Java Swing Components

by Laurent Henocque
Polytech Marseille
University of Science and Technology of Hanoi
laurent.henocque@univ-amu.fr

Creative Commons License BY-NC-SA

References

- This document contains source excerpts created by John Zukowski and Scott Stanchfield, for the MageLang institute, available at
  - http://java.sun.com/developer/onlineTraining/GUI/Swing1/index.html
- John Zukowski is a Software Mage with MageLang Institute. He is the author of John Zukowski's Definitive Guide to Swing for Java 2, Mastering Java 2, Java AWT Reference, and Borland JBuilder: No experience required.
- Scott Stanchfield is a Software Mage with MageLang Institute. He recently developed and is teaching courses on porting OS/2 applications to Java and advanced features of VisualAge for Java. He is the maintainer of the VisualAge for Java Tips and Tricks resource.

Swing Windowed App Skeleton

```
import ...
public class JWinApp extends JFrame{
	public JWinApp(String title, JPanel panel){
		super(title);
		getContentPane().add(panel, BorderLayout.CENTER);
		setSize(200,200);
		addWindowListener(new WindowAdapter(){
			public void windowClosing(WindowEvent we){ exitApp(); }    });
	}
	protected void exitApp(){
		setVisible(false);
		dispose();
		System.exit(0);
	}
	public static void main(String args[]) { new JWinApp(…).setVisible(true); }
}
```

Hello World


```
class WinHelloPanel extends JPanel implements
    ActionListener{

    JLabel label = new JLabel("Hello World ");

    JButton button = new JButton("Click!");

    public WinHelloPanel(){

        add(label);

        add(button);

        button.addActionListener(this);

    }

    public void actionPerformed(ActionEvent ae){

        JOptionPane.showMessageDialog(this, "Thanks
    for Clicking.");

    }

}
```

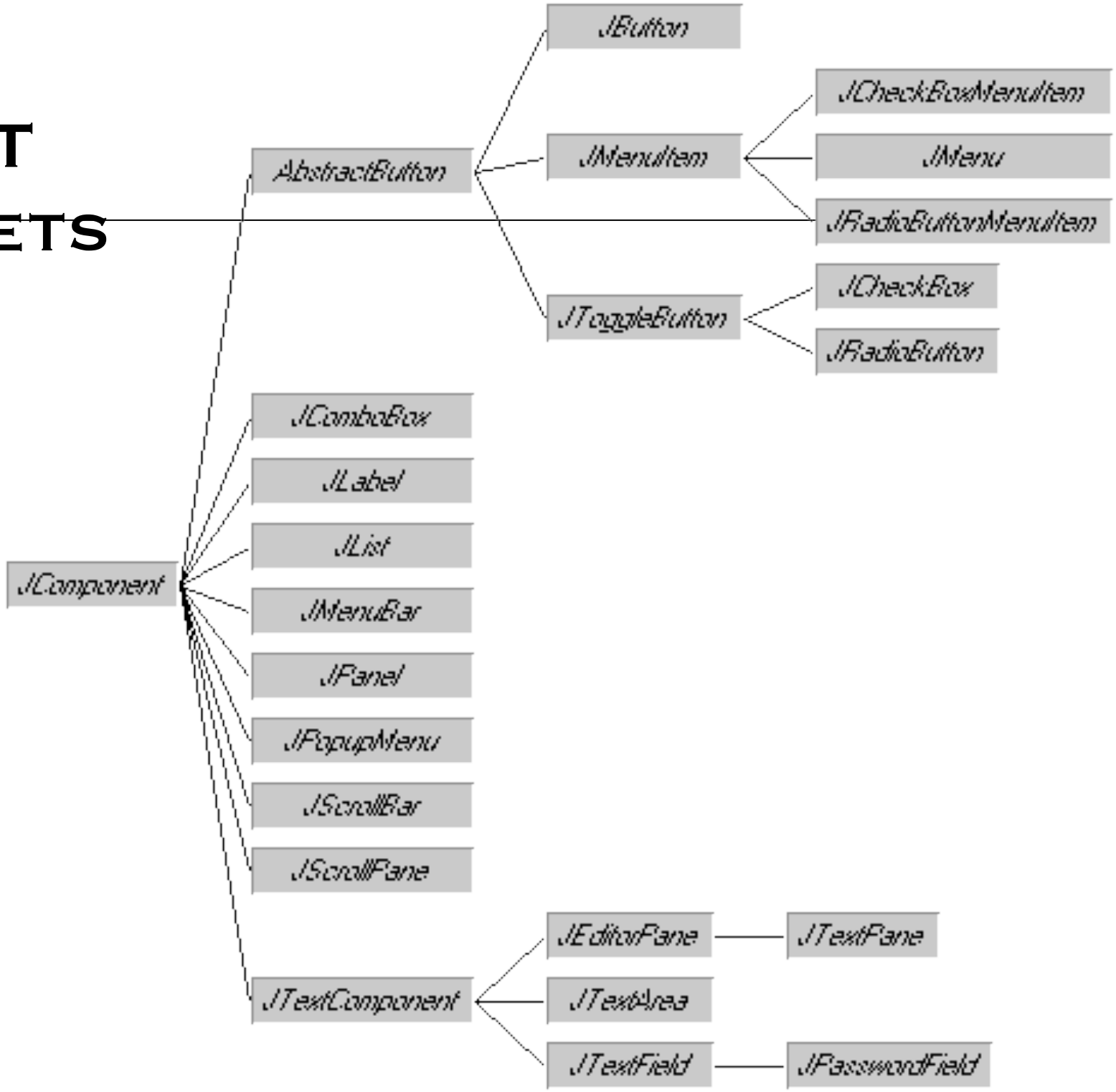
Hello World

- Create a new project for your GUI exercises
- Implement the Hello World program

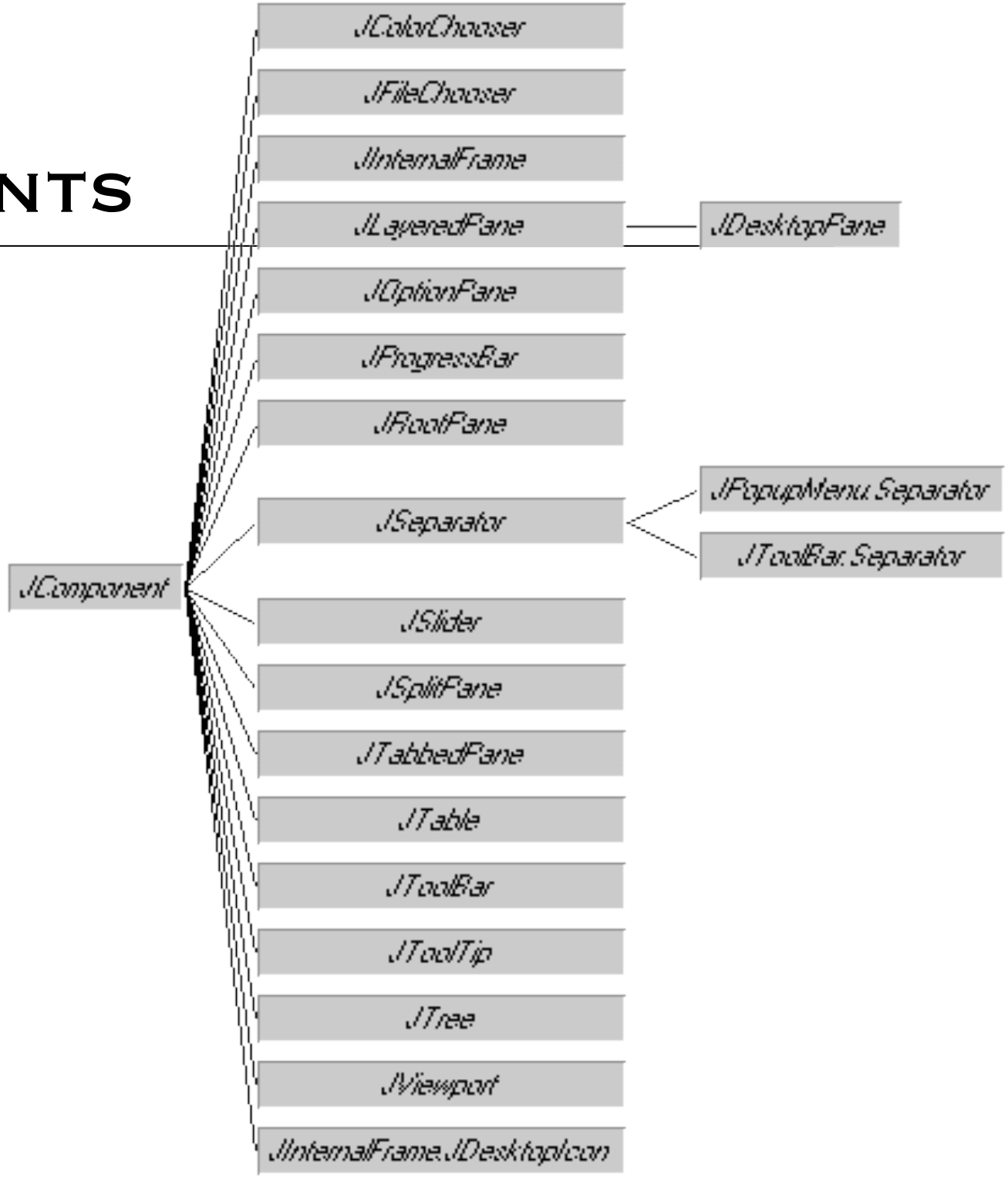
Hello World

```
class HelloWorld extends JPanel implements ActionListener{
JLabel label = new JLabel("Hello World ");
JButton button = new JButton("Click!");
public HelloWorld(){
    add(label);
    add(button);
    button.addActionListener(this);
}
public void actionPerformed(ActionEvent ae){
    JOptionPane.showMessageDialog(this, "Thanks for Clicking.");
}

  public static void main (String args[]) {
    JFrame f = new JFrame ("Hello World");
    JPanel j = new HelloWorld();
    f.addWindowListener(new WindowAdapter() {
      public void windowClosing(WindowEvent e) {
        System.exit(0);
      }
    });
    f.getContentPane().add (j, BorderLayout.CENTER);
    f.setSize (200, 200);
    f.setVisible(true);
  }
}
```

AWT Widgets


JComponent
AbstractButton
JButton
JMenuItem
JCheckBoxMenuItem
JMenu
JRadioButtonMenuItem
JToggleButton
JCheckBox
JRadioButton
JComboBox
JLabel
JList
JMenuBar
JPanel
JPopupMenu
JScrollBar
JScrollPane
JTextComponent
JEditorPane
JTextPane
JTextArea
JTextField
JPasswordField

Swing Components



JPanel

- A lightweight container offering support for double buffering (avoid flickering)

- When the double buffering is active, all components are first drawn into an invisible buffer

ImageIcon

- Icon i = new ImageIcon(“Image.gif");

- Features :
  - url or file,
  - asynchronous: does not block the UI
  - the image is not serializable

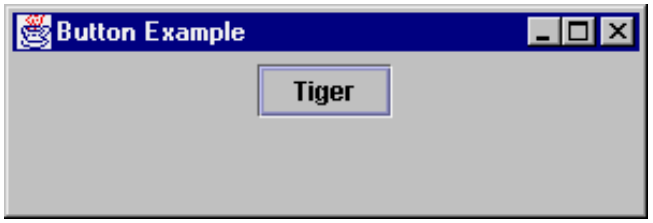
Create a Drawn Icon

```
public class RedOval implements Icon {
  public void paintIcon (
        Component c, Graphics g, int x, int y) {
    g.setColor(Color.red);
    g.drawOval (x, y, getIconWidth(), getIconHeight());
    //g.fillOval (x, y, getIconWidth(), getIconHeight());
  }
  public int getIconWidth() {    return 10;  }
  public int getIconHeight() {     return 10;  }
}
```

JLabel

```
public class LabelPanel extends JPanel {
  public LabelPanel() {
    JLabel plainLabel = new JLabel("Small Label");
    add(plainLabel);

    JLabel fancyLabel = new JLabel(« Fancy Big Label");
    Font fancyFont = new Font("Serif", Font.BOLD | Font.ITALIC, 32);
    fancyLabel.setFont(fancyFont);

    Icon tigerIcon = new ImageIcon("SmallTiger.gif");
    fancyLabel.setIcon(tigerIcon);
    fancyLabel.setHorizontalAlignment(JLabel.RIGHT);
    add(fancyLabel);
  }
}
```

JButton




```
public class ButtonPanel extends JPanel {
  public ButtonPanel() {
    Icon tigerIcon = new ImageIcon("SmallTiger.gif");
    JButton myButton = new JButton("Tiger", tigerIcon);
    add(myButton);
  }
```

AbstractButton

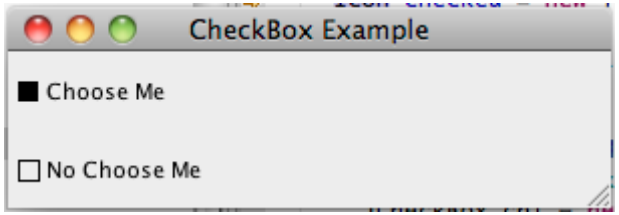
- setMnemonic() – keyboard accelerator: the VK_* constants in KeyEvent
- doClick() – programmatically execute button
- setDisabledIcon(), setDisabledSelectedIcon(), setPressedIcon(), setRolloverIcon(), setRolloverSelectedIcon(), setSelectedIcon() – icon changes
- setVerticalAlignment(), setHorizontalAlignemnt()
- setVerticalTextPosition(), setHorizontalTextPosition() – text vs icon

- Text can be html (<html> … <html>)

JCheckBox

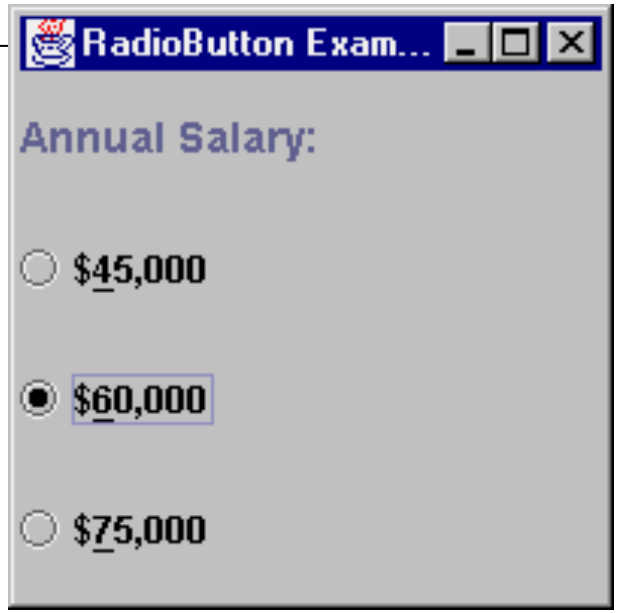
```
class CheckboxPanel extends JPanel {
```

```
  Icon no = new ToggleIcon (false);
  Icon yes = new ToggleIcon (true);

  public CheckboxPanel() {
    setLayout(new GridLayout(2, 1));
    JCheckBox cb1 = new
    JCheckBox("Choose Me", true);
    cb1.setIcon(no);
    cb1.setSelectedIcon(yes);

    JCheckBox cb2 = new
    JCheckBox("No Choose Me", false);
    cb2.setIcon(no);
    cb2.setSelectedIcon(yes);
    add(cb1);     add(cb2);
  }
```



JRadioButton

```
class RadioButtonPanel extends JPanel {
public RadioButtonPanel() {
    setLayout(new GridLayout(4,1));
    JRadioButton radioButton;
    ButtonGroup rbg = new ButtonGroup();
    JLabel label = new JLabel("Annual Salary: ");
    label.setFont(new Font("SansSerif", Font.BOLD, 14));
    add(label);

    radioButton = new JRadioButton("$45,000");
    radioButton.setMnemonic (KeyEvent.VK_4);
    add (radioButton);     rbg.add (radioButton);
    radioButton.setSelected(true);

    radioButton = new JRadioButton("$60,000");
    radioButton.setMnemonic (KeyEvent.VK_6);
    add (radioButton);     rbg.add (radioButton);
    ...
}
```



JToggleButton


```
class ToggleButtonPanel extends JPanel {
  public ToggleButtonPanel() {
    / / Set the layout to a GridLayout
    setLayout(new GridLayout(4,1, 10, 10));

    add (new JToggleButton ("Fe"));
    add (new JToggleButton ("Fi"));
    add (new JToggleButton ("Fo"));
    add (new JToggleButton ("Fum"));
  }
}
```

Methods in JTextComponent

- copy()
- cut()
- paste()
- getSelectedText()
- setSelectionStart()
- setSelectionEnd()
- selectAll()
- replaceSelection()
- getText()
- setText()
- setEditable()
- setCaretPosition()

JTextField & JTextArea

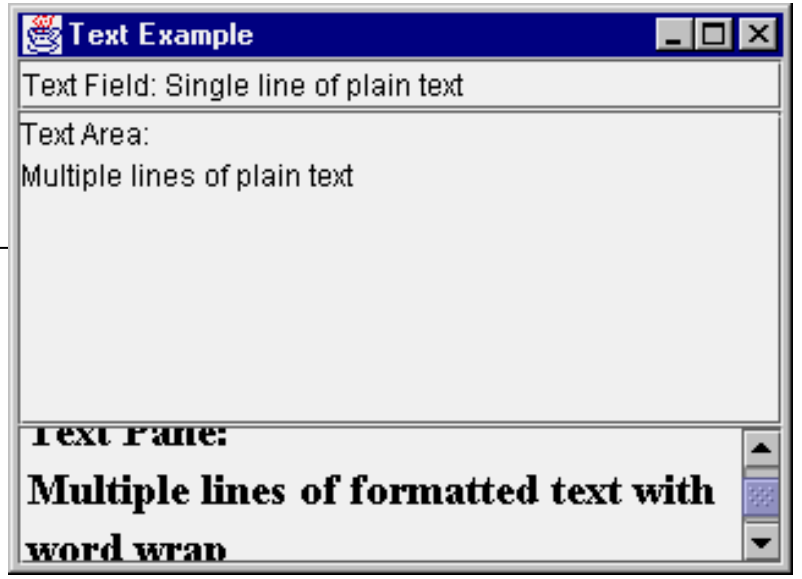
```
JTextField tf = new JTextField();
JTextArea ta = new JTextArea();

tf.setText("TextField");
ta.setText("JTextArea\n Multi Lines");

add(tf);
add(new JScrollPane(ta)); // scroll just in case
```

JTextPane

- JTextPane is a full fledged editor (including image insertion). It deals with a list of styled blocks

```
JTextPane tp = new JTextPane();
MutableAttributeSet attr = new
SimpleAttributeSet();
StyleConstants.setFontFamily(attr, "Serif");
StyleConstants.setFontSize(attr, 18);
StyleConstants.setBold(attr, true);
tp.setCharacterAttributes(attr, false);
add(new JScrollPane(tp));
```

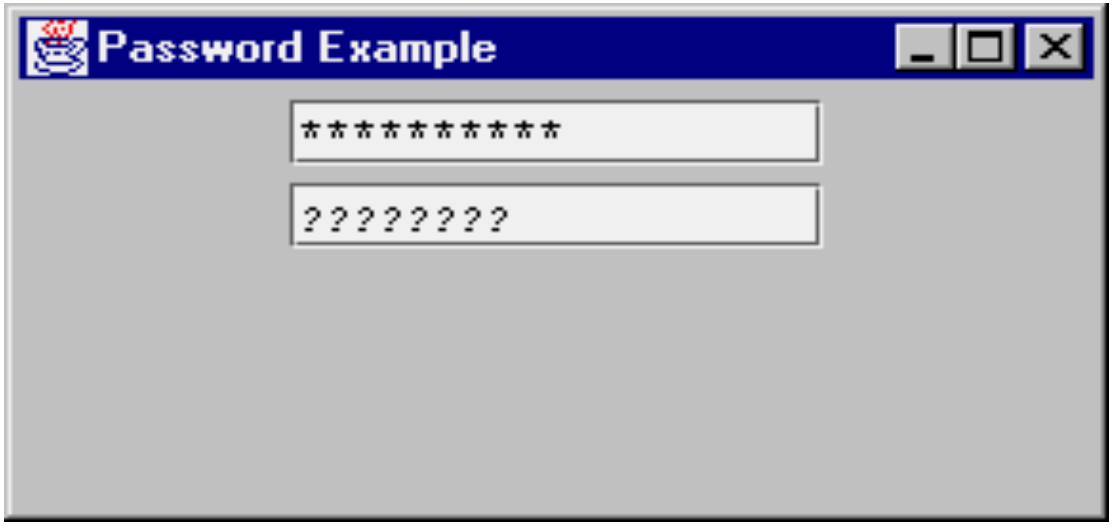
JTextPane example




```
class TextPanel extends JPanel {

  public TextPanel() {
    setLayout(new BorderLayout());

    JTextField textField = new JTextField();
    JTextArea textArea = new JTextArea();
    JTextPane textPane = new JTextPane();

    MutableAttributeSet attr = new SimpleAttributeSet();
    StyleConstants.setFontFamily(attr, "Serif");
    StyleConstants.setFontSize(attr, 18);
    StyleConstants.setBold(attr, true);
    textPane.setCharacterAttributes(attr, false);

    add(textField, BorderLayout.NORTH);
    add(new JScrollPane(textArea), BorderLayout.CENTER);
    add(new JScrollPane(textPane), BorderLayout.SOUTH);}}
```

JPasswordField

```
class PasswordPanel extends JPanel {
  PasswordPanel() {
    JPasswordField p1 = new JPasswordField(20);
    JPasswordField p2 = new JPasswordField(20);
    p2.setEchoChar ('?');
    add(p1);
    add(p2);  }}
```



JEditorPane

- JEditorPane is a text editor allowing the display of html, or rtf text, identified by a URI,
- allowing to follow links

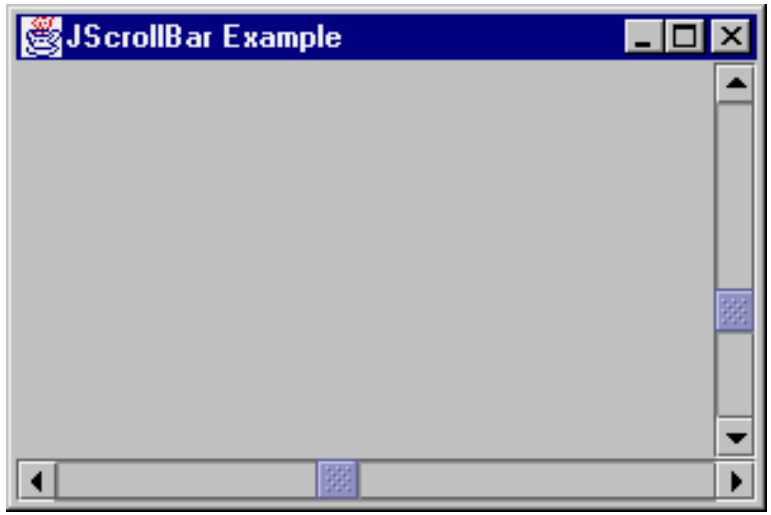
```
class Browser extends JPanel {
  Browser() {
    setLayout (new BorderLayout (5, 5));
    final JEditorPane jt = new JEditorPane();
    final JTextField input = new JTextField("http://java.sun.com");
    jt.setEditable(false);

    // follow links :
    jt.addHyperlinkListener(new HyperlinkListener () {
      public void hyperlinkUpdate(final HyperlinkEvent e) {
        if (e.getEventType() == HyperlinkEvent.EventType.ACTIVATED) {
          SwingUtilities.invokeLater(new Runnable() {
            public void run() { Document doc = jt.getDocument();
              try { URL url = e.getURL(); jt.setPage(url);
                    input.setText (url.toString());
              } catch (IOException io) {
                JOptionPane.showMessageDialog (
                  Browser.this, "Can't follow link", "Invalid Input",
                   JOptionPane.ERROR_MESSAGE);
                jt.setDocument (doc);}}});
}}});
```

CONTINUED

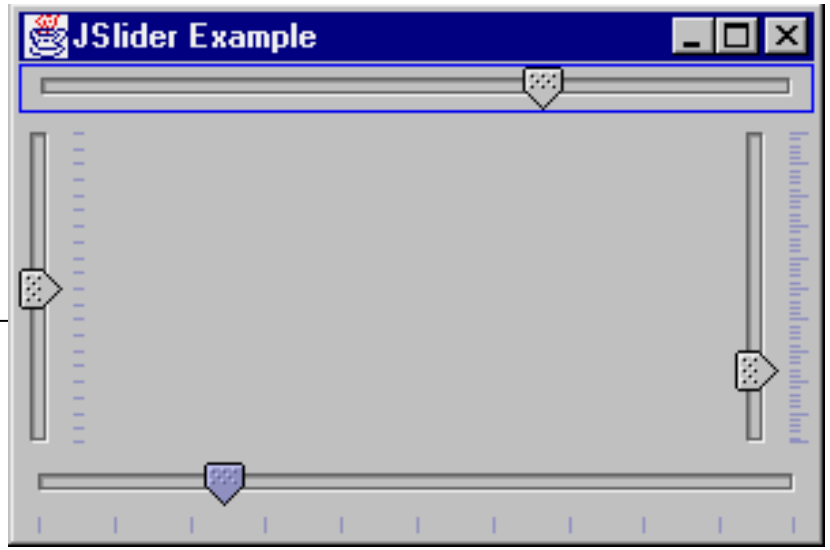
```
JScrollPane pane = new JScrollPane();
   pane.setBorder (
    BorderFactory.createLoweredBevelBorder());
   pane.getViewport().add(jt);
   add(pane, BorderLayout.CENTER);
   input.addActionListener (new ActionListener() {
    public void actionPerformed (ActionEvent e) {
     try {   jt.setPage (input.getText());        }
      catch (IOException ex) {
      JOptionPane.showMessageDialog (
        Browser.this, "Invalid URL",  "Invalid Input",
        JOptionPane.ERROR_MESSAGE);  }
    } });
   add (input, BorderLayout.SOUTH);}}
```

JScrollBar

```
class ScrollbarPanel extends JPanel {
  public ScrollbarPanel() {
    setLayout(new BorderLayout());


    JScrollBar sb1 =
      new JScrollBar (JScrollBar.VERTICAL, 0, 5, 0, 100);
    add(sb1, BorderLayout.EAST);


    JScrollBar sb2 =
      new JScrollBar (JScrollBar.HORIZONTAL, 0, 5, 0, 100);
    add(sb2, BorderLayout.SOUTH);
  }
}
```



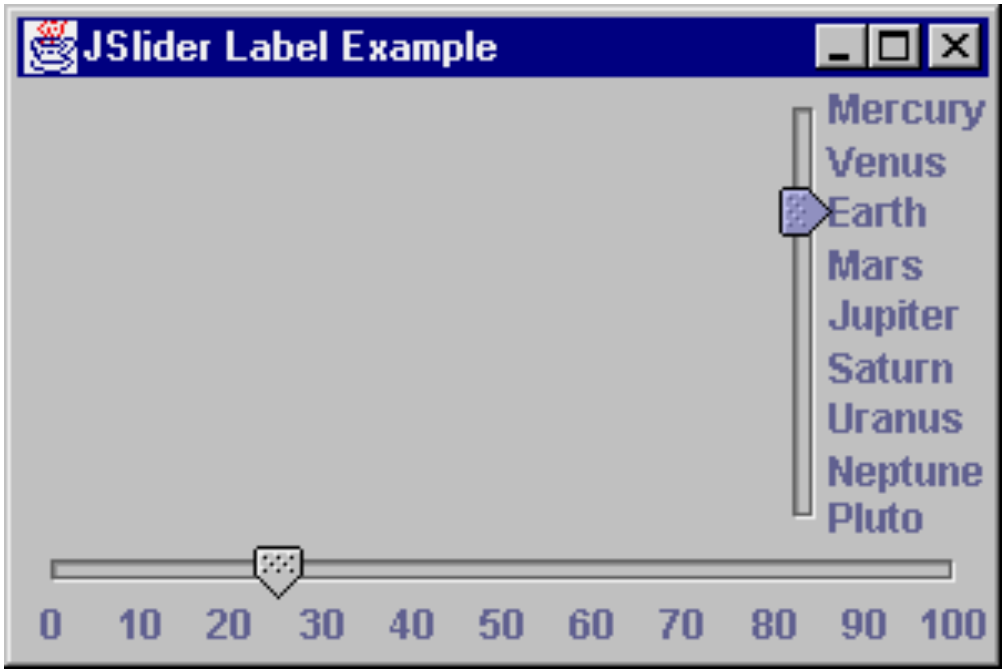
JSLIDER




```
public class SliderPanel extends JPanel {
  public SliderPanel() {
    setLayout(new BorderLayout());

    JSlider s1 = new JSlider (JSlider.VERTICAL, 0, 100, 50);
    s1.setPaintTicks(true);
    s1.setMajorTickSpacing(10);     s1.setMinorTickSpacing(2);
    add(s1, BorderLayout.EAST);

    JSlider s2 = new JSlider (JSlider.VERTICAL, 0, 100, 50);
    s2.setPaintTicks(true);     s2.setMinorTickSpacing(5);
    add(s2, BorderLayout.WEST);

    JSlider s3 = new JSlider (JSlider.HORIZONTAL, 0, 100, 50);
    s3.setPaintTicks(true);     s3.setMajorTickSpacing(10);
    add(s3, BorderLayout.SOUTH);

    JSlider s4 =
      new JSlider (JSlider.HORIZONTAL, 0, 100, 50);
    s4.setBorder(BorderFactory.createLineBorder(Color.blue));
    add(s4, BorderLayout.NORTH);
  }}
```

JSlider And Labels

```
public class SliderPanel2 extends JPanel {
  public SliderPanel2() {
    setLayout(new BorderLayout());

    JSlider right, bottom;
    right = new JSlider(JSlider.VERTICAL, 1, 9, 3);
    Hashtable h = new Hashtable();
    h.put (new Integer (1), new JLabel("Mercure"));

    right.setLabelTable (h);
    right.setPaintLabels (true);
    right.setInverted (true);

    bottom = new JSlider(JSlider.HORIZONTAL, 0, 100, 25);
    bottom.setMajorTickSpacing (10);
    bottom.setPaintLabels (true);
    add(right, BorderLayout.EAST);
    add(bottom, BorderLayout.SOUTH);
  }
}
```



JProgressBar

- Create Progress Bar

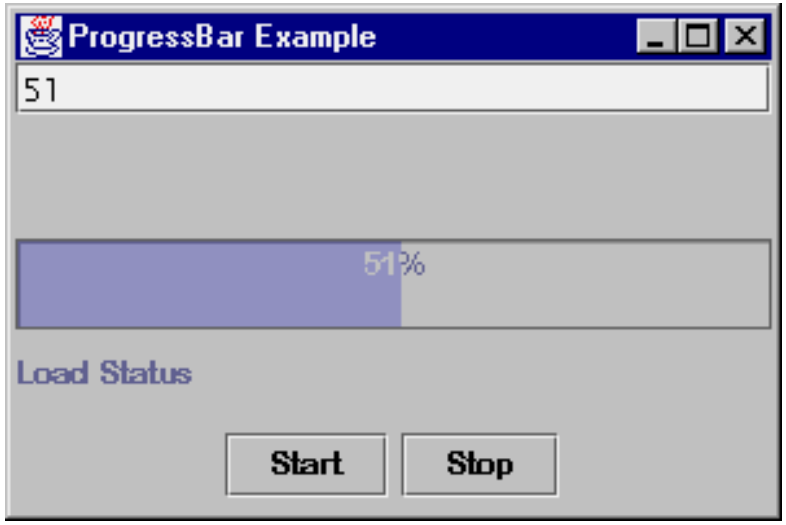
```
progressBar = new JProgressBar(0, task.getLengthOfTask());
progressBar.setValue(0);
progressBar.setStringPainted(true);
```

- Change the current value :

```
progressBar.setValue(task.getCurrent());
```

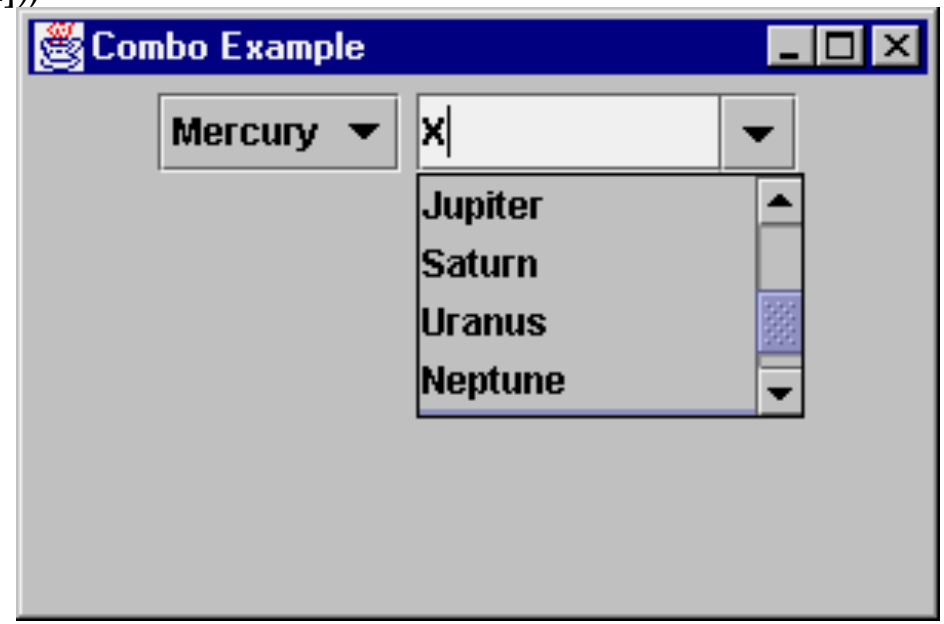
- Utilising the « indeterminate » mode

```
progressBar = new JProgressBar();
progressBar.setIndeterminate(true);
... // now the size is known
progressBar.setMaximum(newLength);
progressBar.setValue(newValue);
progressBar.setIndeterminate(false);
```



JComboBox

```
public class ComboPanel extends JPanel {
  String choices[] = {"Mercure", "Venus", "Terre", "Mars",
"Jupiter", "Saturne", "Uranus", "Neptune", "Pluton"};
  public ComboPanel() {
    JComboBox combo1 = new JComboBox();
    JComboBox combo2 = new JComboBox();
    for (int i=0;i<choices.length;i++) {
      combo1.addItem (choices[i]); combo2.addItem (choices[i]);
    }
    combo2.setEditable(true);
    combo2.setSelectedItem("X");
    combo2.setMaximumRowCount(4);
    add(combo1);     add(combo2);
  }
  public static void main (String args[]) {
…   } }
```

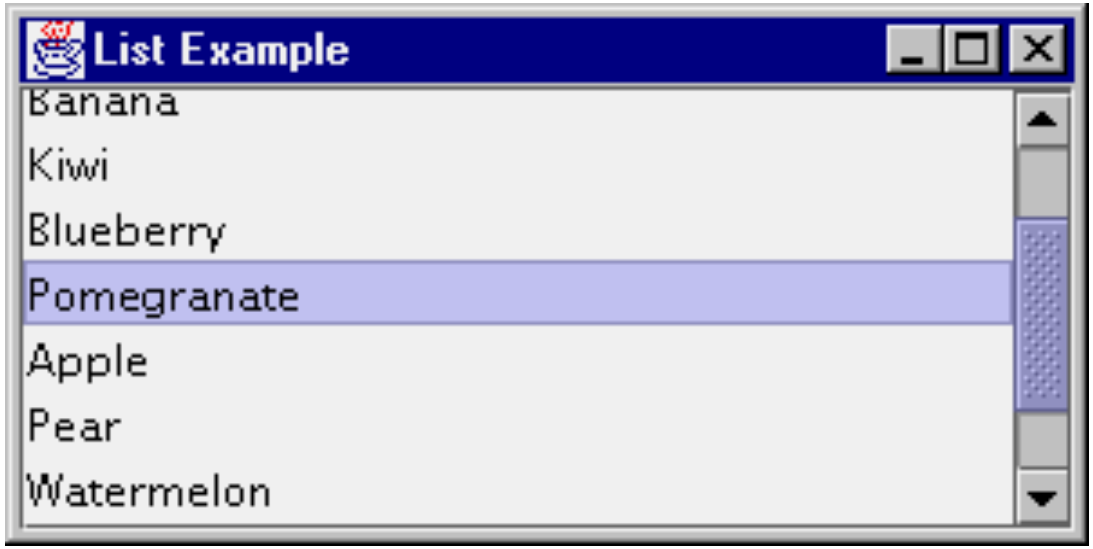


ComboBox Callbacks

```
public class ComboBoxDemo implements ActionListener {
    . . .
    combo.addActionListener(this);
    . . .
    public void actionPerformed(ActionEvent e) {
        JComboBox cb = (JComboBox)e.getSource();
        String item = (String)cb.getSelectedItem();
         . . .
    }
. . .
}
```

JList

```
String label [] = {"a", "b", "c", "d", "e", "f", "g", "h", "i", "j",
    "k" };

JList list = new JList(label);

JScrollPane pane = new JScrollPane(list);
```



JList : Selection

```
static Vector v;

l = new JList(v);

l.setSelectionMode(ListSelectionModel.SINGLE_SEL
    ECTION);

    / / SINGLE_INTERVAL_SELECTION

    / / MULTIPLE_INTERVAL_SELECTION
```

dynamic Modification of JList

```
listModel = new DefaultListModel();
listModel.addElement("A");
listModel.addElement("B");
listModel.addElement("C");

 JList list = new JList(listModel);

 listModel.remove(index);
```

ListSelectionListener

```
public void valueChanged(ListSelectionEvent e) {
    if (e.getValueIsAdjusting()) return;

    JList theList = (JList)e.getSource();
    if (theList.isSelectionEmpty()) {
        ...
    } else {
        int index = theList.getSelectedIndex();
        ...
    }
}
```

Borders

JButton b = new JButton("Empty"); b.setBorder (new EmptyBorder (1,1,1,1));

b = new JButton ("Etched"); b.setBorder (new EtchedBorder ());

b = new JButton ("ColorizedEtched"); b.setBorder (new EtchedBorder (Color.red, Color.green));

b = new JButton ("Titled/Line");
b.setBorder(new TitledBorder (
    new TitledBorder(LineBorder.createGrayLineBorder(),"Hello"), "World",
    TitledBorder.RIGHT, TitledBorder.BOTTOM));

b = new JButton ("Bevel Up"); b.setl

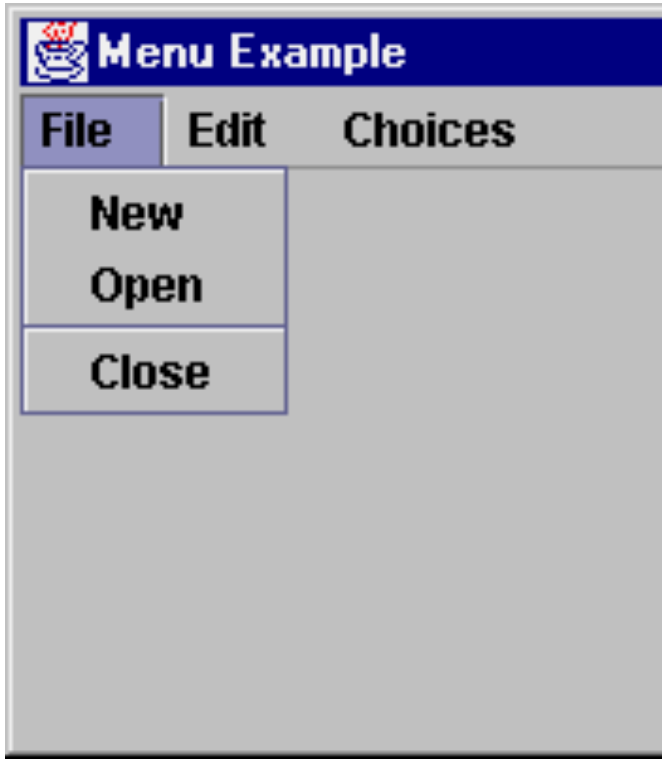
b = new JButton ("Bevel Down"); b.

Borders (2)


```
SoftBevelBorder(SoftBevelBorder.RAISED);
SoftBevelBorder(SoftBevelBorder.LOWERED);
MatteBorder(5, 10, 5, 10, Color.red);
Icon icon = new ImageIcon ("file.gif");
 new MatteBorder(10, 10, 10, 10, icon));
BevelBorder(BevelBorder.RAISED, Color.red, Color.pink));
CompoundBorder(
      new MyBorder(Color.red),
      new CompoundBorder (new MyBorder(Color.green),
                  new MyBorder(Color.blue)));
```

Menus

```
JMenuBar jmb = new JMenuBar();
JMenu file = new JMenu ("File");
file.addMenuListener (new MenuListener() {
    public void menuSelected (MenuEvent e) { ... }
    public void menuDeselected (MenuEvent e) { ... }
    public void menuCanceled (MenuEvent e) { ... }
});
JMenuItem item;
file.add (item = new JMenuItem ("New"));
file.add (item = new JMenuItem ("Open"))
file.addSeparator();
file.add (item = new JMenuItem ("Close"));
jmb.add (file);

 setJMenuBar (jmb);
```



Callbacks on menu items

```
menuItem.addActionListener(this);

...

//JRadioButtonMenuItem:

rbMenuItem.addActionListener(this);

...

//JCheckBoxMenuItem:

cbMenuItem.addItemListener(this);
```

Sub menus

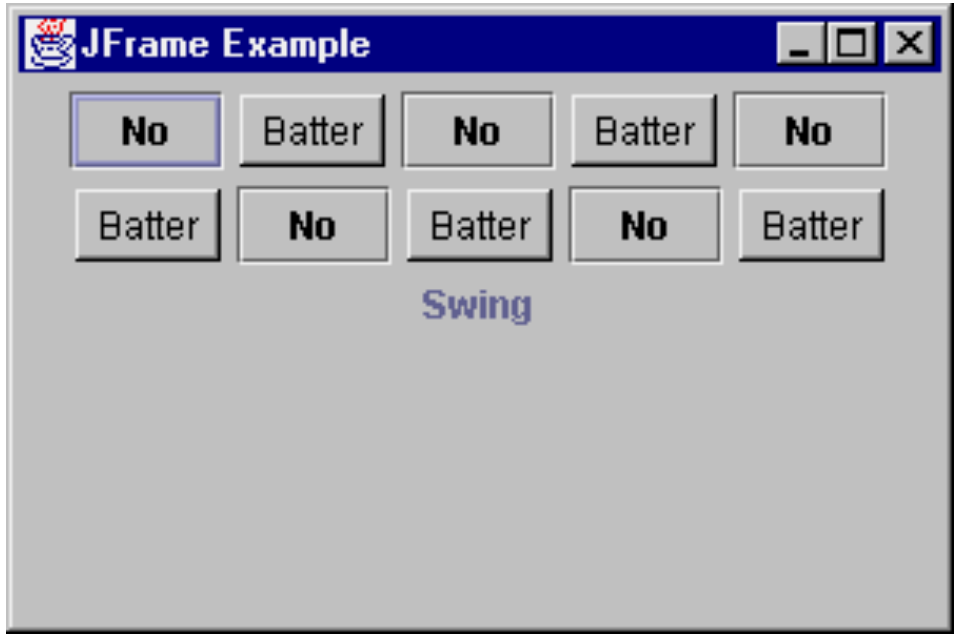
```
submenu = new JMenu("A submenu");
submenu.setMnemonic(KeyEvent.VK_S);
menuItem = new JMenuItem("dans le sous menu");
menuItem.setAccelerator(KeyStroke.getKeyStroke(
        KeyEvent.VK_2, ActionEvent.ALT_MASK));

submenu.add(menuItem);
...
menu.add(submenu);
```

JPopupMenu

```
public class PopupPanel extends JPanel {
  JPopupMenu popup = new JPopupMenu ();
  public PopupPanel() {
       popup.add (new JMenuItem ("Cut"));
       ...
       popup.setInvoker (this);

       addMouseListener (new MouseAdapter() {
        public void mousePressed (MouseEvent e) {
          if (e.isPopupTrigger()) {
             popup.show (e.getComponent(), e.getX(), e.getY());
          } }
         public void mouseReleased (MouseEvent e) {
          if (e.isPopupTrigger()) {
            popup.show (e.getComponent(), e.getX(), e.getY());
       } } });  }
```

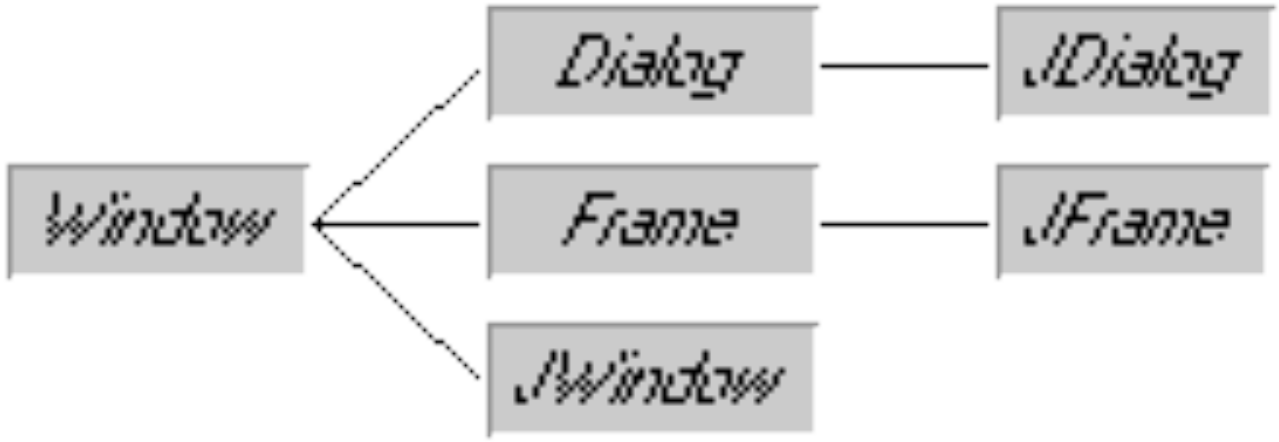
JFrame and Windows

```
public class FrameTester {
  public static void main (String args[]) {
    JFrame f = new JFrame ("JFrame Example");
    Container c = f.getContentPane();
    c.setLayout (new FlowLayout());
    for (int i = 0; i < 5; i++) {
      c.add (new JButton ("No"));
      c.add (new Button ("Batter"));
    }
    c.add (new JLabel ("Swing"));
    ...
}
```



Hierarchy of Swing Windows

- Located below the AWT class 'Window'



- Not "lightweight", bound to a graphic window, and cannot be transparent.
- Support setJMenuBar().
- As for JWindow et JDialog, elements must be added to a container Panel obtained using getContentPane()

JFrame default close operation

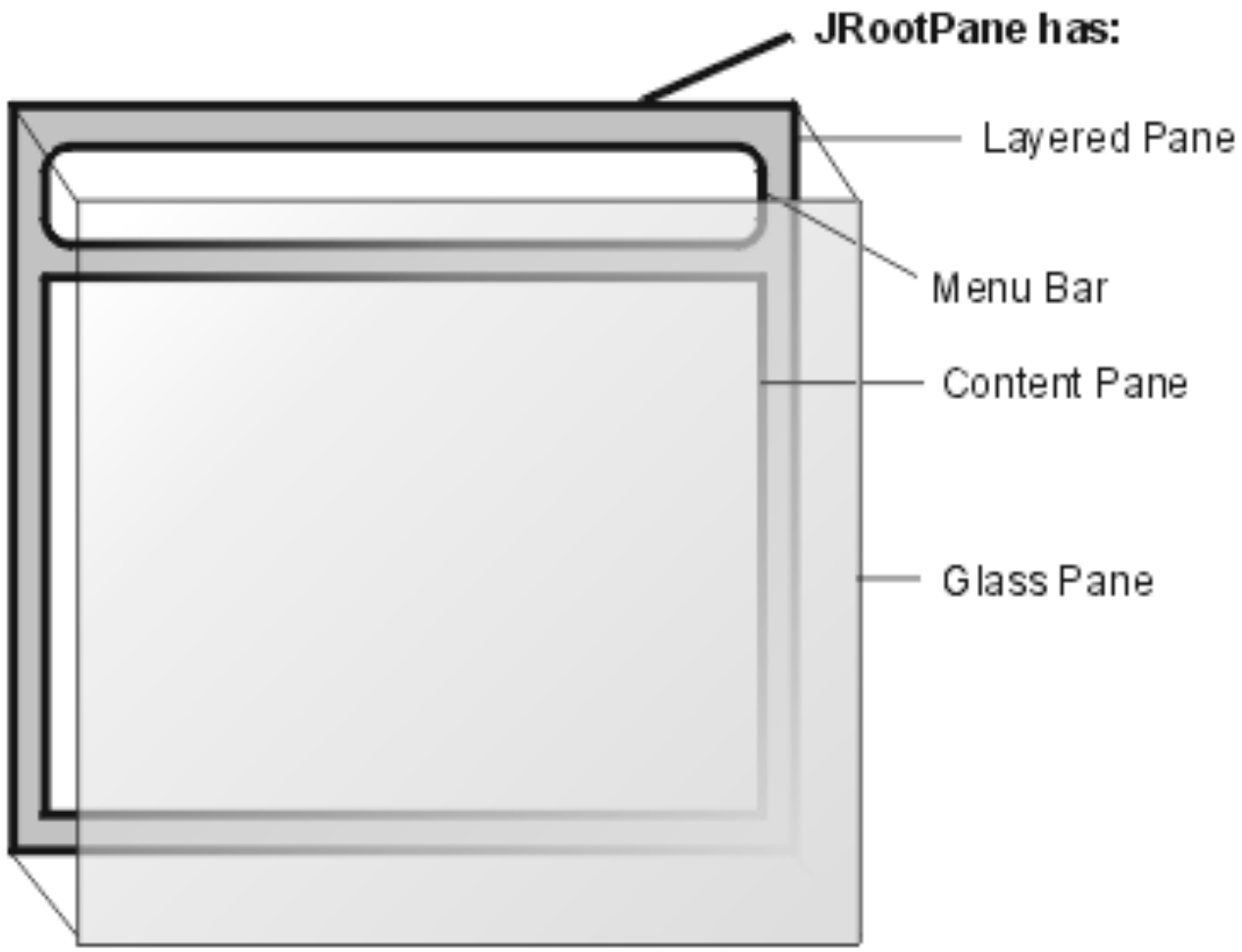
- setDefaultCloseOperation(int operation) :
  - EXIT_ON_CLOSE: use for the main app window
  - DO_NOTHING_ON_CLOSE:
  - HIDE_ON_CLOSE: default (setVisible(true) maps the frame again)
  - DISPOSE_ON_CLOSE: like HIDE but saves the resources (makes remapping slower)

- HIDE_ON_CLOSE et DISPOSE_ON_CLOSE let event listeners execute

JRootPane, JLayeredPane

- A JFrame displays using a JRootPane, composed of two objects : a glass pane and a layered pane.
- The glass pane is invisible, but always in front of the layered pane, to display tooltips and popups
- The layered pane has an optional menubar, and a content pane
- As the name suggests, the layered pane allows displaying in layers

JFrame Structure



JLayeredPane

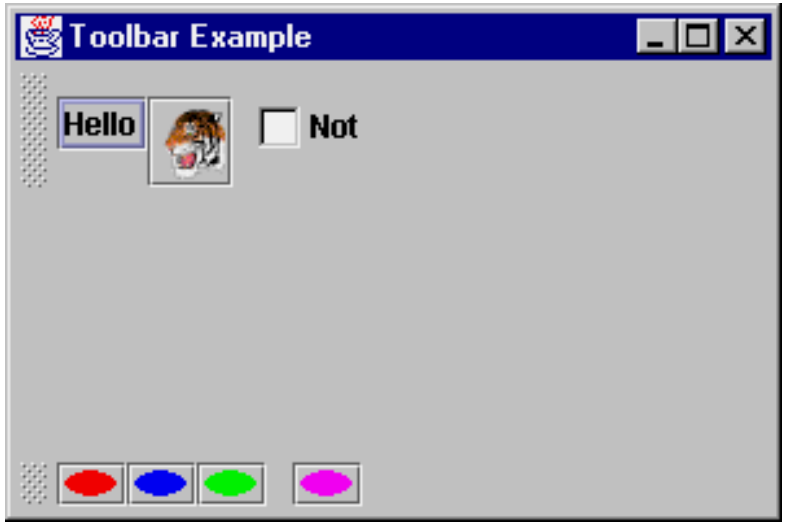
- The JLayeredPane allows for superpositions:
- layeredPane.add (component, new Integer(5));
- The default layer is JLayeredPane.DEFAULT_LAYER.
- Objects may be placed relative to this layer, front or back
- The LayoutManager computes the print sequence, and blocks superpositions within the same layer

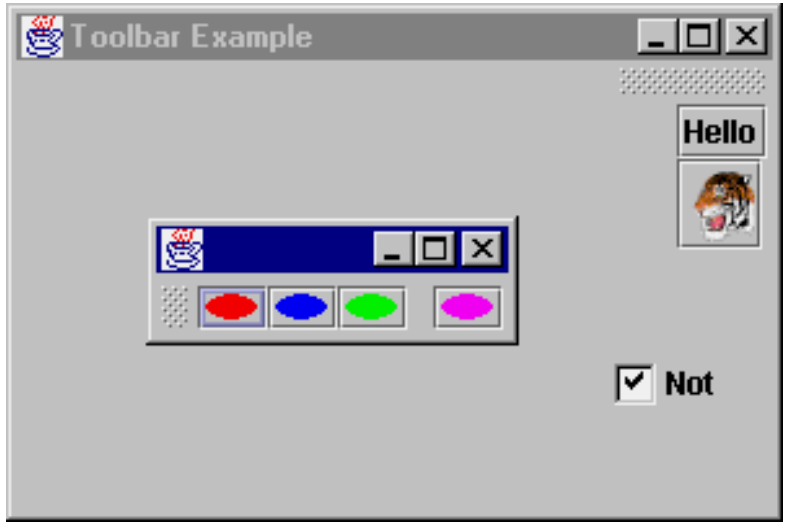
Tooltips

```
public class TooltipPanel extends JPanel {
  public TooltipPanel() {
    JButton myButton = new JButton("Hello");
    myButton.setToolTipText ("World");
    add(myButton);
  }
```

Toolbars

- JToolbar allows for moving toolbars around, even in a container different from the original.
- The orientation is context based
- Floating can be disabled.
- aToolBar.setFloatable (false);



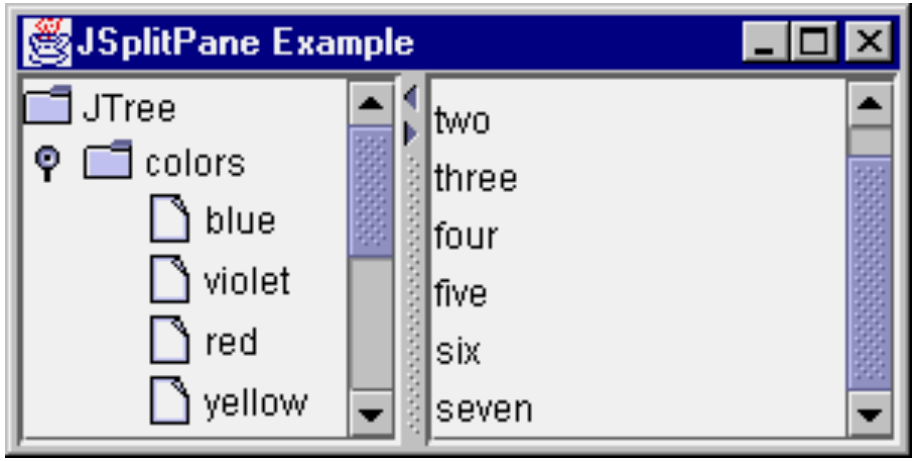


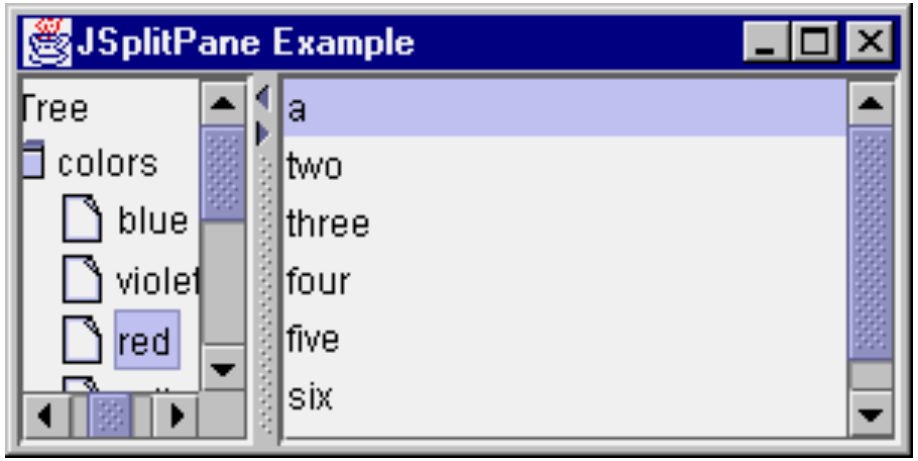
JTabbedPane

- JTabbedPane allows for handling tabbed panes.
- Tabs are added using addTab(). There is tooltip support
- Any component may fit in a tab
  - addTab(String title, Component component)
  - addTab(String title, Icon icon, Component component)
  - addTab(String title, Icon icon, Component component, String tip)

JSplitPane

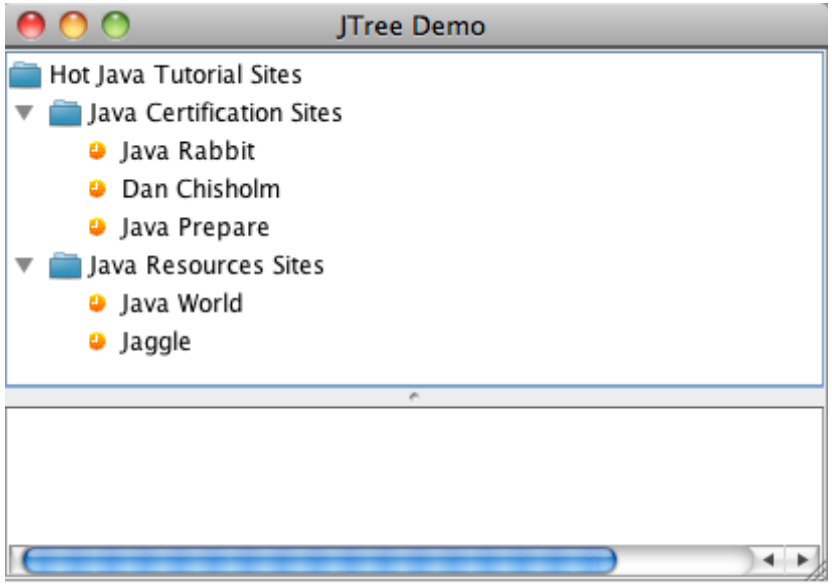
- JSplitPane allows to negotiate the space between two panels
- JSplitPane can be embedded: an alternative to complex layouts
- setContinuousLayout allows to view the change in real time
- The "dividerLocation" may be programmatically set





JTree

- The JTree component allows for handling hierarchical trees, that can be folded/expanded manually or programmatically



Lab Work: a Swing Tutorial

- Follow the 'Fundamentals of Swing Tutorial'

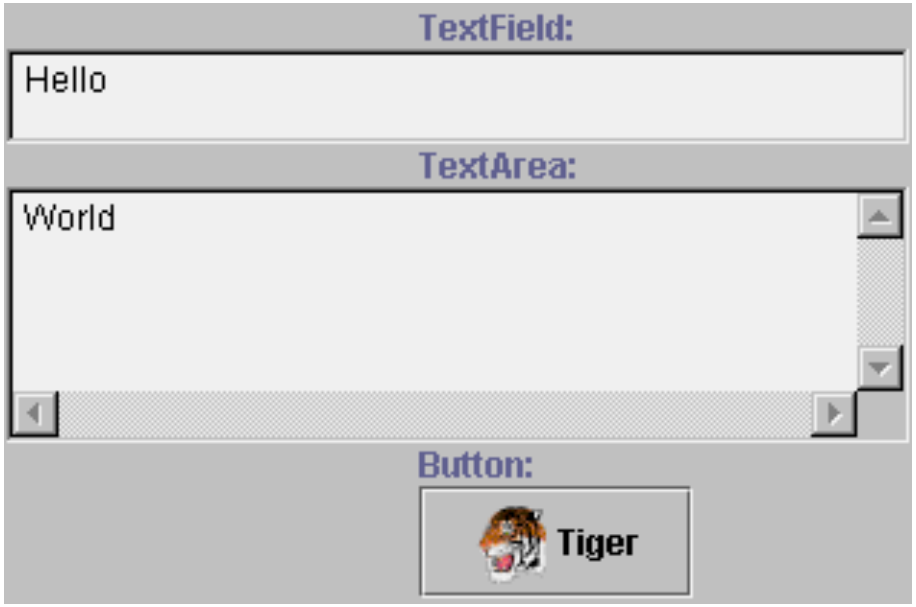
Layouts

What are layouts?

- Layout managers help grouping objects within the displayed interface
- They implement constraints controlling objects preferred sizes, placement preferences, groupings, spaces between them, alignment etc.
- There exist several kinds of layouts

BoxLayout

- The BoxLayout places components along the horizontal or vertical axis like the grid layout, but with more flexibility, as proportions may vary
- The components that cannot be re-dimensioned will be centered
- setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
- add(myComponent);



Tips on Choosing a Layout Manager

- You need to display a **single component** in as much space as it can get.
  - Use BorderLayout or GridLayout (opt. GridBagLayout)
  - BorderLayout: put the component in the center.
  - GridBagLayout, set the constraints so that fill=GridBagConstraints.BOTH.
- You need to display **a few components in a compact row at their natural size**.
  - Use a JPanel to group the components and use either the JPanel's default FlowLayout manager or the BoxLayout manager. (opt. SpringLayout).
- You need to display **a few components of the same size in rows and columns**.
  - GridLayout is perfect for this.
- You need to display a few components in a row or column, possibly with varying amounts of space between them, custom alignment, or custom component sizes.
  - BoxLayout is perfect for this.

Tips on Choosing a Layout Manager

- You need to display **aligned columns**, as in a form-like interface where a column of labels is used to describe text fields in an adjacent column.
  - SpringLayout. (however normally used by GUI builders).
- You have a **complex layout with many components**.
  - Either use GridBagLayout or SpringLayout, or group into JPanels to recursively simplify layout. (Each JPanel a different layout).
- Third Party Layout Managers for **very complex layouts**
  - MiGLayout
  - Karsten Lentzsch's FormLayout

Conclusion

- A starting point for Javax Swing
- Now practise
- More on (reference and examples) on http://www.javasoft.com