CS 580 Client-Server Programming
Spring Semester, 2005

Doc 11 Some Java GUI

Reference:

CS 596 Java Programming On-line lecture notes, doc 24, 29, 31

Building GUIs - Abstract Window Toolkit (AWT)

Types of things in AWT

Components

Basic UI things like

Buttons, Checkboxes, Choices,
Lists, Menus, and Text Fields
Scrollbars and Labels

Containers

Contain components and containers

Window

LayoutManagers

Position items in window

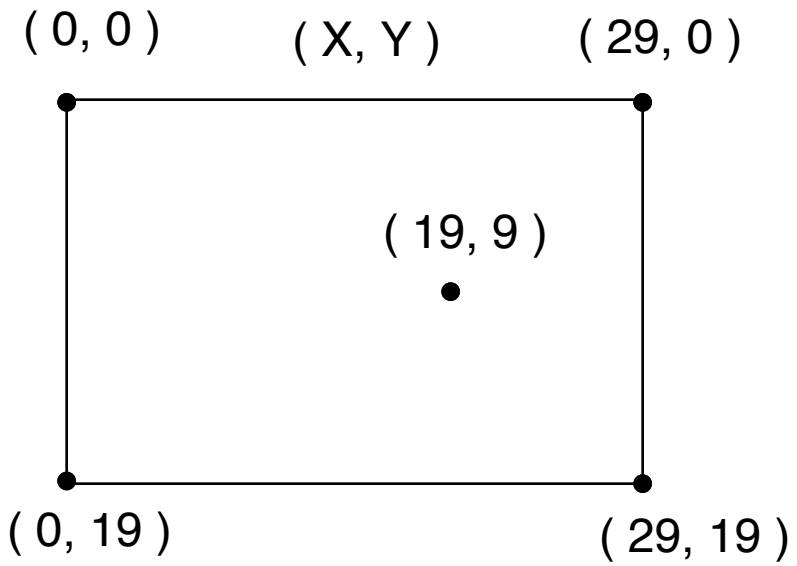
Graphics

Drawing on the screen

Events

Graphics 101

Window Coordinate System



First Example

This example shows a simple application. Subclass Frame to get a w
The paint() method is called when window needs to be drawn. Do no
paint() directly. Becareful! You need to kill the process to quit this pro

```
import java.awt.*;

class HelloApplication extends Frame
   {
   public void paint( Graphics display )
      {
      int startX = 30;
      int startY = 40;
      display.drawString( "Hello World", startX, startY );
      }
   }

class Test
   {
   public static void main( String args[] )
      {
      HelloApplication mainWindow = new HelloApplication();

      int width = 150; // in pixels
      int height = 80;
      int left = 20;
      int top = 40;
      mainWindow.setSize( width, height );
      mainWindow.setLocation( left, top);
      mainWindow.setTitle( "Hello" );
      mainWindow.show();
      }
   }
```

Hello Again

This example uses a Text label instead of drawing text.

```
import  java.awt.*;

class  HelloLabel  extends  Frame
   {
   public HelloLabel( int left, int top, int width, int height, String title )
      {
      super( title );
      setSize( width, height );
      setLocation( left, top);

      Label hello = new Label( "Hello World", Label.CENTER);
      add( hello, BorderLayout.CENTER );

      show();
      }
   }

class Test
   {
   public  static  void  main( String  args[] )
      {
      HelloLabel  mainWindow = new  HelloLabel( 20, 40, 150, 80, "Hi Dad");
      }
   }
```

.

Delegation Event Model

Event Sources

AWT items generate events (mouse down, mouse moved) that c
AWT items must respond to.

Event Listeners

AWT items that must respond to events, need to "listen" the item
generate the events

Events

Event objects are created and sent to AWT listeners. The event
contain information about the "event" that triggered the generatic
event object.

Example - Window Events
Types of Events

windowActivated(WindowEvent)
Invoked when a window is activated.

windowClosed(WindowEvent)
Invoked when a window has been closed.

windowClosing(WindowEvent)
Invoked when a window is in the process of being closed.

windowDeactivated(WindowEvent)
Invoked when a window is de-activated.

windowDeiconified(WindowEvent)
Invoked when a window is de-iconified.

windowIconified(WindowEvent)
Invoked when a window is iconified.

windowOpened(WindowEvent)
Invoked when a window has been opened.

WindowEvent
Methods

getWindow()
Returns the window where this event originated.
getComponent()
Returns the component where this event originated.
paramString()
consume()
getID()
Returns the event type.
getSource()
isConsumed()

Public Final Fields

WINDOW_ACTIVATED
The window activated event type.
WINDOW_CLOSED
The window closed event type.
WINDOW_CLOSING
The window closing event type.
WINDOW_DEACTIVATED
The window deactivated event type.
WINDOW_DEICONIFIED
The window deiconified event type.
WINDOW_FIRST
Marks the first integer id for the range of window event ids.
WINDOW_ICONIFIED
The window iconified event type.
WINDOW_LAST
Marks the last integer id for the range of window event ids.
WINDOW_OPENED
The window opened event type.

Window Event Example

This example shows how to use a WindowListener to quit an applica
Note that you cannot use System.exit(0) in an applet.

```
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.Frame;
import java.awt.Label;

class QuitableApplication1 extends  Frame
   {
   public QuitableApplication1( int left, int top, int width, int height, String title )
      {
      super( title );
      setSize( width, height );
      setLocation( left, top);

      Label hello = new Label( "Hello World", Label.CENTER);
      add( hello, BorderLayout.CENTER );

      addWindowListener( new QuitWindow( this ) );
      show();
      }
   }

class QuitWindow implements WindowListener
   {
   Frame myWindow;
   public QuitWindow( Frame aWindow )
      {
      myWindow = aWindow;
      }

   public void windowClosing(WindowEvent event)
      {
      myWindow.dispose();
      System.exit(0);
      }

   public void windowActivated(WindowEvent event) {};
   public void windowClosed(WindowEvent event) {};
```

WindowAdapter

The WindowAdapter class implements, with null methods, all the me
the WindowListener interface. By subclassing the WindowAdapter cl
can just implement the WindowListener methods you need.

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.Frame;
import java.awt.Label;

class QuitableApplication2 extends  Frame
    {
    public QuitableApplication2( int left, int top, int width, int height, String title )
        {
        super( title );
        setSize( width, height );
        setLocation( left, top);

        Label hello = new Label( "Hello World", Label.CENTER);
        add( hello, BorderLayout.CENTER );

        addWindowListener( new QuitWindowAdapter( this ) );
        show();
        }
    }

class QuitWindowAdapter extends WindowAdapter
    {
    Frame myWindow;

    public QuitWindowAdapter( Frame aWindow )
        {
        myWindow = aWindow;
        }

    public void windowClosing( WindowEvent event )
        {
        myWindow.dispose();
```

Using Anonymous Classes

Using an anonymous class shortens the example. This use of anony classes is the main reason for adding them to the language.

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.Frame;
import java.awt.Label;

class QuitableApplication3 extends  Frame
    {
    public QuitableApplication3( int left, int top, int width, int height, String title )
        {
        super( title );
        setSize( width, height );
        setLocation( left, top);

        Label hello = new Label( "Hello World", Label.CENTER);
        add( hello, BorderLayout.CENTER );

        addWindowListener( new WindowAdapter( ) {
                public void windowClosing( WindowEvent event )
                    {
                    dispose();
                    System.exit(0);
                    }
                }
            );
        show();
        }
    }
```

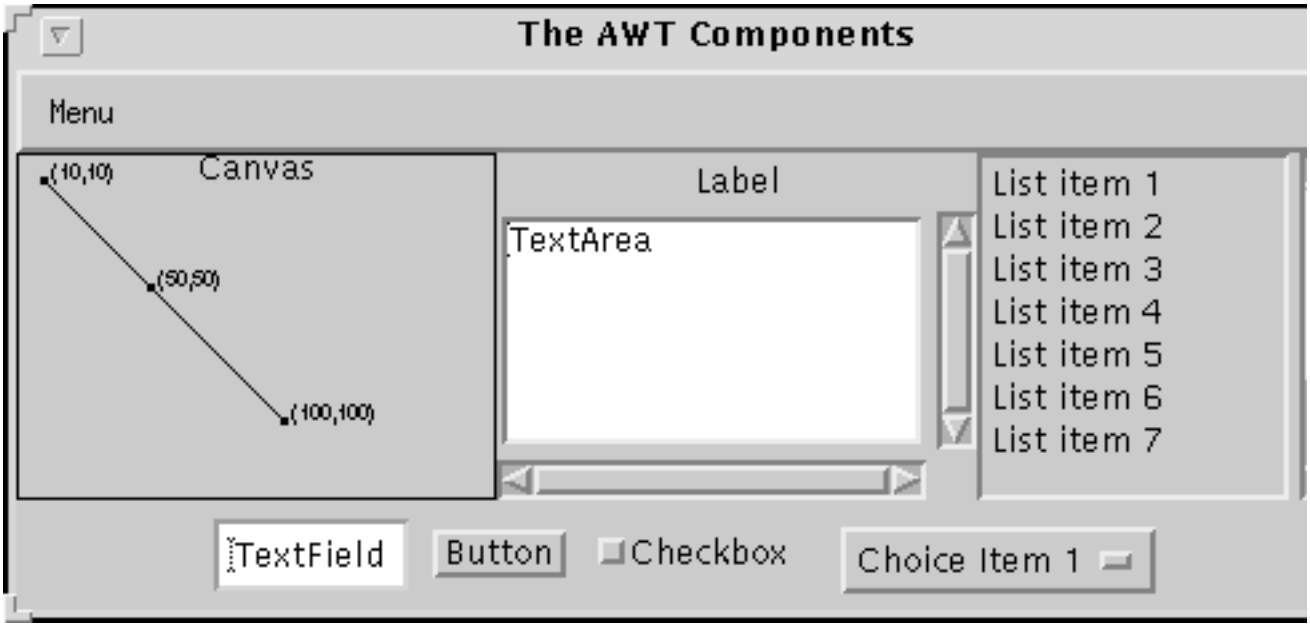
Listening to Yourself

This example shows how use an adapter to have a class listen to itse
allows the class to handle its own events.

```
import java.awt.event.WindowAdapter;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.awt.Frame;
import java.awt.Label;

class QuitableApplication4 extends  Frame
   {
   public QuitableApplication4( int left, int top, int width, int height, String title )
      {
      super( title );
      setSize( width, height );
      setLocation( left, top);

      Label hello = new Label( "Hello World", Label.CENTER);
      add( hello, BorderLayout.CENTER );

      addWindowListener( new WindowAdapter( ) {
            public void windowClosing( WindowEvent event )
               {
               quit();
               }
            }
         );
      show();
      }

   public void quit()
      {
      dispose();
      System.exit( 0 );
      }
   }
```

Overview of AWT Parts
Components

Components are basic UI things



Buttons, Checkboxes, Choices, Lists, Menus, and Text Fields

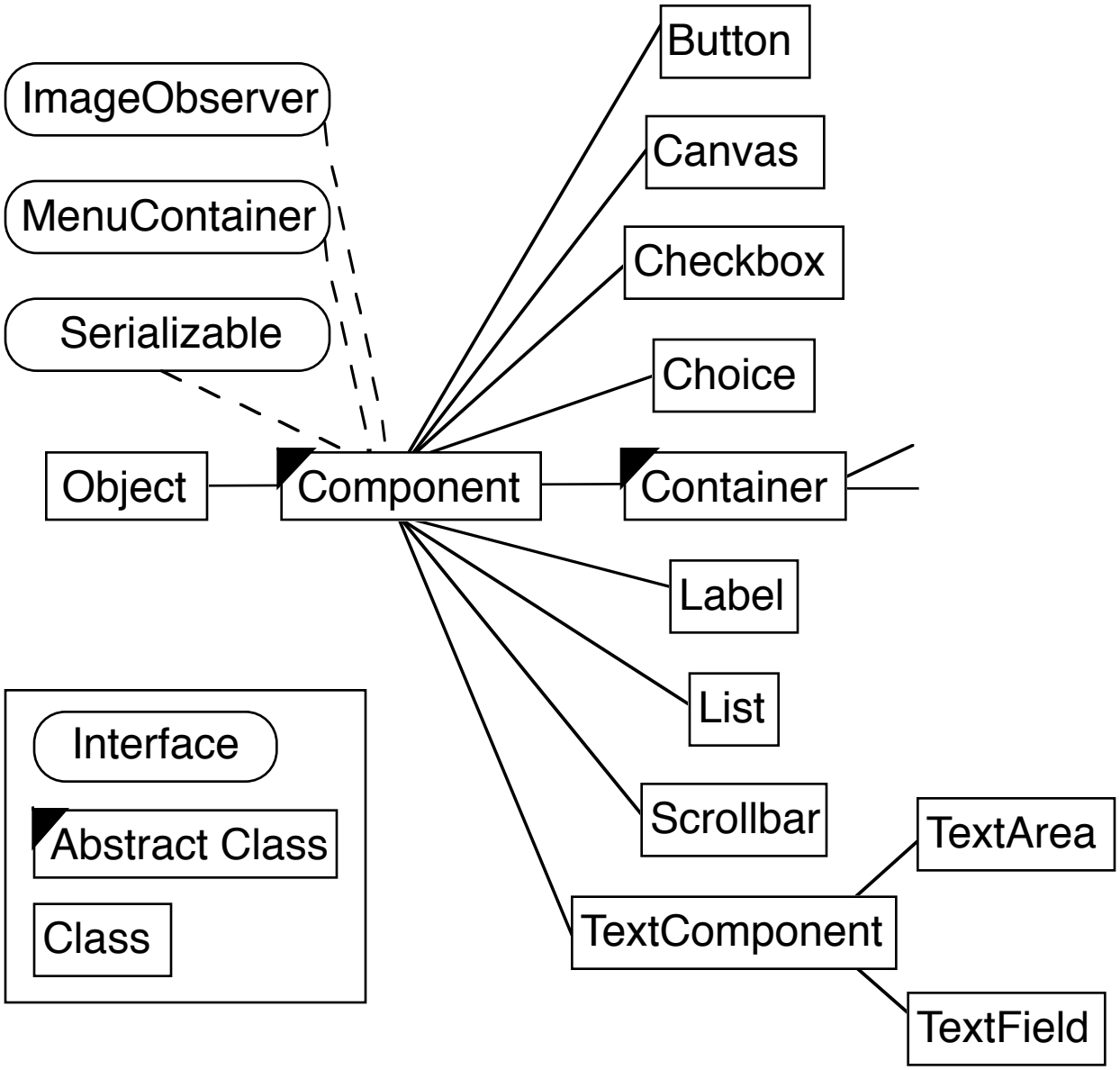
When a user activates one of these controls -- by clicking a butto
example -- it posts generates an event.

Canvases and Text Areas

Ways of Getting User Input

Scrollbars and Labels

Component Classes



Containers: Windows and Panels

Containers contain components and containers

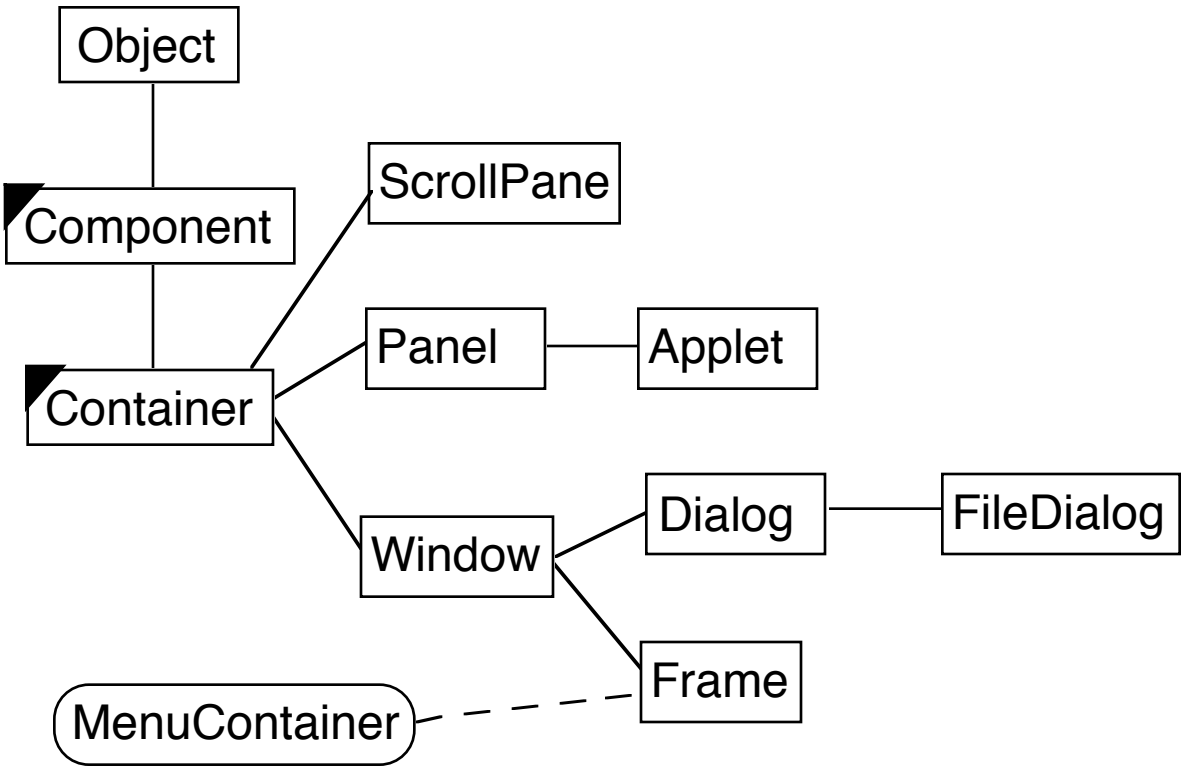
Window

- Dialog, FileDialog, and Frame are subclasses of Window
- Provide windows to contain components

Panels

- Group components within an area of an existing window

Container Classes



Component

Background Color	void setBackground(Color) Color getBackground()
Bounds	void setBounds(Rectangle) void setBounds(int, int, int, int) Rectangle getBounds()
Cursor	void setCursor(Cursor) Cursor getCursor()
Drop Target	void setDropTarget(DropTarget) DropTarget getDropTarget()
Enabled	void setEnabled(boolean) boolean getEnabled()
Font	void setFont(Font) Font getFont()
Foreground Color	void setForeground(Color) Color getForeground()
Locale	void setLocale(Locale) Locale getLocale()
Location	void setLocation(Point) void setLocation(int, int) Point getLocation() Point getLocationOnScreen()
Name	void setName(String) String getName()
Size	void setSize(Dimension) Dimension getSize()
Visible	void setVisible(boolean) boolean getVisible()

Deprecated Methods

The get/set convention was not strictly followed in JDK 1.0. In JDK 1

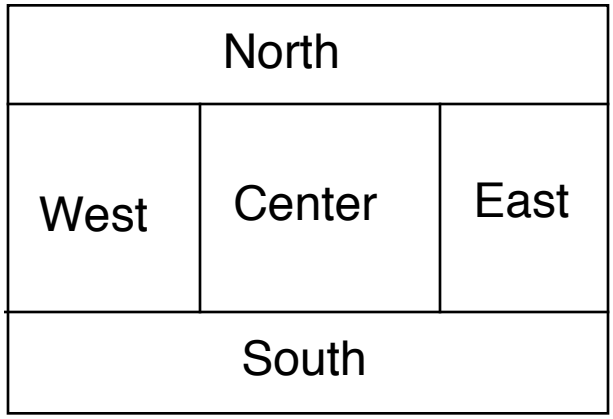
Layouts

We need to specify where items will appear in a window. This has to
on window that changes size and on different platforms: PC, Mac, U
toasters, and TVs. Layouts provide a flexible way to place items in a
without specifying coordinates.

AWT Layouts
- BorderLayout
- CardLayout
- FlowLayout
- GridBagLayout
- GridLayout

BorderLayout

Default layout for a frame
Divides area into named regions:



FlowLayout

Displays components left to right, top to bottom

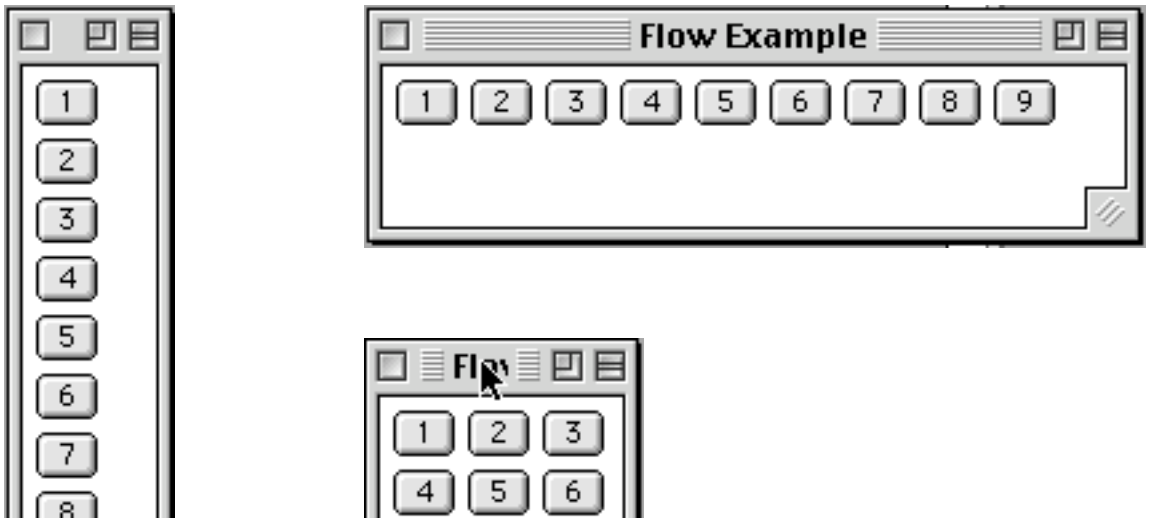
Button Examples

We will use buttons to illustrate some basics of layouts, components,
events.

FlowLayout

This example creates a FlowLayout that aligns its components to the
"new FlowLayout()" will create a FlowLayout that centers its compon
Note how the buttons in the window "flow" as the window's shape is

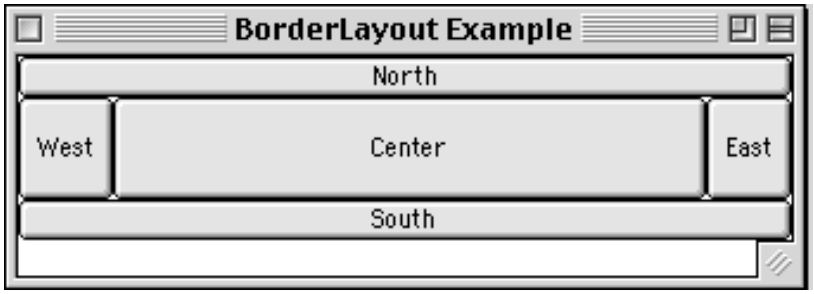
```
import java.awt.*;

class  FlowLayoutExample  extends Frame  {

    public FlowLayoutExample( int  width, int height ) {
        setTitle( "Flow Example" );
        setSize( width, height );
        setLayout( new FlowLayout( FlowLayout.LEFT) );

        for ( int label = 1; label < 10; label++ )
            add( new Button( String.valueOf( label ) ) );
        show();
    }

    public  static  void  main( String  args[] ) {
        new  FlowLayoutExample( 175, 100 );
    }
}
```

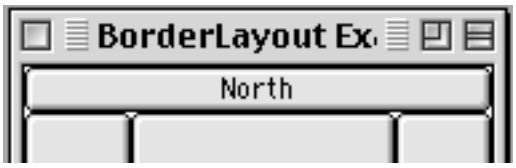


BorderLayout

Since the BorderLayout is the default, we do not have to use the setl method to specify the BorderLayout. It was done here just to insure t readers know that we are using the BorderLayout. Note how the layc the buttons as the window is resized.

```
import java.awt.*;
class  BorderLayoutExample  extends Frame  {

   public BorderLayoutExample( int  widthInPixels,
                               int heightInPixels ) {
      setTitle( "BorderLayout Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new BorderLayout() );

      add( new Button( "North" ), BorderLayout.NORTH );
      add( new Button( "Center" ), BorderLayout.CENTER );
      add( new Button( "East" ), BorderLayout.EAST );
      add( new Button( "West" ), BorderLayout.WEST );
      add(  new Button( "South" ), BorderLayout.SOUTH );

      show();
   }

   public  static  void  main( String  args[] ) {
      new  BorderLayoutExample( 175, 100 );
   }
}
```





Button Events

Buttons support ActionListeners. When the mouse is pressed and re on a button, all listeners are send the actionPerformed() method. In t example we only need to know when the button was selected, so we use the ActionEvent.

```
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class  ButtonExample  extends Frame {
   Button red = new Button( "Red" );
   Button blue = new Button( "Blue" );

   public ButtonExample( int  widthInPixels, int heightInPixels ) {
      setTitle( "Button Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );
      add( red );
      add( blue );

      red.addActionListener( new ColorActionListener( Color.red) );
      blue.addActionListener( new ColorActionListener( Color.blue) );

      show();
   }

   public  static  void  main( String  args[] ){
      ButtonExample  window = new  ButtonExample(200, 50);
      }

   class ColorActionListener implements ActionListener {
      Color backgroundColor;

      public ColorActionListener( Color aColor ) {
         backgroundColor = aColor;
      }

      public void actionPerformed( ActionEvent event )  {
         setBackground( backgroundColor );
         repaint();     // Show effect of color change
      }
   }
```

Button ActionEvent

This example, which takes two slides, show how to use the ActionEv

```
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class  ActionEventDataExample  extends Frame
   {
   Button left = new Button( "Left" );
   Button right = new Button( "Right" );

   public ActionEventDataExample( int  widthInPixels, int heightInPixels )
      {
      setTitle( "Button Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );
      add( left );
      add( right );

      left.addActionListener( new ActionData( ) );
      right.addActionListener( new ActionData( ) );

      show();
      }

   public  static  void  main( String  args[] ){
      ActionEventDataExample  window =
         new  ActionEventDataExample(200, 50);
      }
```

Button ActionEvent Continued

```
class ActionData implements ActionListener
   {
   public void actionPerformed( ActionEvent event )
      {
      String command = event.getActionCommand();
      int modifiers = event.getModifiers();
      String parameters = event.paramString();
      System.out.println( "Command " + command);
      System.out.println( "Parameters " + parameters);

      switch ( modifiers )
         {
         case ActionEvent.ALT_MASK:
            System.out.println( "Alt Key was pressed");
            break;
         case ActionEvent.CTRL_MASK:
            System.out.println( "Control Key was pressed");
            break;
         case ActionEvent.META_MASK:
            System.out.println( "Meta Key was pressed");
            break;
         case ActionEvent.SHIFT_MASK:
            System.out.println( "Shift Key was pressed");
            break;
         default:
            System.out.println( "No modifier keys were pressed");
         }
      }
   }
}
```

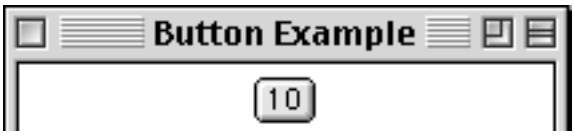
Output

Custom Buttons

We can create subclasses of Button, which will add functionality to th Button class.

```
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

class CountButton extends Button implements ActionListener {
   int count = 0;

   public CountButton() {
      super( "0" );
      addActionListener( this );
   }

   public void actionPerformed( ActionEvent event )  {
      if (event.getSource() == this ) {
         count++;
         setLabel( String.valueOf( count ) );
      }
   }
}

class  SpecialButtonExample  extends Frame {
   public SpecialButtonExample( int  widthInPixels, int heightInPixels ) {
      setTitle( "Button Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );
      add( new CountButton() );

      show();
   }

   public  static  void  main( String  args[] ){
      SpecialButtonExample  window = new  SpecialButtonExample(200, 50);
   }
}
```

Output (after a number of mouse clicks)

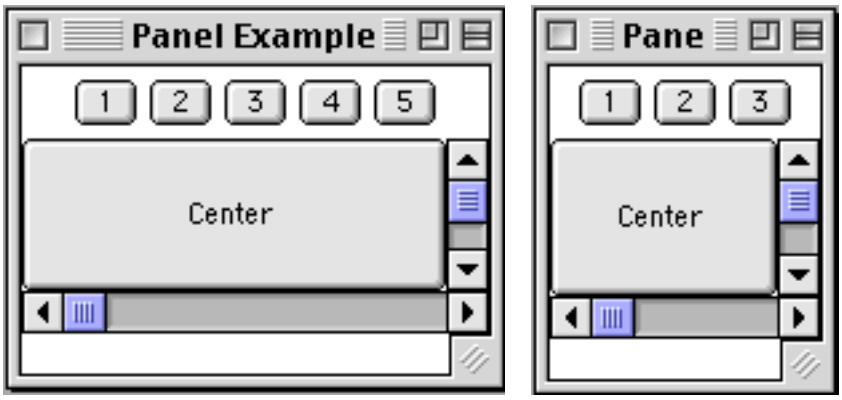


Containers
Panels

Panels allow use of one layout inside of another layout. A panel is a container, so it holds components. A panel has a layout. The default for all panels is FlowLayout. Mixing layouts increases the flexibility of In the example below, a panel full of buttons is added to the north reg Borderlayout.

```
import java.awt.*;

class  PanelExample  extends Frame  {

   public PanelExample( int  widthInPixels, int h  eightInPixels ) {
      setTitle( "Panel Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new BorderLayout() );

      Panel buttonDisplay = new Panel();
      buttonDisplay.setLayout( new FlowLayout() );

      for ( int label = 1; label < 6; label++ )
         buttonDisplay.add( new Button( String.valueOf( label )) );

      add( buttonDisplay, BorderLayout.NORTH );
      add( new Button( "Center" ) , BorderLayout.CENTER );
      add( new Scrollbar( Scrollbar.VERTICAL ), BorderLayout.EAST );
      add( new Scrollbar( Scrollbar.HORIZONTAL ) , BorderLayout.SOUTH );

      show();
   }

   public  static  void  main( String  args[] ) {
      new  PanelExample( 175, 100 );
   }
}
```

Result of Panel Example



Note that the buttons do not wrap to second row of buttons event tho
are in a FlowLayout! The FlowLayout is in north part of a BorderLayo
one row is available!

Label

Label provides a way to display text on the screen. It does not suppo
events other that those supported by the Component class.

Label Constructors

Label()
Constructs an empty label.

Label(String)
Constructs a new label with the specified String of text.

Label(String, int)
Constructs a new label with the specified String of text and the s
alignment.

Label Constants

CENTER LEFT RIGHT

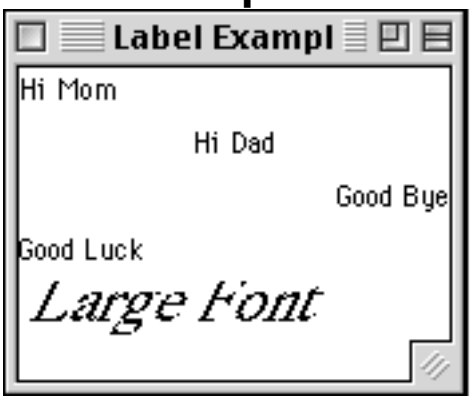
Some Label Methods

addNotify()	paramString()
getAlignment()	setAlignment(int)
getText()	setText(String)

Label Example

```
class  LabelExample  extends Frame  {

   public LabelExample( int  widthInPixels, int heightInPixels ) {
      setTitle( "Label Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new GridLayout(5, 1) );

      add( new Label( "Hi Mom" ) );
      add( new Label( "Hi Dad", Label.CENTER  ) );
      add( new Label( "Good Bye", Label.RIGHT ) );
      add( new Label( "Good Luck", Label.LEFT ) );

      Font largeFont = new Font( "TimesRoman", Font.ITALIC, 24 );
      Label largeText = new Label( "Large Font" );
      largeText.setFont( largeFont );
      add( largeText );
      show();
   }

   public  static  void  main( String  args[] ) {
      new  LabelExample(150, 100);
   }
}
```

Output



Button

Button supports the ActionEvent.

Component	Events	Meaning
Button	ActionEvent	Button clicked

Event Class	Listener Interface	Listener Methods
ActionEvent	ActionListener	actionPerformed()

We have seen buttons before. Here we set the font of a button.

```
class  ButtonTextExample  extends Frame  {

   public ButtonTextExample( int widthInPixels, int heightInPixels) {
      setTitle( "Button Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );

      Font largeFont = new Font( "TimesRoman", Font.ITALIC, 24 );
      Button largeText = new Button( "Large Font" );
      largeText.setFont( largeFont );
      add( largeText );
      show();
   }

   public  static  void  main( String  args[] ) {
      new  ButtonTextExample(150, 100);
   }
}
```

Choice - Drop-down Lists

Component	Events	Meaning
Choice	ItemEvent	Item selected or deselected

Event Class	Listener Interface	Listener Methods
ItemEvent	ItemListener	itemStateChanged()

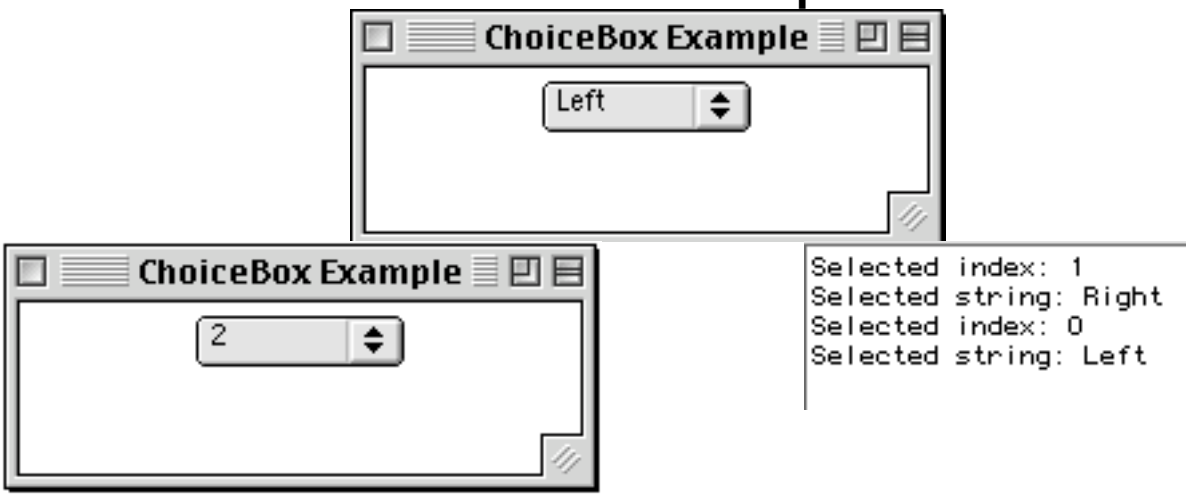
Some Choice Methods

add(String)
addItem(String)
addItemListener(ItemListener)
removeItemListener(ItemListener)
getItem(int)
getItemCount()
getSelectedIndex()
getSelectedItem()
getSelectedObjects()
insert(String, int)
paramString()
remove(int)
remove(String)
removeAll()
select(int)
select(String)

Choice Example

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

class  ChoiceBoxes  extends Frame  {

   Choice directions = new Choice();

   public ChoiceBoxes( int  widthInPixels, int heightInPixels ) {
      setTitle( "ChoiceBox Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );

      directions.addItem( "Left" );
      directions.addItem( "Right" );
      directions.addItem( "Up" );
      directions.addItem( "Down" );
      directions.addItemListener( new ChoiceListener() );
      add( directions );
      show();
   }

   class ChoiceListener implements ItemListener {
      int  count = 1;
      public void itemStateChanged( ItemEvent event ) {
         Choice selected = (Choice) event.getItemSelectable();
         int selectedIndex = selected.getSelectedIndex();
         String selectedText = selected.getSelectedItem();
         System.out.println( "Selected index: " +  selectedIndex);
         System.out.println( "Selected string: " + selectedText );
         selected.insert( String.valueOf( count++ ), selectedIndex );
         selected.remove( selectedText );
      }
   }
   public  static  void  main( String  args[] ) { new  ChoiceBoxes(200, 200); }
}
```

Choice Example Continued
Output



List

Component	Events	Meaning
List	ActionEvent	List item doubled clicked
	ItemEvent	List item selected or deselected

Event Class	Listener Interface	Listener Methods
ActionEvent	ActionListener	actionPerformed()
ItemEvent	ItemListener	itemStateChanged()

List Constructors

List()

Creates a new scrolling list initialized with no visible Lines or mu selections.

List(int, boolean)

Creates a new scrolling list initialized with the specified number lines and a boolean stating whether multiple selections are allow not.

List Methods

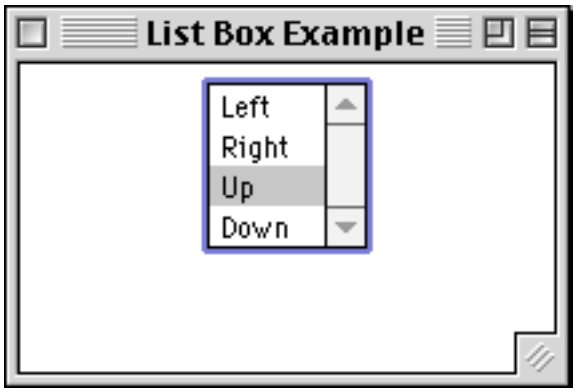
add(String)
add(String, int)
addActionListener(ActionListener)
addItem(String)
addItem(String, int)
addItemListener(ItemListener)
addNotify()
delItem(int)
deselect(int)
getItem(int)
getItemCount()
getItems()
getMinimumSize()
getMinimumSize(int)
getPreferredSize()
getPreferredSize(int)
getRows()
getSelectedIndex()
getSelectedIndexes()
getSelectedItem()
getSelectedItems()
getSelectedObjects()
getVisibleIndex()
isIndexSelected(int)
isMultipleMode()
makeVisible(int)
paramString()
remove(int)
remove(String)
removeActionListener(ActionListener)
removeAll()
removeItemListener(ItemListener)
removeNotify()
replaceItem(String, int)
select(int)
setMultipleMode(boolean)

List Example

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

class  ListExample  extends Frame  {

   public ListExample( int  widthInPixels, int heightInPixels ) {
      setTitle( "List Box Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );

      boolean areMultipleSelectionsAllowed = false;
      int numberDisplayed = 4;
      List directions = new List( numberDisplayed,
                        areMultipleSelectionsAllowed );

      directions.addItem( "Left" );
      directions.addItem( "Right" );
      directions.addItem( "Up" );
      directions.addItem( "Down" );
      directions.addItemListener( new ListListener() );
      add( directions );
      show();
   }
```

List Example

```
    class ListListener implements ItemListener {
        public void itemStateChanged( ItemEvent event ) {
            List selected = (List) event.getItemSelectable();
            int selectedIndex = selected.getSelectedIndex();
            String selectedText = selected.getSelectedItem();
            System.out.println( "Selected index: " +  selectedIndex);
            System.out.println( "Selected string: " + selectedText );
        }
    }

    public  static  void  main( String  args[] ) {
        new  ListExample(200, 100);
    }
}
```



Output

```
Selected index: 1
Selected string: Right
Selected index: 2
Selected string: Up
Selected index: 2
Selected string: Up
```

TextArea

A TextArea is a scrollable region that users can enter text. A TextAre contain multiple rows and columns of text. The number of characters fit on one row depends on the font, font size and number of columns.

Component	Events	Meaning
TextComponent	TextEvent	User changed text

Event Class	Listener Interface	Listener Methods
TextEvent	TextListener	textValueChanged()

TextArea Constructors

TextArea()
Constructs a new TextArea.

TextArea(int, int)
Constructs a new TextArea with the specified number of rows ar columns.

TextArea(String)
Constructs a new TextArea with the specified text displayed.

TextArea(String, int, int)
Constructs a new TextArea with the specified text and number c and columns.

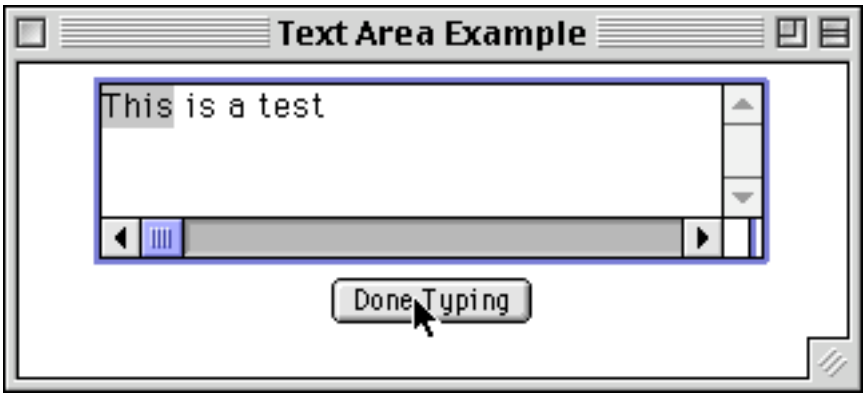
TextArea Methods

addNotify()
append(String)
getColumns()
getMinimumSize()
getMinimumSize(int, int)
getPreferredSize()
getPreferredSize(int, int)
getRows()
getScrollbarVisibility()
insert(String, int)
paramString()
replaceRange(String, int, int)
setColumns(int)
setRows(int)

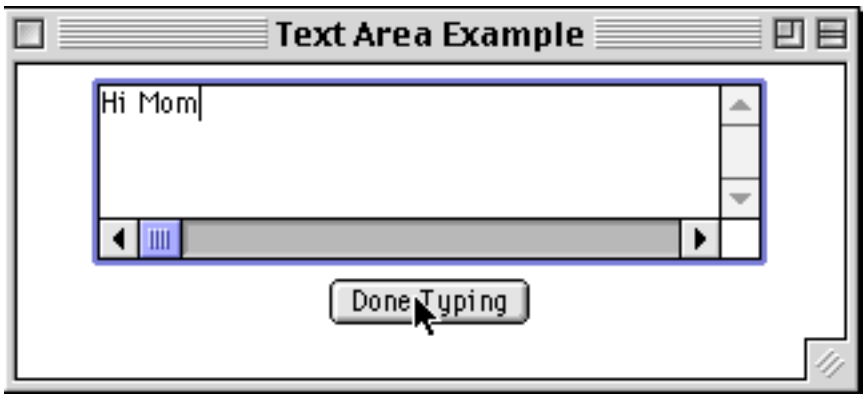
TextArea Example

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

class  TextAreas  extends Frame  {

   TextArea userInput;
   Button done = new Button( "Done Typing" );

   public TextAreas( int  widthInPixels, int heightInPixels ) {
      setTitle( "Text Area Example" );
      setSize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );

      int numberOfLines = 3;
      int numberOfColumns = 30;
      userInput = new TextArea( numberOfLines, numberOfColumns );
      add( userInput );
      add( done );

      done.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
               checkUserInput();
            }
         });
      show();
   }

   private void checkUserInput() {
      System.out.println( "You typed: " +  userInput.getText() );
      System.out.println( "Selected text>" +
                        userInput.getSelectedText() +"<" );
      userInput.setText( "Hi Mom");
   }
   public  static  void  main( String  args[] ) {
      new  TextAreas(300, 150);
   }
}
```

Output




```
You typed: This is a test
Selected text>This<
```



TextField

A TextField is a region that users can enter text. A TextField contains one row of text. The number of characters that can be seen in a Text depends on the font, font size and number of columns.

Component	Events	Meaning
TextComponent	TextEvent	User changed text
TextField	ActionEvent	User finished editing text

Event Class	Listener Interface	Listener Methods
ActionEvent	ActionListener	actionPerformed()
TextEvent	TextListener	textValueChanged()

TextField Constructors

TextField()
: Constructs a new TextField.

TextField(int)
: Constructs a new TextField initialized with the specified columns

TextField(String)
: Constructs a new TextField initialized with the specified text.

TextField(String, int)
: Constructs a new TextField initialized with the specified text and columns.

Some TextField Methods

addActionListener(ActionListener)
removeActionListener(ActionListener)
addNotify()
echoCharIsSet()
getColumns()
getEchoChar()
getMinimumSize()
getMinimumSize(int)
getPreferredSize()
getPreferredSize(int)
paramString()
setColumns(int)
setEchoChar(char)

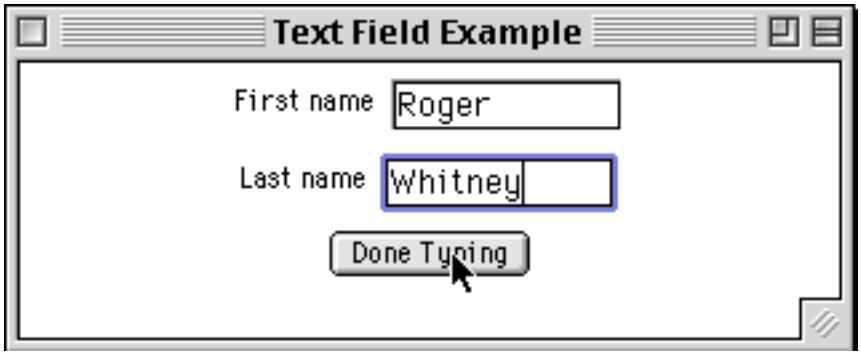
TextField Example

```
class  TextFieldExample  extends Frame  {

    TextField firstName;
    TextField lastName;
    Button done = new Button( "Done Typing" );

    public TextFieldExample( int  widthInPixels, int heightInPixels ) {
        setTitle( "Text Field Example" );
        setSize( widthInPixels, heightInPixels );
        setLayout( new GridLayout(3, 1) );

        int numberOfColumns = 10;
        firstName = new TextField(  numberOfColumns );
        Panel first = new Panel( new FlowLayout() );
        first.add( new Label( "First name", Label.RIGHT ));
        first.add( firstName );

        lastName = new TextField(  numberOfColumns );
        Panel last = new Panel( new FlowLayout() );
        last.add( new Label( "Last name", Label.RIGHT ) );
        last.add( lastName );

        Panel buttons =
            new Panel( new FlowLayout( FlowLayout.CENTER));
        buttons.add( done );

        add( first );
        add( last );
        add( buttons );
```

TextField Example Continued

```
        // Called when user hits the enter key
        firstName.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                System.out.println( "Action Event " +
                    event.getActionCommand() );
            }
        });

        // Called when button is pressed.
        done.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
                System.out.println("Your name is: " +
                        firstName.getText() +" " + lastName.getText() );
            }
        });
        show();
    }

    public static void main( String args[] ) {
        new TextFieldExample(300, 150);
    }
}
```

```
Your name is: Roger Whitney
```



Dialogs

Dialogs can be either modal or nonmodal.

Modal

A modal dialog blocks:

- Thread execution
- Input to other windows

Dialog Constructors

Dialog(Frame parent)
Constructs an initially invisible Dialog with an empty title.

Dialog(Frame parent, boolean modal)
Constructs an initially invisible Dialog with an empty title. If moda
then block input from the parent frame when dialog is shown.

Dialog(Frame parent, String title)
Constructs an initially invisible Dialog with a title.

Dialog(Frame parent, String title, boolean modal)
Constructs an initially invisible Dialog with a title. If modal is true
block input from the parent frame when dialog is shown.

Some Dialog Methods

getTitle()	paramString()	setTitle(String)
isModal()	setModal(boolean)	show()
isResizable()	setResizable(boolean)	

Dialog Example

On this slide, a LoginDialog class is defined. On the next slide, it is u

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

class  LoginDialog  extends Dialog  {

   TextField password;
   Button done = new Button( "OK" );

   public LoginDialog( Frame owner ) {
      super( owner, "Log in", true );
      setSize( 200, 100 );
      setLocation( 50, 50 );
      setLayout( new GridLayout(2, 1) );

      int numberOfColumns = 10;
      password = new TextField( numberOfColumns );
      password.setEchoChar( '*' );
      password = new TextField(  numberOfColumns );
      Panel first = new Panel( new FlowLayout() );
      first.add( new Label( "Password", Label.RIGHT ));
      first.add( password );

      Panel buttons =
         new Panel( new FlowLayout( FlowLayout.CENTER));
      buttons.add( done );

      add( first );     add( buttons );

      done.addActionListener( new ActionListener() {
            public void actionPerformed( ActionEvent event ) {
               setVisible(false);
            }
         });
   }

   public String getPassword() {  return password.getText(); }
}
```

Dialog Example Continued

```
class  DialogExample  extends Frame  {

   public DialogExample( int  widthInPixels, int heightInPixels ) {
      setTitle( "Text Validation Example" );
      resize( widthInPixels, heightInPixels );
      setLayout( new FlowLayout() );

      Button login = new Button( "Log in" );
      login.addActionListener( new Login() );
      add( login );
      show();
   }

   public  static  void  main( String  args[] ) {
      new  DialogExample(100, 50);
   }

   class Login implements ActionListener {
      public  void actionPerformed( ActionEvent event ) {
         LoginDialog passwordDialog =
               new LoginDialog( DialogExample.this );

         passwordDialog.show(  );
         System.out.println( "Password is: " +
            passwordDialog.getPassword() );
      }
   }
}
```

Dialog Example Continued