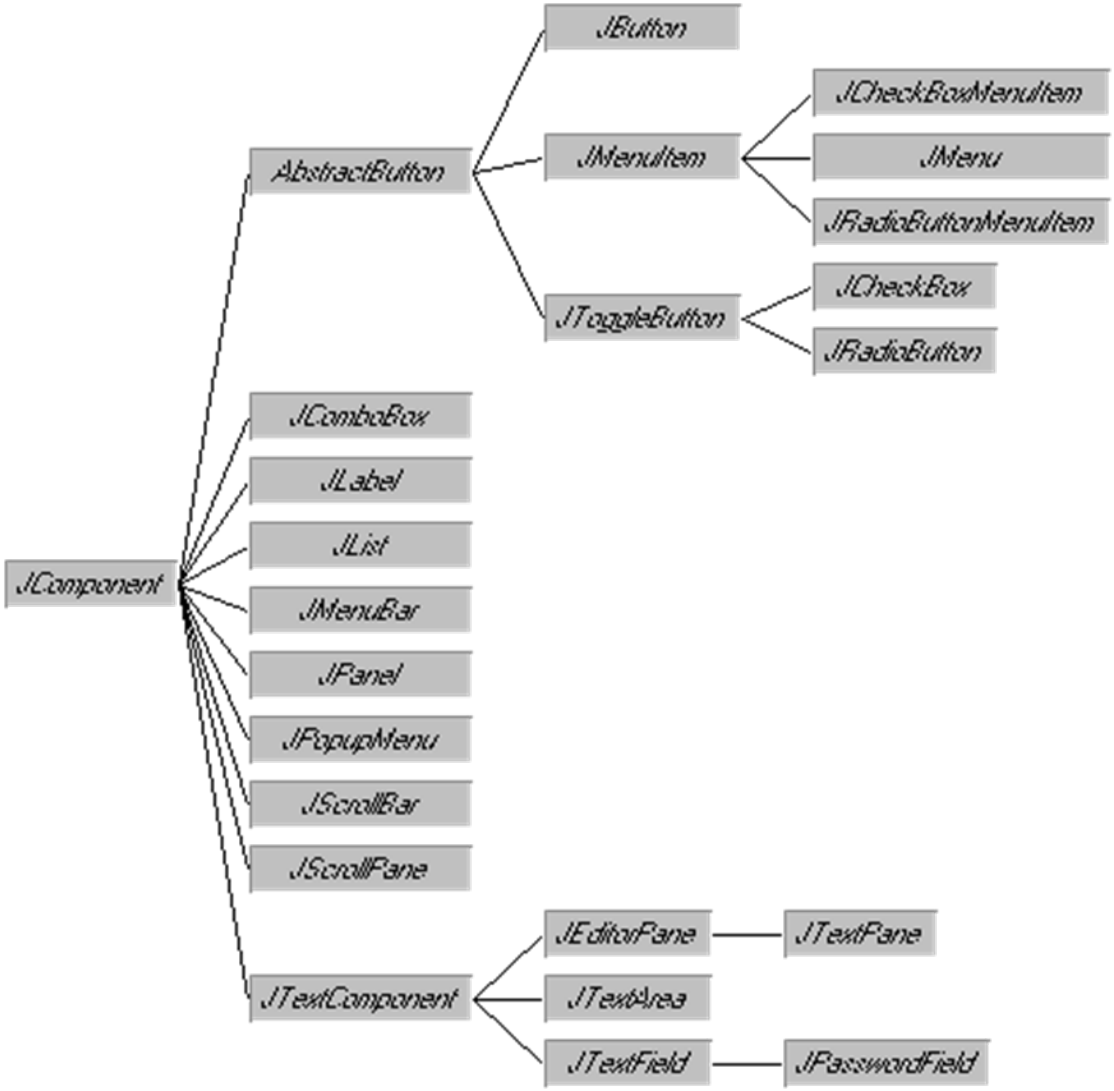
Layouts (3) – FlowLayout

```
contentPane.setLayout(new FlowLayout());

contentPane.add(new JButton("Button 1"));
contentPane.add(new JButton("Button 2"));
contentPane.add(new JButton("Button 3"));
contentPane.add(new JButton("Long-Named Button 4"));
contentPane.add(new JButton("5"));
```

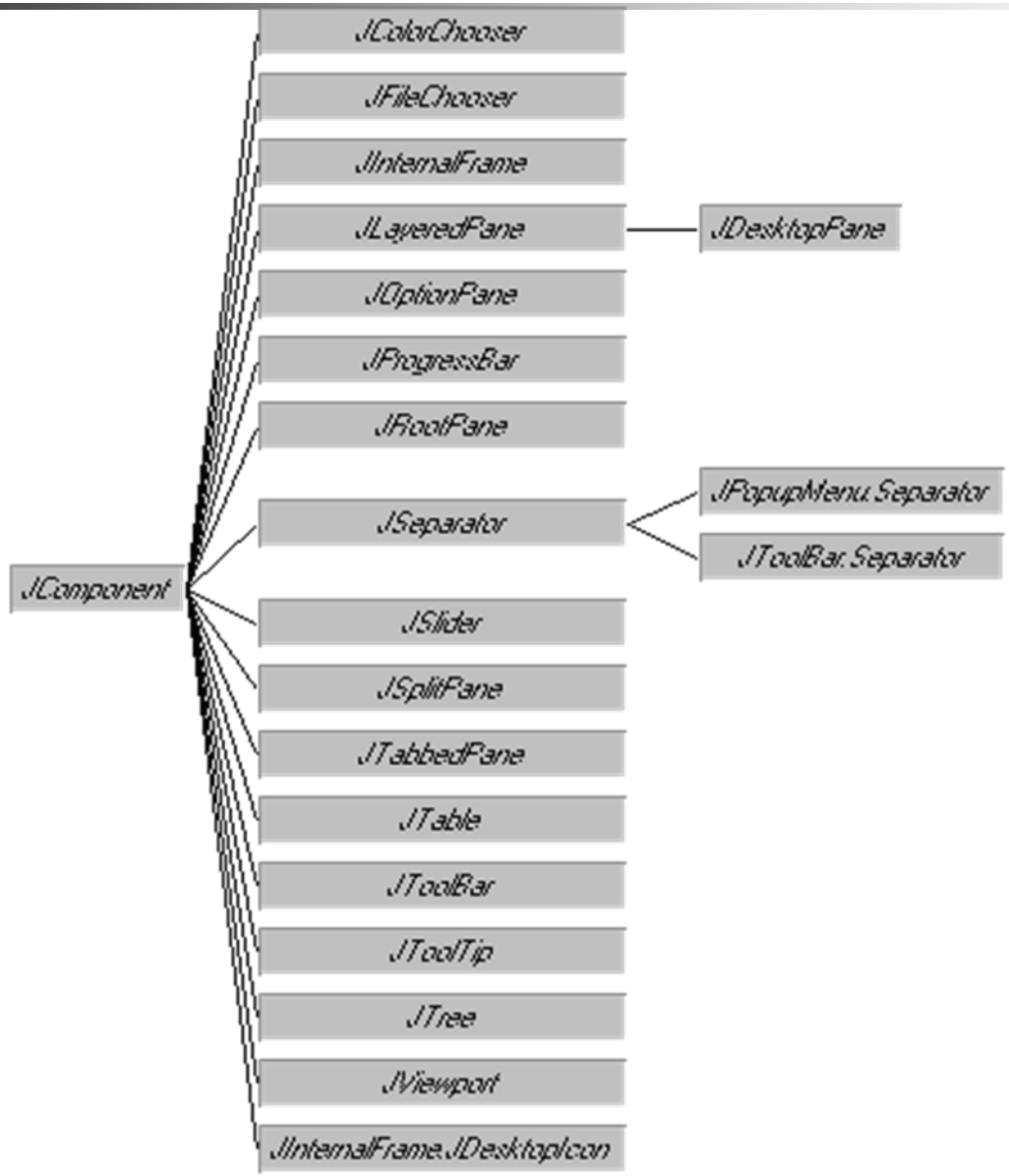
Low-Level: SWING – how it differs

- Provide a more polished look and feel than the standard AWT component set
- A subset of SWING widgets is analogous to the basic AWT widgets (they are lightweight components, rather than peer-based components) - separation of model from the view and controller
- Additional components e.g. trees, tabbed panes
- Programmable look and feel
- Renderers and editors
- typically starting with "J" (e.g. JButton)

SWING – similarities with AWT at the Component level



SWING – differences to AWT



Example: do Label and JLabel really differ?

- Additional functionality of JLabel:
  - Add an Icon
  - Set the vertical and horizontal position of text relative to the Icon
  - Set the relative position of contents within component

Example – JLabel in Action


```
public class LabelPanel extends JPanel {
  public LabelPanel() {
    // Create and add a JLabel
    JLabel plainLabel = new JLabel("Plain Small Label");
    add(plainLabel);
    // Create a 2nd JLabel
    JLabel fancyLabel = new JLabel("Fancy Big Label");
    // Instantiate a Font object to use for the label
    Font fancyFont =
      new Font("Serif", Font.BOLD | Font.ITALIC, 32);
    // Associate the font with the label
    fancyLabel.setFont(fancyFont);
    // Create an Icon
    Icon tigerIcon = new ImageIcon("SmallTiger.gif");
    // Place the Icon in the label
    fancyLabel.setIcon(tigerIcon);
    // Align the text to the right of the Icon
    fancyLabel.setHorizontalAlignment(JLabel.RIGHT);
    // Add to panel
    add(fancyLabel);
  }
}
```

Exercise 1 – Listeners + Layouts

- Write an J2SE application or applet that displays the x,y location of the last mouse click and provides a button to reset the displayed x,y coordinates to 0,0