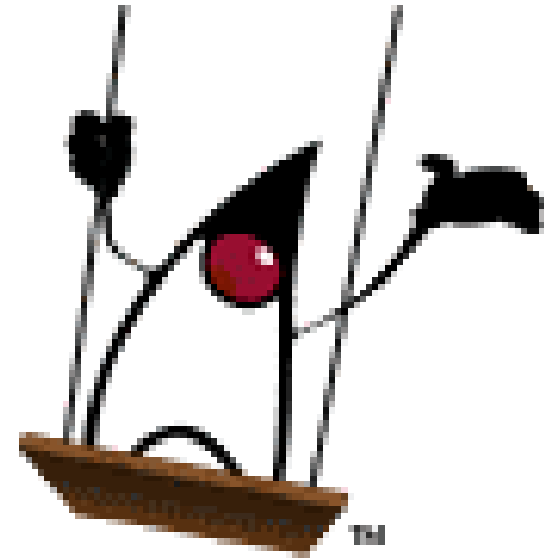
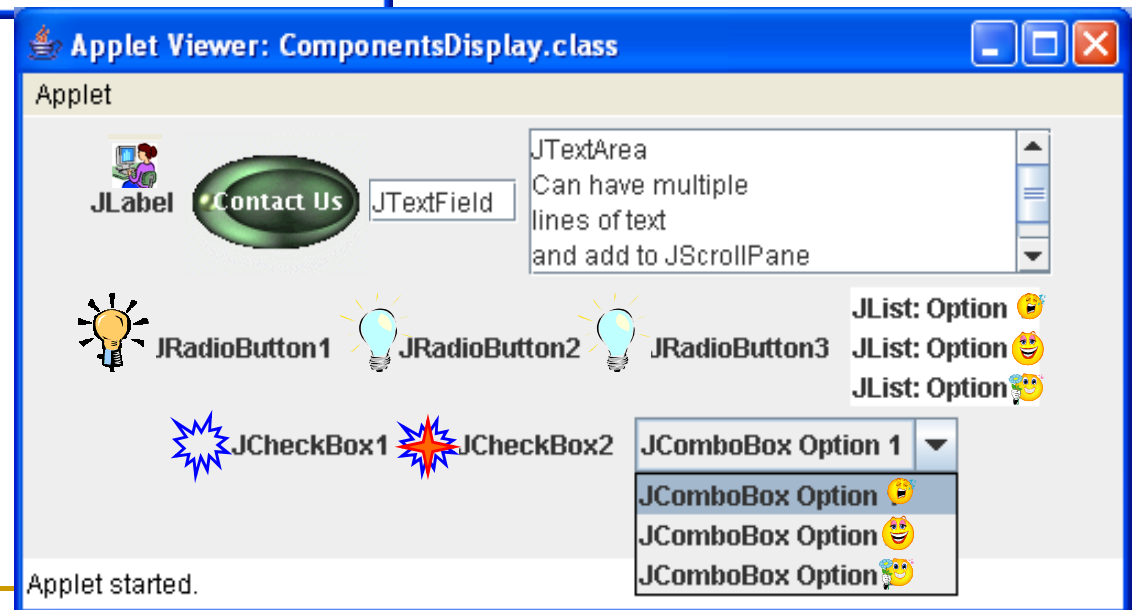
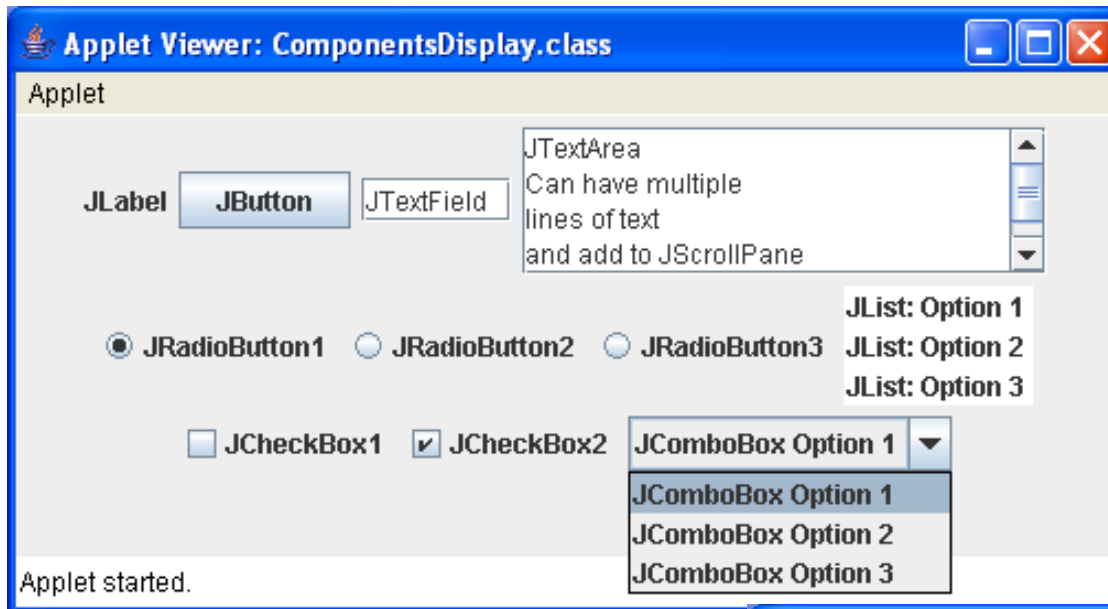

Swing! Components and Images



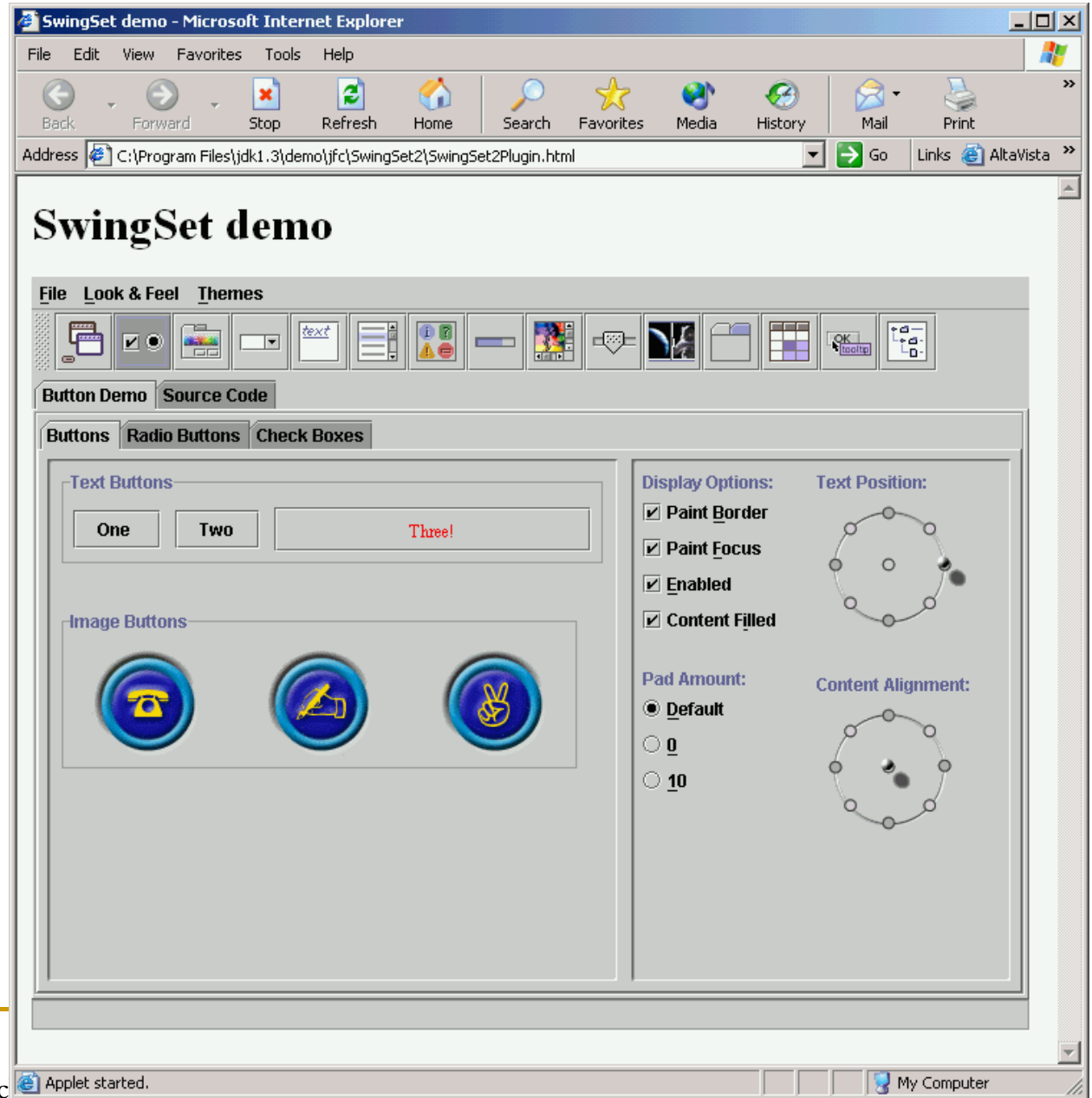
Chapter 4 - Lecture Slides

Example Swing Components



SwingSet demo

■ java.sun.com



Swing

- Swing is a graphics package
- Enhancement of the `java.awt` package
 - + Swing has extra functionality (tooltips, double buffering keyboard shortcuts, etc.)
 - + You can make Swing components look the same on all platforms, whereas awt components will vary on different platforms
 - - Swing is bigger and more complicated
 - - Swing is not supported by earlier JVMs
 - - Awt components tend to be faster than Swing components
- Most class names begin with the capital letter 'J'
 - JLabel, JTextField, JCheckbox, JPanel
- Need to
 - `import javax.swing.*;`

Adding components

Layout managers

add

Adding components

- General structure of a program:

comments with you as author and purpose of the applet
import statements

```
public class name extends JApplet
{
    public void init( )
    {
        create/modify components

        add components to the applet
    }
}
```

Adding components

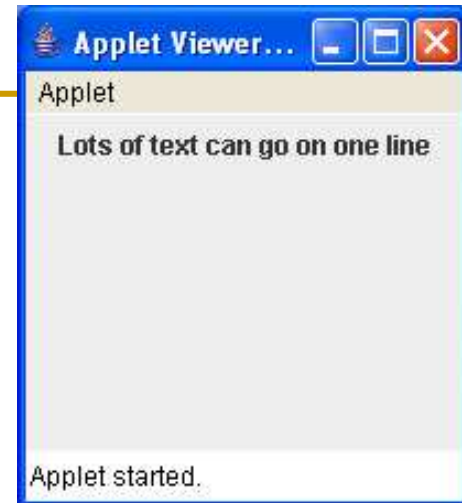
- General structure of a program:

// E.S.Boese Fall 2007 Displays two components

```
import javax.swing.*;
public class Components extends JApplet
{
    // declare components here - data type and variable name
    JLabel title;
    JTextField fullName;
    public void init( )
    {
        // initialize variables here
        title = new JLabel( "My Lovely Applet" );
        fullName = new JTextField( 20 );
        setLayout( new FlowLayout( ) );
        add( title );
        add( fullName );
    }
}
```



Labels



JLabel
HTML



Labels

- Swing component: **JLabel**
- can have



JLabel

- Declare a JLabel

```
JLabel label;
```

- Create a JLabel multiple ways:

- With just text

```
label = new JLabel( "text" );
```

OR

```
label = new JLabel( "text", HorizontalAlignment );
```

- With just an image (image inside an ImageIcon object – see later notes)

```
JLabel label;  
label = new JLabel( ImageIcon );
```

- With text and an image

```
JLabel label;  
label = new JLabel( "text", ImageIcon, HorizontalAlignment );
```

- Where horizontal alignment is either

- JLabel.LEFT
- JLabel.CENTER
- JLabel.RIGHT

JLabel Example

```
Import javax.swing.*;
```

```
Import java.awt.*;
```

```
public class JLabelEx extends JApplet
```

```
{
```

```
    JLabel mylabel;
```

```
    public void init( )
```

```
{
```

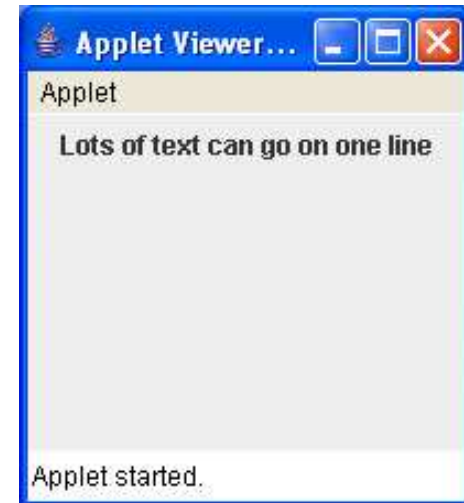
```
    setLayout( new FlowLayout( ) );
```

```
    mylabel = new JLabel( "Lots of text can go on one line" );
```

```
    add( mylabel );
```

```
}
```

```
}
```



JLabel across multiple lines

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

```
public class JLabelExMultiLine extends JApplet  
{
```

```
    JLabel mylabel;
```

```
    public void init( )  
{
```

```
        setLayout( new FlowLayout( ) );
```

```
        mylabel = new JLabel( "<HTML>Lots of text<P>on<P>separate lines" );
```

```
        add( mylabel );
```

```
    }
```

```
}
```



JLabel with funky text

- Lots of flexibility, but need to use **HTML tags**
- HTML tags are *descriptions* of how the text should be displayed
- Each tag is enclosed in angle brackets
- To use these tags, you need to begin the JLabel string with the
- Tags usually have end tags as well, such as **</HTML>** where the slash designates the end of the formatting type

Some HTML tags

- `<P>` or `
`
 - Go to the next line
- `text`
 - Puts the **text** in bold
- `<I>text</I>`
 - Puts the *text* in italics
- `<B I>text</B I>`
 - Puts ***text*** in bold and italics
- `<P style="text-align:center">text</P>`
 - Puts the text centered on each line
- `<P style="color:red">text</P>`
 - Puts **text** in red
- `text`
 - Puts the **text** in a bigger size

JLabel Example with HTML tags

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

```
public class JLabelExWithHTML extends JApplet
```

```
{
```

```
    JLabel label;
```

```
    public void init( )
```

```
    {
```

```
        setLayout( new FlowLayout( ) );
```

```
        label = new JLabel("<HTML>Hi <FONT SIZE=+4 COLOR=RED>there </FONT> <P>world" );
```

```
        add ( label );
```

```
    }
```

```
}
```



Images

Image

ImageIcon

JLabel

Images

- Java can handle the following image types
 - jpg / jpeg
 - gif
 - png
- We can put images in lots of components
 - JLabel
 - JButton
 - JList
 - JCheckBox
 - JRadioButton
 - JTabbedPane
 - More...

Images

- **Add images to applet following 4 steps:**

1) Call to getImage

```
Image img;  
img = getImage ( getCodeBase( ), "computer.gif" );
```

2) Create an ImageIcon with the Image variable created above

```
ImageIcon ic;  
ic = new ImageIcon ( img );
```

3) Create a JLabel with the ImageIcon variable created above

```
JLabel label;  
label = new JLabel ( ic );
```

4) Add label to applet

```
setLayout ( new FlowLayout( ) );  
add ( label );
```

Image Example

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

```
public class ImageExx extends JApplet
```

```
{  
    Image img;  
    ImageIcon ic;  
    JLabel label;  
    public void init ( )  
    {  
        img = getImage( getCodeBase( ), "afraid.gif" );  
        // CANNOT do this:: add( img );  
  
        ic = new ImageIcon( img );  
        // CANNOT DO THIS:: add( ic );  
  
        label = new JLabel( ic );           // have to add to a JLabel  
  
        // then add to the applet  
        setLayout( new FlowLayout( ) );  
        add( label );  
    }  
}
```

Buttons



JButton

Buttons

- Swing component: `JButton`
- Can have
 -
 -
 -
- Customize images based on Mouse movements
 -
 -

Buttons

- Can have



- Text

```
JButton mybutton;  
mybutton = new JButton( "Hello World" );
```



- Image

```
JButton mybutton;  
mybutton = new JButton(
```



- Text and image

```
JButton mybutton;  
mybutton = new JButton(
```



- Text overlaid on top of image

```
JButton mybutton;  
mybutton = new JButton( "text w/ image", ImageIconVariable );  
mybutton.
```

Button Example

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

```
public class JButtonImg extends JApplet  
{
```

```
    JButton one, two, three;
```

```
    Image img;  
    ImageIcon icon;
```

```
public void init( )
```

```
{  
    setLayout( new FlowLayout( ) );
```

```
    one = new JButton( "one" );
```

```
    img = getImage( getCodeBase( ), "idea.png" );
```

```
    icon = new ImageIcon( img );
```

```
    two = new JButton( icon );
```

```
    img = getImage( getCodeBase( ), "oncomputer.png" );
```

```
    three = new JButton( "three", new ImageIcon( img ) );
```

```
    add( one );
```

```
    add( two );
```

```
    add( three );
```

```
}
```

```
}
```



Image Buttons



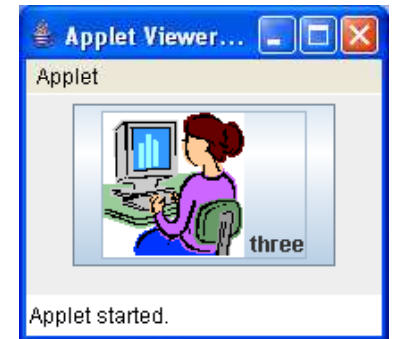
```
JButton mybutton = new JButton( imgIcon );
```

- without border
 - `mybutton.set`
- without background
 - `mybutton.set`
- without border and without background
 - (both lines above)
- without any margins
 - `mybutton.setMargin(`
where first 0 is top margin, second 0 is left, third 0 is bottom margin and last 0 is right
- Without focus border
 - `mybutton.`



Image Buttons with Text on top

- Set the foreground color for the text
 - `mybutton.setForeground(Color.white);`
 - Set the horizontal position with respect to the image
 - `mybutton.setHorizontalTextPosition(JButton.CENTER);`
 - Set the position with respect to the image
 - `mybutton.setVerticalTextPosition(JButton.BOTTOM);`
 - Set the both horizontal (center) and vertical (bottom)
-
- Locations:
 - `JButton.LEFT`
 - `JButton.RIGHT`
 - `JButton.TOP`
 - `JButton.BOTTOM`



JButton Example: font, color, center

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

public class JButtonImg2 extends JApplet
{
    JButton one, two, three;
    Image img;
    ImageIcon icon;

    public void init( )
    {
        setLayout( new FlowLayout( ) );

        one = new JButton( "one" );
        one.setForeground( Color.red );    // change color to red

        img = getImage( getCodeBase( ), "idea.png" );
        icon = new ImageIcon( img );
        two = new JButton( icon );

        img = getImage( getCodeBase( ), "oncomputer.png" );
        three = new JButton( "three is it", new ImageIcon( img ) );
        three.setFont( new Font( "Serif", Font.BOLD, 26 ) ); // change font
        three.setHorizontalTextPosition( JButton.CENTER ); // center

        add( one );
        add( two );
        add( three );
    }
}
```



JButton image manipulation

Customize images based on Mouse movements

- when mouse rolls over

`mybutton.` `(rolloverImageIcon);`



- when user clicks on button

`mybutton.` `(pressedImageIcon);`

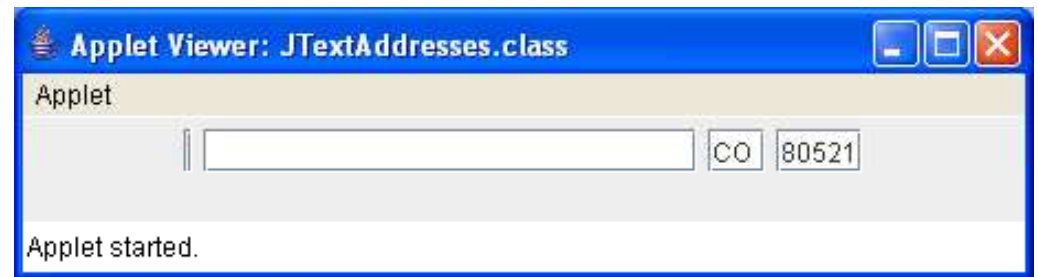


- when button is selected

`mybutton.` `(selectedImageIcon);`

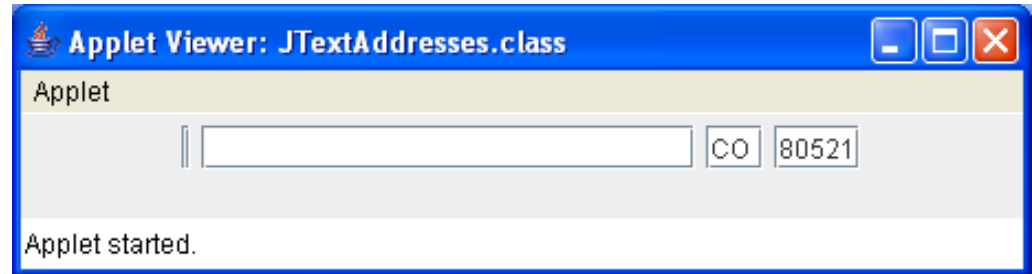


Text Components



JTextField

JTextField



- Box where users can enter ONE line of text

```
JTextField street, city, state, zip;
```

```
street = new JTextField();  
city = new JTextField( 20 );  
state = new JTextField( "CO", 2 );  
zip = new JTextField( "80521" );
```

- **JPasswordField**
 - ❑ for entering passwords
 - ❑ displays stars * while user types, otherwise is just like JTextField

JTextField

- Constructors

- `JTextField tf = new JTextField();`

- creates a text field with a default number (0) of columns

- `JTextField tf = new JTextField(2);`

- creates a text field with 2 columns (good for states: NC, IL)

- `JTextField tf = new JTextField("I love JaVa");`

- creates a text field with the text "I love JaVa" inside the text box – box size fits to text

- `JTextField tf = new JTextField("Java rocks", 10);`

- creates a text field with the text "Java rocks" inside the text box which has 10 columns visible (width)

- **Note:** 1 column = width of 'M' character in current font (though seems more like W)

JTextField

- Useful methods:

- String getText()

- Returns the text that is inside the box

```
String theText = textField.getText( );
```

- void setFont(Font f)

- Set the font for the text box

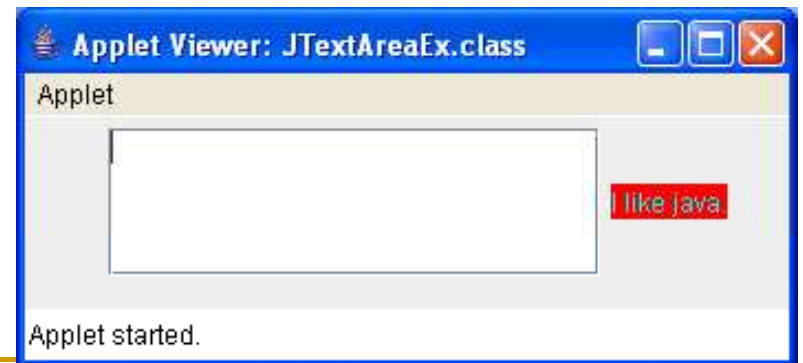
```
Font fnt = new Font( "Serif", Font.BOLD, 18 );  
textField.setFont( fnt );
```

- void setText(String t)

- Enters the text in the string t into the text box

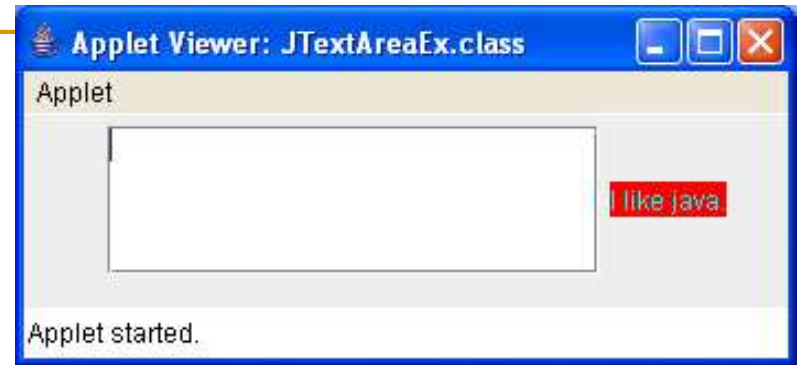
```
textField.setText( "I like to learn java" );
```

Text Components



JTextArea

JTextArea



- Box for users to enter MULTIPLE lines of text
- Specify number of rows and columns
-

```
JTextArea message;  
JScrollPane sp;  
message = new JTextArea( "", 3, 50 );  
sp = new JScrollPane( message );  
add( sp );
```

JTextArea

■ Constructors

- `JTextArea ta;`
`ta = new JTextArea ();`
 - Creates a text area with 0 rows 0 columns

- `JTextArea ta;`
`ta = new JTextArea ();`
 - Creates a text area with 5 rows and 10 columns

- `JTextArea ta;`
`ta = new JTextArea ("I love JaVa");`
 - Creates a text area with the text "I love JaVa" inside the text box

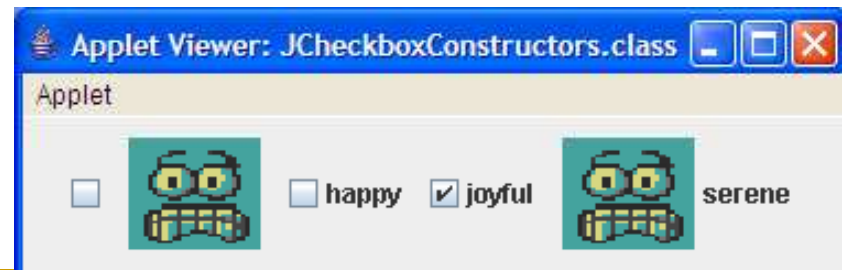
- `JTextArea ta;`
`ta = new JTextArea ();`
 - Creates a text area with the text "Java rocks" inside the text box which has 4 rows visible (height) and 10 columns visible (width)



JTextArea

- Useful methods:
 - String getText()
 - Returns the text that is inside the box
 - `String theText = textField.getText();`
 - void setFont(Font f)
 - Set the font for the text box
 - `Font fnt = new Font("Serif", Font.BOLD, 18);`
 - `textArea.setFont(fnt);`
 - void setText(String t)
 - Enters the text in the string t into the text box
 - `textArea.setText("I like to learn java");`

Lists



JCheckBox

JCheckBox

happy joyful

- A ***checkbox*** can be toggled on or off

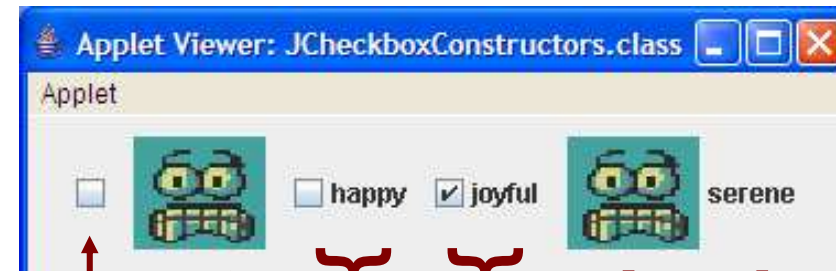


(as opposed to radio buttons)

JCheckBox - constructors

■ Constructors

- ❑ `JCheckBox cb = new JCheckBox();`
 - Creates a new checkbox with no text nor image
- ❑ `JCheckBox cb = new JCheckBox(imgIcon);`
 - Creates a new checkbox with an image
- ❑ `JCheckBox cb = new JCheckBox("happy");`
 - Creates a new checkbox with the text "happy"
- ❑ `JCheckBox cb = new JCheckBox("joyful", true);`
 - Creates a new checkbox with the text "happy" and initially selected
- ❑ `JCheckBox cb = new JCheckBox("serene", imgicon);`
 - Creates a new checkbox with the text "happy" and an image



JCheckBox - methods

■ Useful Methods

□ String getText()

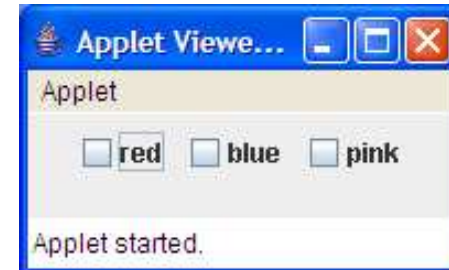
- Returns the checkbox's text
- `String theText = cbox.getText();`

□ void setEnabled(boolean b)

- Enables or disables the checkbox
- `cbox.setEnabled(false);`
- `cbox.setEnabled(true);`

JCheckBox Example

```
/** Example showing use of JCheckBox
 * @author : E.S.Boese (c) Fall 2005
 */
import javax.swing.*;
import java.awt.*;
public class JCheckBoxEx extends JApplet
{
    JCheckBox cb1, cb2, cb3;
    public void init( )
    {
        setLayout( new FlowLayout( ) );
        cb1 = new JCheckBox( "red" );
        cb2 = new JCheckBox( "blue" );
        cb3 = new JCheckBox( "pink" );
        add( cb1 );
        add( cb2 );
        add( cb3 );
    }
}
```



Choices



JRadioButton

Radio Buttons

- A group of *radio buttons* represents a set of mutually exclusive options – only one can be selected at any given time
- When a radio button from a group is selected, the button that is currently "on" in the group is automatically toggled off
- To define the group of radio buttons that will work together,

JRadioButton - constructors

■ Constructors

□ `JRadioButton rb;`

```
rb = new JRadioButton( );
```

- Creates a radio button with an image
- Useful if you set two images: one for selected, one for unselected (otherwise can't tell the difference!)

□ `JRadioButton rb;`

```
rb = new JRadioButton( );
```

- Creates a radio button with the text "red"

□ `JRadioButton rb;`

```
rb = new JRadioButton( );
```

- Creates a radio button with the text "blue" and initially selected

□ `ButtonGroup group;`

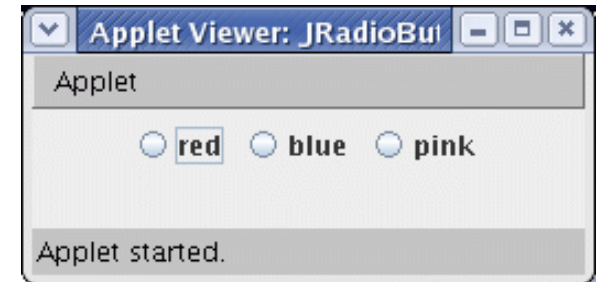
```
group = new ButtonGroup( );
```

- Creates a group that radio buttons can be associated together

JRadioButton - Example

```
Import javax.swing.*;
Import java.awt.*;
public class JRadioButtonEx extends JApplet
{
    JRadioButton jb1, jb2, jb3;
    ButtonGroup group;
    public void init( )
    {
        jb1 = new JRadioButton( "red" );
        jb2 = new JRadioButton( "blue" );
        jb3 = new JRadioButton( "pink" );
        group = new ButtonGroup( );
        group.add( jb1 );
        group.add( jb2 );
        group.add( jb3 );

        setLayout( new FlowLayout( ) );
        add( jb1 );
        add( jb2 );
        add( jb3 );
    }
}
```



How can you add another set of selections for “Favorite Beverage”, such that user can select one color and one beverage?

JRadioButton - methods

- Useful Methods

- String getText()

- returns the text for the radio button
String text = radioBut.getText();

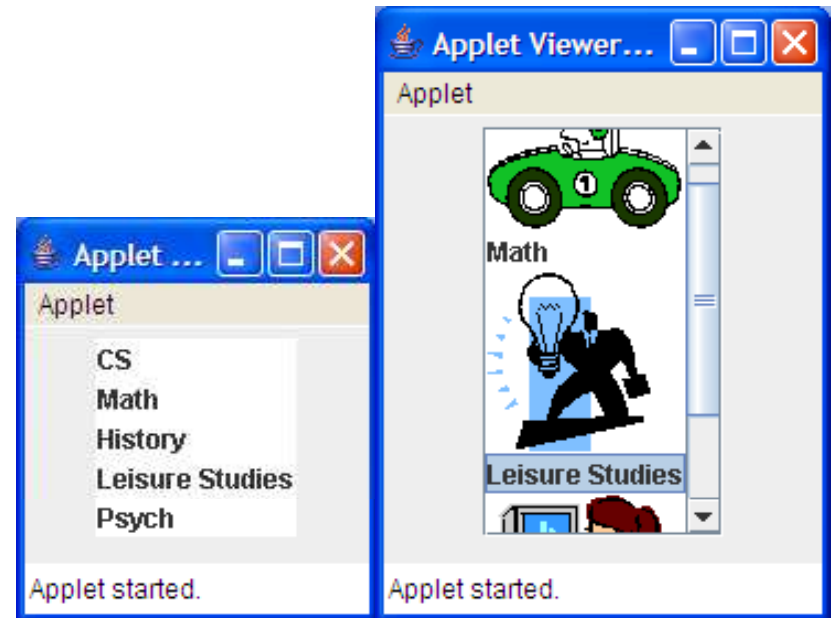
- void setEnabled(boolean)

- enables or disables the radio button
radioBut.setEnabled(false);

- void setFocusable(boolean)

- Removes the focus border around the radio button when the user has selected it

List Choices



JList

JList

- Display a list of items



JList

- First element is at index **0** (zero)
- Create a model to work on the list:
 - `DefaultListModel model;`
`model = new DefaultListModel();`
- Create the list with this model
 - `JList list;`
`list = new JList(model);`
- To add items, use the **addElement** method on the model variable to add items at the end of the list or the **add** method on the model variable and specify the index in the list of where to add it.
 -
 -

JList

- To specify how many rows should be visible (the rest are scrollable) use:

- `list.`

- To remove items, use the **remove** method on the model

- `list.` `// remove item at index 1`

- `list.`

- To get scrollbars, put in a JScrollPane

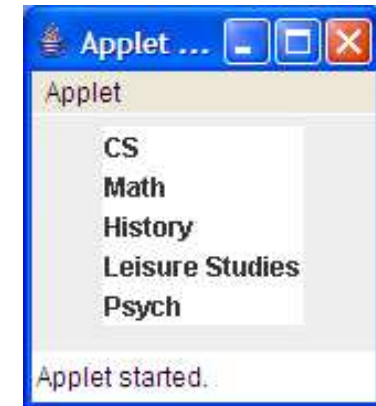
```
JScrollPane spane;  
spane = new JScrollPane( list );  
add( spane );
```

JList – Example - add

```
import javax.swing.*;
import java.awt.*;
public class JListEx extends JApplet
{
    DefaultListModel model;
    JList majors;

    public void init( )
    {
        model = new DefaultListModel( );
        majors = new JList ( model );
        model.add( 0, "CS" );
        model.add( 1, "Math" );
        model.add( 2, "History" );
        model.add( 3, "Leisure Studies" );
        model.add( 4, "Psych" );

        setLayout( new FlowLayout( ) );
        add( majors );
    }
}
```



JList – Example - addElement

```
import javax.swing.*;
import java.awt.*;
public class JListEx extends JApplet
{
    DefaultListModel model;
    JList majors;

    public void init( )
    {
        model = new DefaultListModel( );
        majors = new JList( model );
        model.addElement( "CS" );
        model.addElement( "Math" );
        model.addElement( "History" );
        model.addElement( "Leisure Studies" );
        model.addElement( "Psych" );

        setLayout( new FlowLayout( ) );
        add( majors );
    }
}
```



JList – Example - JScrollPane

```
import javax.swing.*;
import java.awt.*;
public class JListEx extends JApplet
{
    DefaultListModel model;
    JList majors;

    public void init( )
    {
        model = new DefaultListModel( );
        majors = new JList( model );
        model.addElement( "CS" );
        model.addElement( "Math" );
        model.addElement( "History" );
        model.addElement( "Leisure Studies" );
        model.addElement( "Psych" );

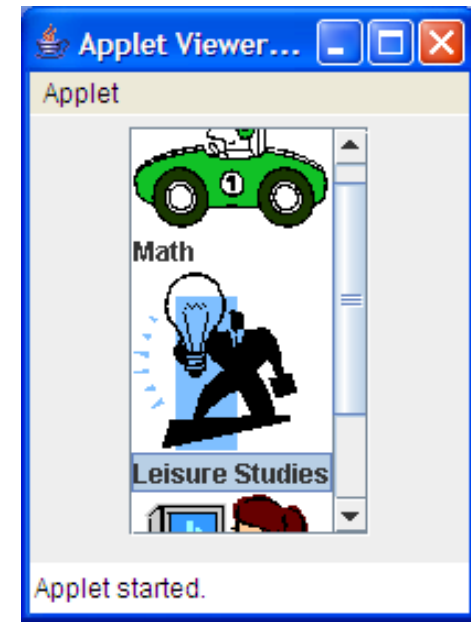
        majors.setVisibleRowCount( 3 );
        JScrollPane spane = new JScrollPane( majors );
        add( spane );
    }
}
```



JList – Example - Images

```
import javax.swing.*;
import java.awt.*;
public class JListExImage extends JApplet
{
    DefaultListModel model;
    JList majors;
    public void init( )
    {
        model = new DefaultListModel( );
        majors = new JList( model );
        Image img = getImage( getCodeBase(), "afraid.gif" );
        ImageIcon ic = new ImageIcon( img );
        model.add( 0, ic );
        model.add( 1, "Math" );
        img = getImage( getCodeBase(), "cars.gif" );
        ic = new ImageIcon( img );
        model.add( 2, ic );
        model.addElement( "Leisure Studies" );
        img = getImage( getCodeBase(), "ferarri.gif" );
        ic = new ImageIcon( img );
        model.addElement( ic );

        setLayout( new FlowLayout( ) );
        majors.setVisibleRowCount( 3 );
        JScrollPane spane = new JScrollPane( majors );
        add( spane );
    }
}
```



JList - constructors

■ Constructors

□ `JList list;`
`list = new JList ();`

- Creates a new empty list

□ `DefaultListModel model;`
`JList list;`
`model = new DefaultListModel();`
`list = new JList(model);`

- Creates a list based on a default model, such that you can add items to the model

□ `JList list;`
`list = new JList(array);`

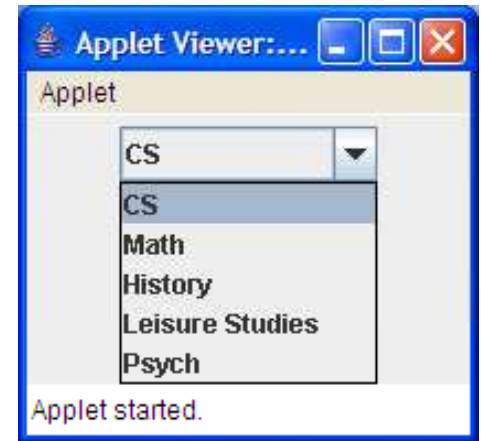
- Creates a list based on the array
- We haven't done arrays yet.... stay tuned

JList - methods

■ Useful Methods

- `int getSelectedIndex()`
 - Returns the index of the item that is selected
- `Object getSelectedValue()`
 - Returns the object that is selected
- `void setSelectionMode(int mode)`
 - Where mode can be:
 - `ListSelectionMode.SINGLE_SELECTION`
 - `ListSelectionMode.MULTIPLE_INTERVAL_SELECTION` (default)
allows users to select more than one in the list
- ... more in API in book and on the website

List Choices



JComboBox

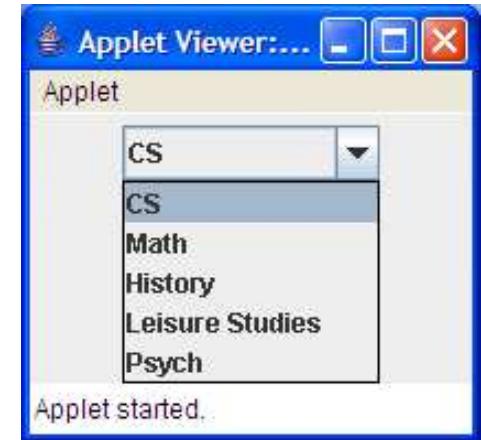
JComboBox

- Drop-down box
 - Only one element can be selected
- Can be:
 - User-editable:
 - Allows user to type in a value in the field similar to a JTextField
 - User-un-editable:
 - User must select an entry from the drop-down list
 - Default

JComboBox - example

```
import javax.swing.*;
import java.awt.*;
public class JComboEx extends JApplet
{
    JComboBox majors;
    public void init( )
    {
        majors = new JComboBox( );
        majors.addItem( "CS" );
        majors.addItem( "Math" );
        majors.addItem( "History" );
        majors.addItem( "Leisure Studies" );
        majors.addItem( "Psych" );

        setLayout( new FlowLayout( ) );
        add( majors );
    }
}
```



JComboBox - constructors

- Constructors

- `JComboBox droplist;`
`droplist = new JComboBox();`
 - Creates an empty combo box

- `JComboBox droplist;`
`droplist = new JComboBox(array);`
 - Creates a list based on the array
 - We haven't done arrays yet.... stay tuned

JComboBox - methods

■ Useful Methods

- `int getItemCount()`
 - Returns the number of items in the list
 - `int numItemsInList = combolist.getItemCount();`
- `int getSelectedIndex()`
 - Returns the index of the selected item
 - `int selectedIndex = combolist.getSelectedIndex();`
- `Object getSelectedItem()`
 - Returns the selected item
 - `String selectedItem = combolist.getSelectedItem();`
- `void removeItem(Object obj)`
 - Removes the object from the list
 - `combolist.remove("sad");`
- `void removeItemAt(int index)`
 - Removes the object at the specified index
 - `combolist.remove(2);`

JComboBox – methods (con't)

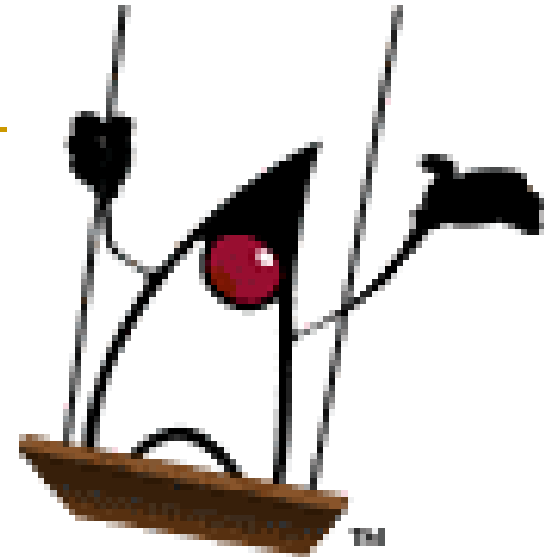
- Useful Methods (con't)
 - void setEditable(boolean flag)
 - Determines whether the combo box is editable
 - `comboBox.setEditable(true);`
 - void setEnabled(boolean flag)
 - Enables or disables the combo box
 - `comboBox.setEnabled(false);`



Components



by
Elizabeth Sugar Boese
© 2005



Component

- These widgets *inherit* from the Component class
 - All methods available in the Component class are also available in all subclasses

```
Component
|
|--JButton
|--JComboBox
|--JLabel
|--JList
|--JTextField
|--JTextArea
```

Components Manipulation

- The Component class has additional methods we can use on most components:
 - ❑ setBackground(Color)
 - ❑
 - ❑ setFont(Font)
 - ❑
 - ❑ setBorder(Border)
 - ❑ setToolTipText(String)
 - ❑ setPreferredSize(Dimension)
 - ❑

```

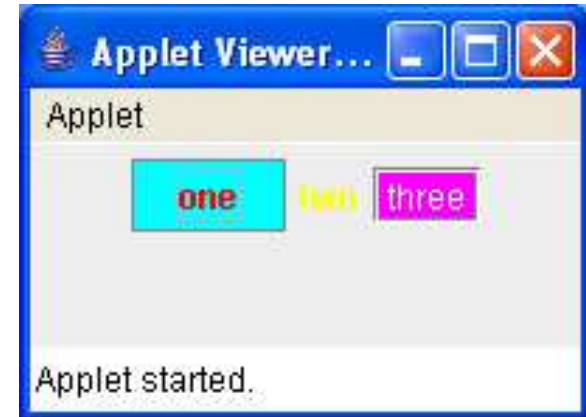
import java.awt.*;
import javax.swing.*;
public class setFGBG JApplet
{
    JButton one;
    JLabel two;
    JTextField three;
    public void init()
    {
        setLayout( new FlowLayout( ) );
        setBackground( Color.BLUE);    // Doesn't work!!

        one = new JButton( "one" );
        one.setForeground( Color.RED );
        one.setBackground( Color.CYAN );

        two = new JLabel( "two" );
        two.setForeground( Color.yellow );
        two.setBackground( Color.blue );    // JLabel is not opaque!
        //two.setOpaque(true);

        three = new JTextField( " three " );
        three.setBackground( Color.MAGENTA );
        three.setForeground( Color.WHITE );
        add( one );
        add( two );
        add( three );
    }
}

```



setBackground, setOpaque

```
import java.awt.*;
import javax.swing.*;
public class OpaqueEx extends JApplet
{
    JCheckBox cb1, cb2, cb3;
    JPanel pane;
    public void init()
    {
        setLayout( new FlowLayout( ) );

        pane = new JPanel( );
        pane.setBackground( Color.ORANGE );
        cb1 = new JCheckBox( "periwinkle" );
        cb2 = new JCheckBox( "snowflake" );
        cb2.setBackground( Color.ORANGE );
        cb3 = new JCheckBox( "magenta" );
        cb3.setOpaque(false);
        pane.add( cb1 );  pane.add( cb2 );  pane.add( cb3 );
        add( pane );
    }
}
```



setEnabled

```
import javax.swing.*;
import java.awt.*;
public class setEnabledEx extends JApplet
{
    JButton one;
    JButton two;
    JButton three;
    public void init( )
    {
        setLayout( new FlowLayout( ) );
        one = new JButton( "one" );
        two = new JButton( "two" );
        three = new JButton( "three" );
        one.setEnabled( false );
        three.setEnabled( false );
        add( one );      add( two );      add( three );
    }
}
```



Summary

- Swing Components
- Labels
- Images
- Buttons
- Text Components
- Choices
- Components