Java Swing
Eléments de Base

ESIL

Laurent Henocque

Cours mis à jour en Février 2008

Licence Creative Commons

Introduction

- Swing est la bibliothèque Java performante pour la réalisation des interfaces homme machine.
- Swing est actuellement en compétition avec SWT (Eclipse) et bientôt Flash et Javascript (adobe flex et air)
- http://java.sun.com/docs/books/tutorial/uiswing/index.html

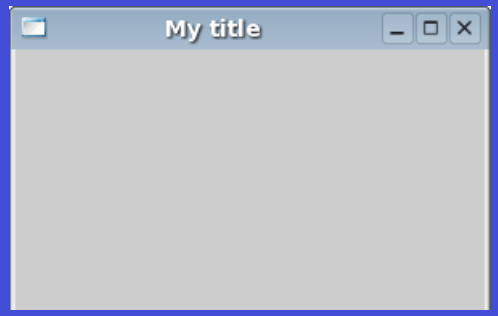
Présentation très rapide des concepts

Création d'une classe "frame"

```
import javax.swing.JFrame;

  public class MainFrame extends
  JFrame
  {
     public MainFrame()
     {
        super("My title");
        setSize(300, 300);
     }
  }
```

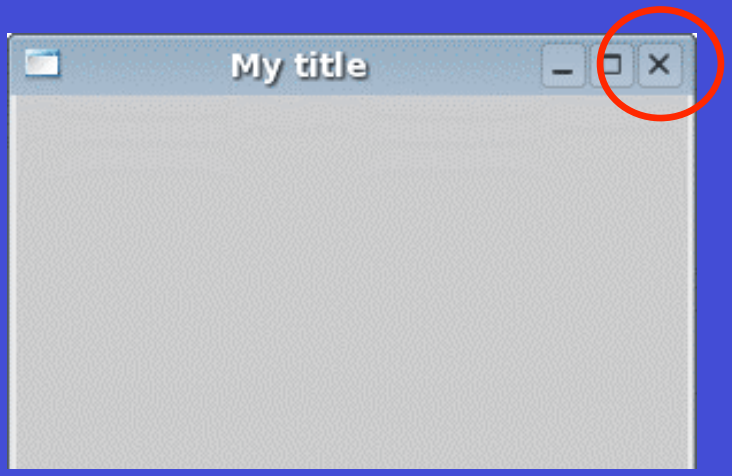
Création d'une application

```
public class Application
{
    public static void main(String[] args)
    {
        // perform any initialization
        MainFrame mf = new MainFrame();
        mf.show();
    }
}
```



Fermer l'application

```
setDefaultCloseOperation(
    JFrame.EXIT_ON_CLOSE);
```

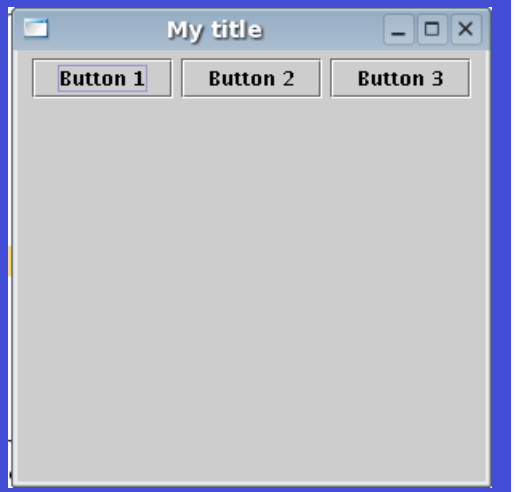


Ajout de composants

```
Container content = getContentPane();
content.add(new JButton("Button 1"));
```

Grouper des composants

```
JPanel panel=new JPanel();
panel.add(new JButton("Button 1"));
panel.add(new JButton("Button 2"));
panel.add(new JButton("Button 3"));
content.add(panel);
```

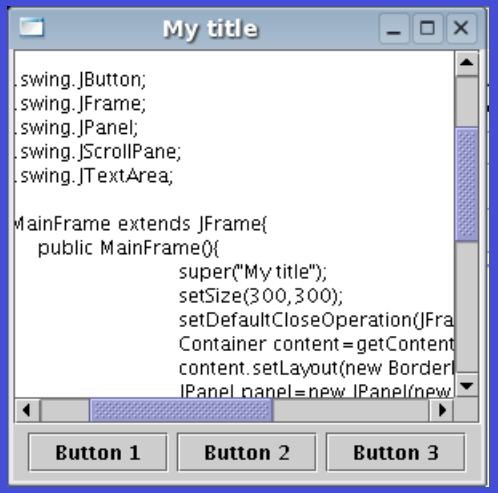


Gérer le Positionnement Variable


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

public class MainFrame extends JFrame{
   public MainFrame() {
      super("My title");
      setSize(300,300);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      Container content = getContentPane();
      content.setLayout(new BorderLayout());
      JPanel panel = new JPanel(new FlowLayout());
      panel.add(new JButton("Button 1"));
      panel.add(new JButton("Button 2"));
      panel.add(new JButton("Button 3"));
      content.add(panel, BorderLayout.SOUTH);
      content.add(new JTextArea(), BorderLayout.CENTER);
 }}
```

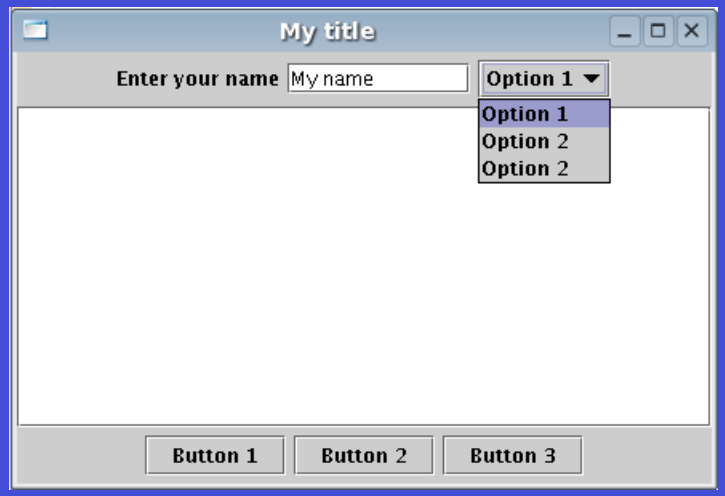
Ascenseurs

content.add(new JScrollPane(new JTextArea()), BorderLayout.CENTER);



Mixer les layouts

```
panel=new JPanel(new FlowLayout());
panel.add(new JLabel("Enter your name"));
panel.add(new JTextField(10));
String options[] = new String[]{ "Option 1","Option 2","Option 2" };
panel.add(new JComboBox(options));
content.add(panel, BorderLayout.NORTH);
```



Evénements

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

public class MainFrame extends JFrame {
  public MainFrame(){
    super("My title");
...
    JButton button1 = new JButton("Button 1");
    panel.add(button1);
    button1.addActionListener( new MyButtonListener(this));
...  }

  private class MyButtonListener implements ActionListener
  {
    private JFrame parentComponent;
    MyButtonListener(JFrame parentComponent){
      this.parentComponent=parentComponent; }

    public void actionPerformed(ActionEvent e) {
    JOptionPane.showMessageDialog(parentComponent, "BUTTON PRESSED!");}
  }
}
```

Evénements (version 2)


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

public class MainFrame extends JFrame {
   public MainFrame(){
      super("My title");
...
      JButton button1 = new JButton("Button 1");
      panel.add(button1);
      button1.addActionListener( new ActionListener() {
            public void actionPerformed(ActionEvent e) {
            JOptionPane.showMessageDialog(
                        e.getSource().parentComponent, "BUTTON PRESSED!");}
            });
...    }
}
```

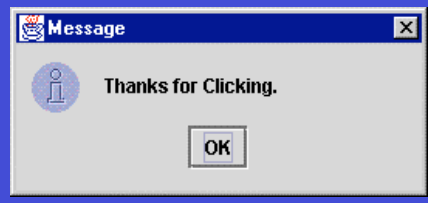
Présentation de la Bibliothèque

Squelette d'application Swing

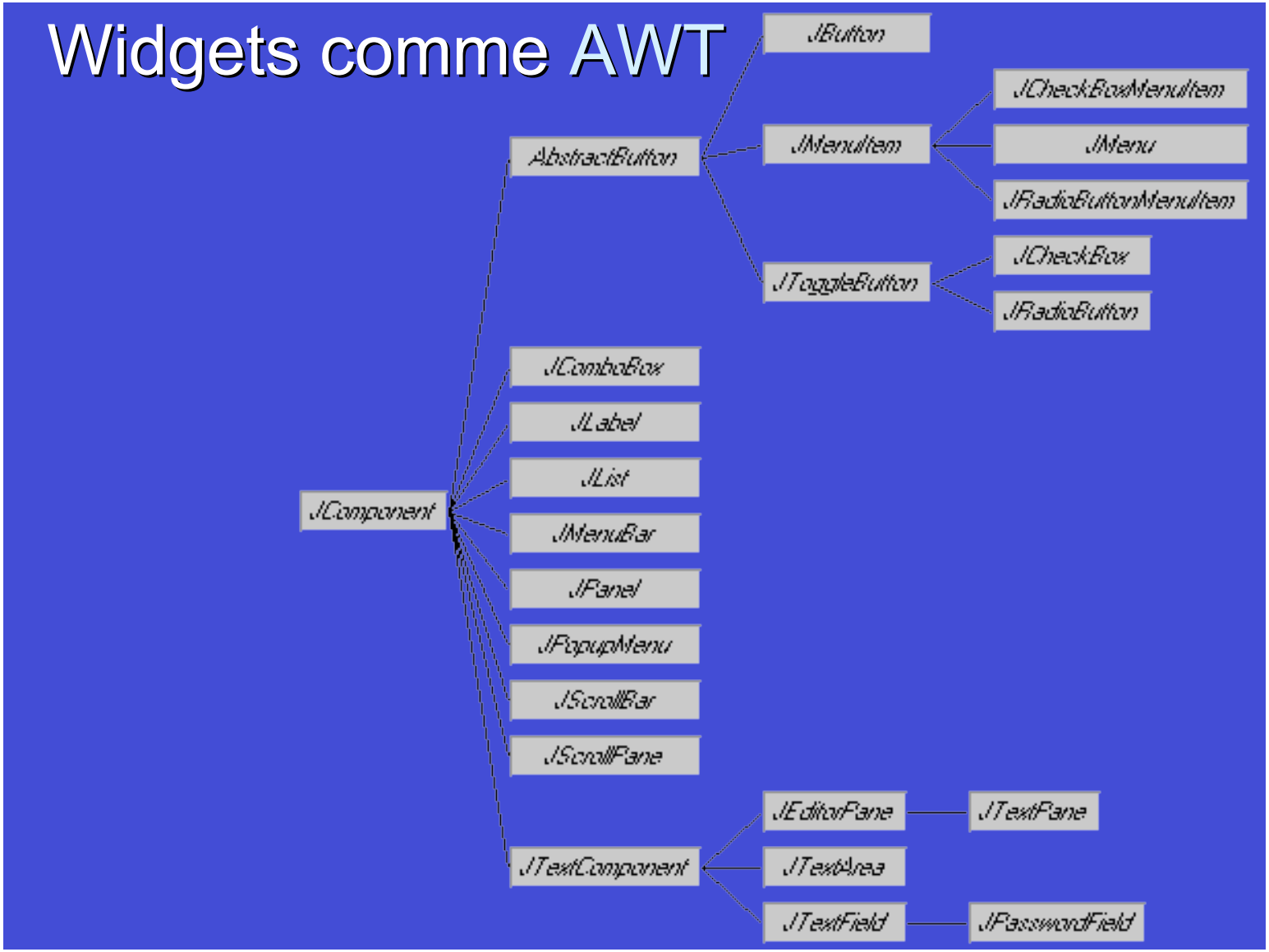
```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
public class JWinApp extends JFrame{
    public JWinApp(String title, JPanel panel){
        super(title);
        getContentPane().add(panel, BorderLayout.CENTER);
        setSize(200,200);
        addWindowListener(new WindowAdapter(){
                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

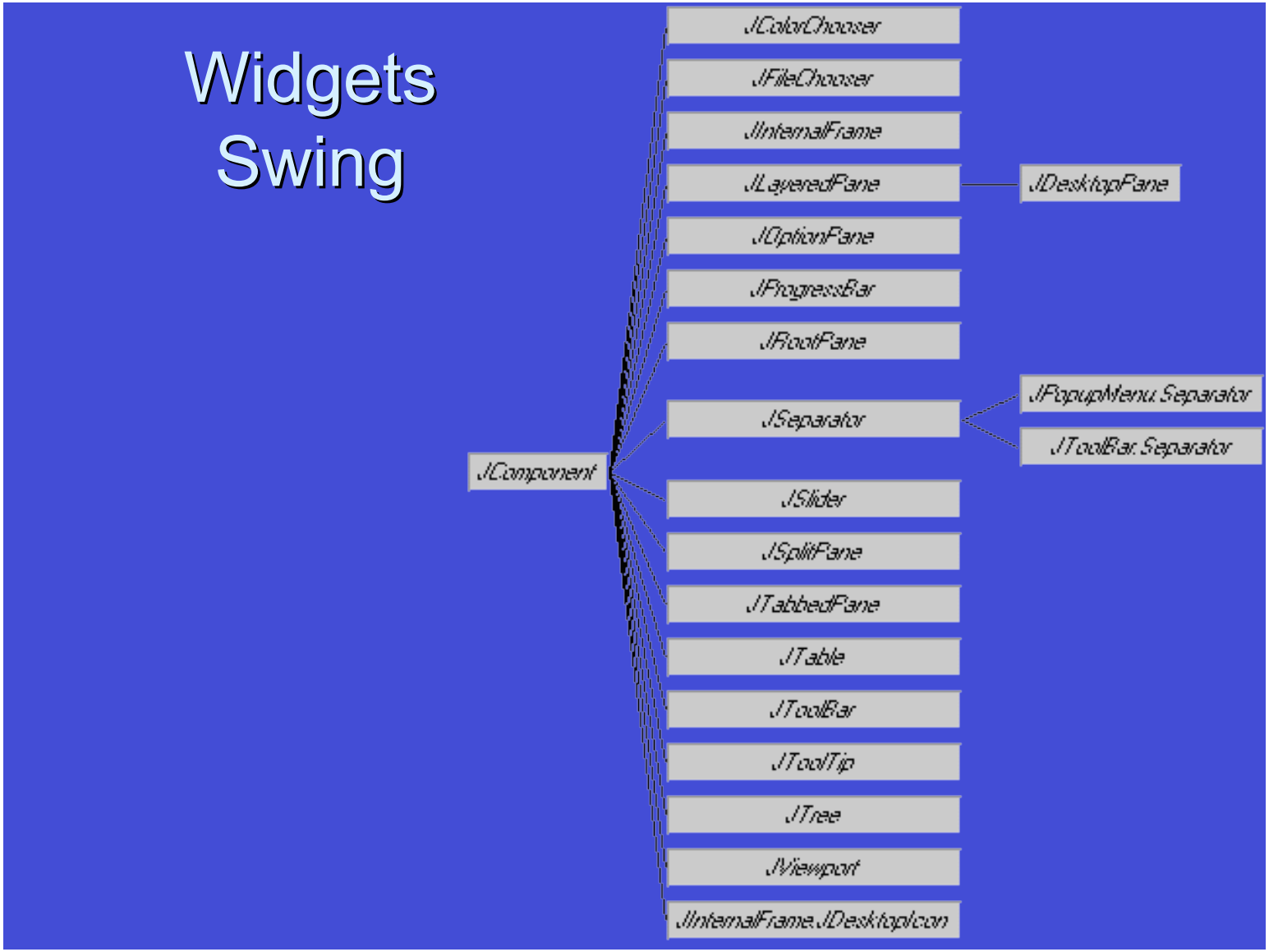

```
class WinHelloPanel extends JPanel implements
    ActionListener{
    JLabel label = new JLabel("Hello World "); // un label
    JButton button = new JButton("Click!"); // un bouton
    public WinHelloPanel(){
          add(label);
          add(button);
          button.addActionListener(this);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



Widgets comme AWT



Widgets Swing



JPanel

- C'est un Panel léger offrant un support pour le double buffering (technique d'affichage en deux temps permettant d'éviter les scintillements et défaut d'aspects)

- Quand le buffering est activé (constructeur) tous les composants se dessinent d'abord dans un buffer non affiché

Icones

- Les icones sont utilisées avec tous les boutons ou autres composants.

```
public interface Icon {
  void paintIcon(Component c, Graphics g, int x, int y);
  int getIconWidth();
  int getIconHeight();
}
```

- l'argument "c" sert à fournir une information complémentaire au moment du dessin (police, couleur)
- x et y spécifient l'origine du dessin

ImageIcon

```
Icon i = new ImageIcon(“Image.gif");
Icon j = new ImageIcon(
   new URL(“http://…Image.gif”));
```

- Avantages :
  - url ou fichier,
  - chargement asynchrone : pas de blocage de l’interface
  - l’image n’est pas sérializable

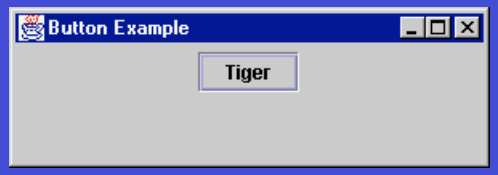
Créer sa propre icone

```
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());
  }
  public int getIconWidth() {    return 10;  }
  public int getIconHeight() {     return 10;  }
}
```

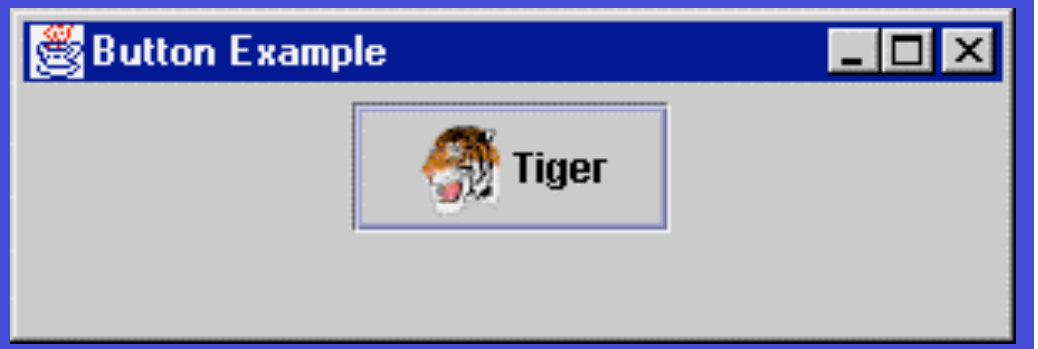
JLabel

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

    JLabel fancyLabel = new JLabel(« Super Beau 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

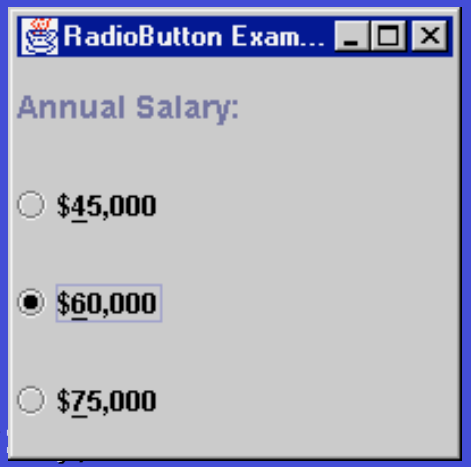
- Plusieurs classes Swing implémentent abstractButton
- setMnemonic() – accélérateur clavier : les constantes VK_* de KeyEvent
- doClick() – appel du bouton par programme
- setDisabledIcon(), setDisabledSelectedIcon(), setPressedIcon(), setRolloverIcon(), setRolloverSelectedIcon(), setSelectedIcon() – modifications dynamique de l'icone
- setVerticalAlignment(), setHorizontalAlignemnt()
- setVerticalTextPosition(), setHorizontalTextPosition() – place le texte par rapport à l'icone
- On peut spécifier le texte du label en html (<html> … <html>) à partir de swing 1.1.1)

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);
  }
```

```
class ToggleIcon implements Icon {
    boolean state;
    public ToggleIcon (boolean s) {
      state = s;
    }

    public void paintIcon (Component c,
     Graphics g, int x, int y) {

      int width = getIconWidth();
      int height = getIconHeight();
      g.setColor (Color.black);
      if (state) g.fillRect (x, y, width,
     height);
      else  g.drawRect (x, y, width,
     height);
    }

    public int getIconWidth() { return
     10;  }
    public int getIconHeight() { return
     10; }
  }
}
```

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"));
  }
}
```

Méthodes de JTextComponent

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

JTextField & JTextArea

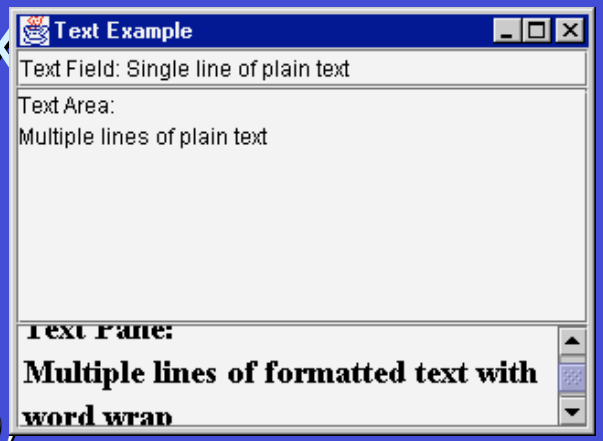
```
JTextField tf = new JTextField();

JTextArea ta = new JTextArea();

tf.setText("TextField");

ta.setText("JTextArea\n Multi Lignes");

add(tf);

add(new JScrollPane(ta)); // scroll au cas où
```

JTextPane

JTextPane est un éditeur de texte complet (avec insertions d'images). Il s'appuie sur une liste de blocs dotés de styles

- 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));

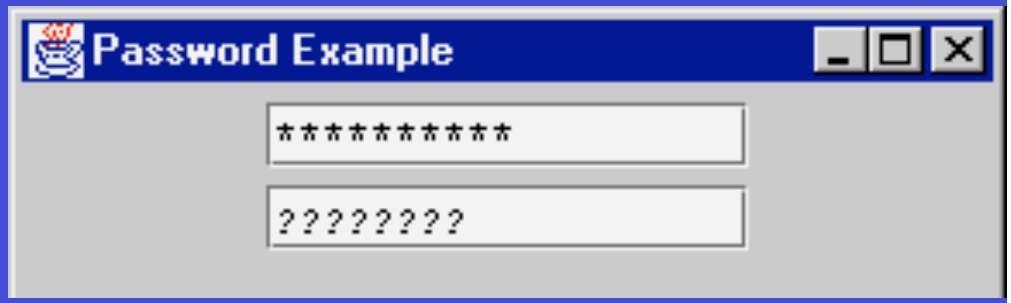
JTextPane ex




```
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 est un éditeur de textes
- permettant l'affichage de contenu html, ou rtf, identifié par une URL,
- et permettant de suivre les liens

Source

```
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);

    // suivre les liens :
    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);}}});
}}});
```

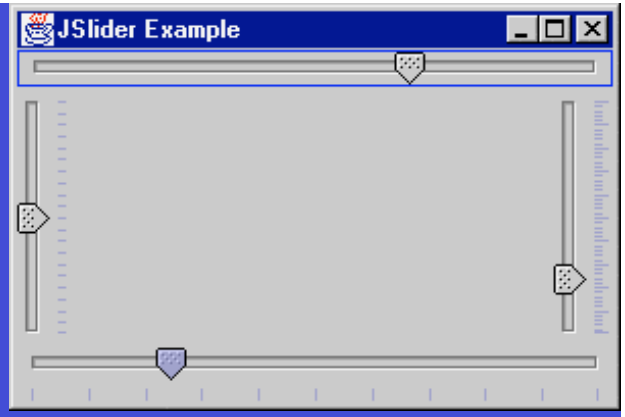
suite

```
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);
  }
}
```

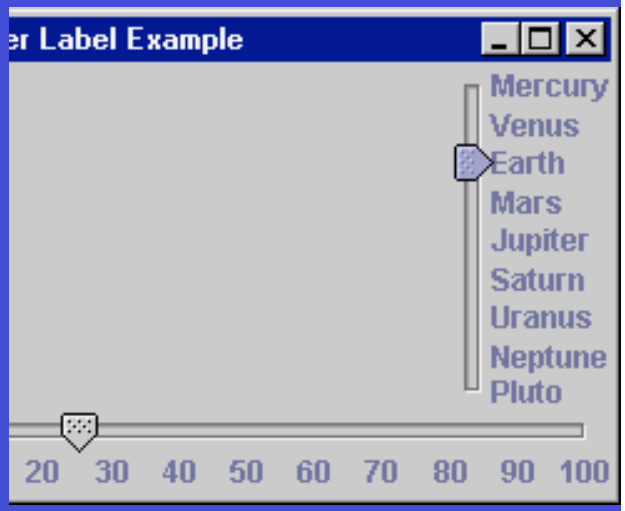
JSlider



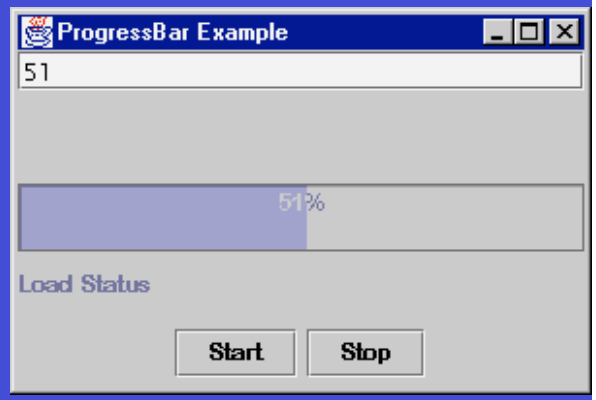

```
public class SliderPanel extends JPanel {
  public SliderPanel() {
    setLayout(new BorderLayout());

    JSlider s1 = new JSlider (JSlider.VERTICAL,
    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 et 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);
  }
}
```



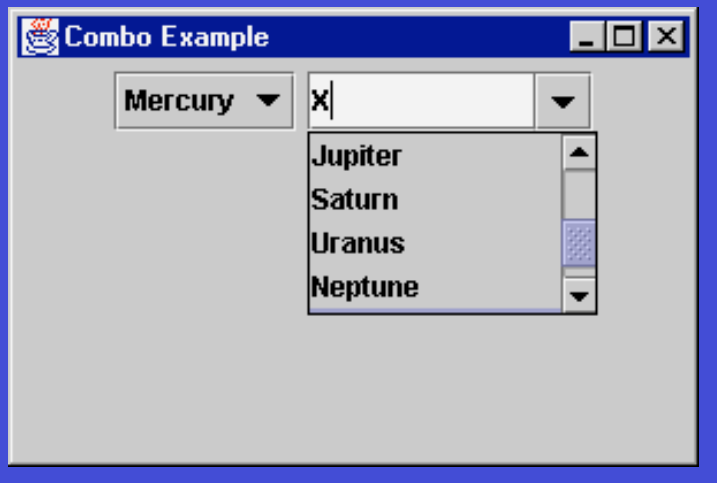
JProgressBar



- Créer une Progress Bar
  - progressBar = new JProgressBar(0, task.getLengthOfTask());
  - progressBar.setValue(0);
  - progressBar.setStringPainted(true);
- Changer la valeur courante :
  - progressBar.setValue(task.getCurrent());
- Utilisation du mode « indeterminate »
  - progressBar = new JProgressBar();
  - progressBar.setIndeterminate(true);
  - *... // la taille devient connue*
  - 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[]) {
…   } }
```

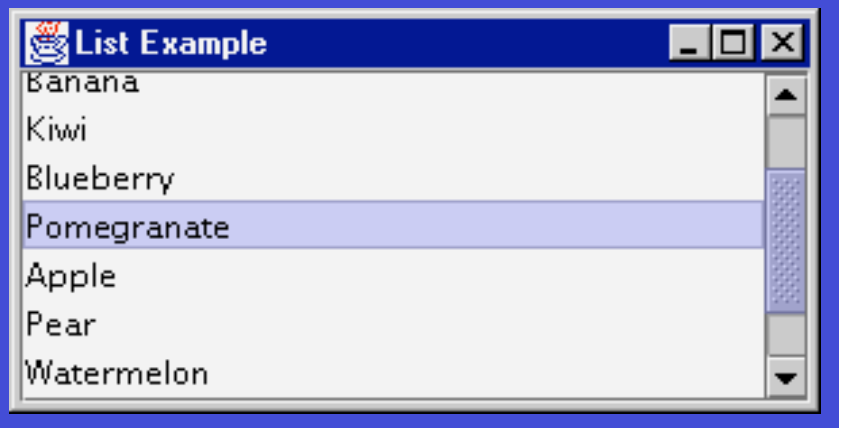


Callbacks ComboBox

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

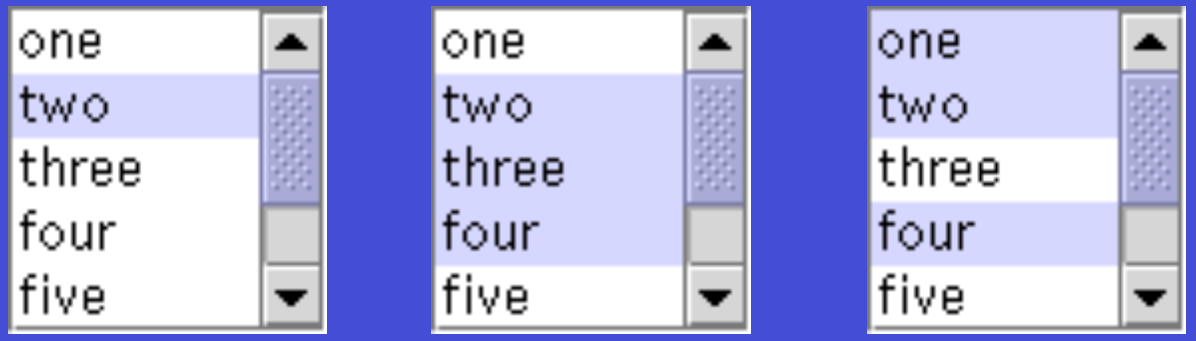
JList

```
String label [] = {"a", "b", "c", "d", "e",
   "f", "g", "h", "i", "j", "k" };
JList list = new JList(label);
JScrollPane pane = new JScrollPane(list);
```



JList : Sélection

```
static Vector v;
l = new JList(v);
l.setSelectionMode(
    ListSelectionModel.SINGLE_SELECTION);
    // SINGLE_INTERVAL_SELECTION
    // MULTIPLE_INTERVAL_SELECTION
```



Modification dynamique de JList

```
//Schéma Modèle/Vue(s)/Controleur
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

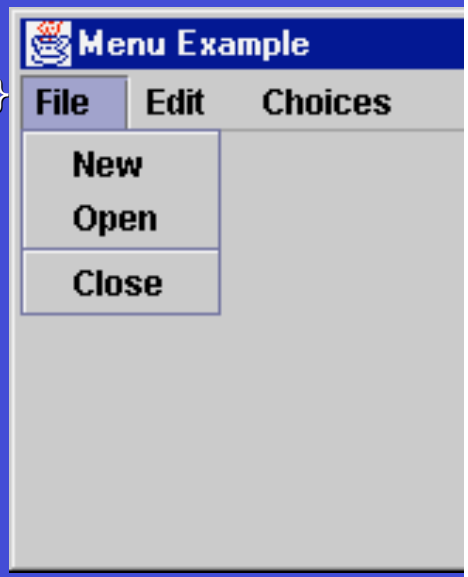
```
JButton b = new JButton("Empty"); b.setBorder (new EmptyBorder (1,1,1,1));

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.setBorder(new
    BevelBorder(BevelBorder.RAISED));

b = new JButton ("Bevel Down");  b.setBorder(new
    BevelBorder(BevelBorder.LOWERED));
```

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 sur menu items

```
menuItem.addActionListener(this);
...
//JRadioButtonMenuItem:
rbMenuItem.addActionListener(this);
...
//JCheckBoxMenuItem:
cbMenuItem.addItemListener(this);
```

Sous menus

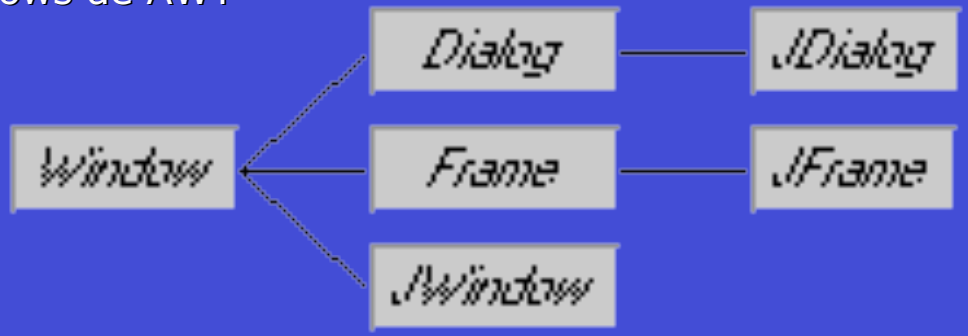
```
//submenu
submenu = new JMenu("A submenu");
submenu.setMnemonic(KeyEvent.VK_S);

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());
       } } });  }
```

Hiérarchie des Fenêtres

- La hiérarchie des classes fenêtre Swing s'intègre sous la classe Windows de AWT



- Elles ne sont donc pas "lightweight", sont associées à une fenêtre graphique, et ne peuvent pas être transparentes.
- On peut utiliser setJMenuBar().
- De même que pour JWindow et JDialog, on doit ajouter les éléments à un container obtenu par getContentPane()

JFrame et Close

- Contrairement à Frame, JFrame se ferme sur "close".
- setDefaultCloseOperation() :
  - DO_NOTHING_ON_CLOSE: comme AWT
  - HIDE_ON_CLOSE: le défaut (setVisible(true) remappe la fenêtre)
  - DISPOSE_ON_CLOSE: récupère les ressources
- HIDE_ON_CLOSE et DISPOSE_ON_CLOSE laissent s'exécuter les event listeners

JRootPane

- Le dispositif d'affichage d'un JFrame est un JRootPane, composé de deux objets : un glass pane et un layered pane.
- Le glass pane est invisible, mais toujours devant le layered pane, et permet l'affichage des tooltips et des popups
- Le layered pane est constitué d'un menubar optionnel, et d'un content pane, utilisé habituellement

JLayeredPane

- Le JLayeredPane permet d'afficher des composants dans des couches différentes, ce qui permet des superpositions:
- layeredPane.add (component, new Integer(5));
- Le défaut est JLayeredPane.DEFAULT_LAYER.
- On peut placer des objets relativement à cette couche, devant ou derrière
- Le LayoutManager détermine l'ordre d'affichage, et empêche les superpositions au sein d'une même couche.

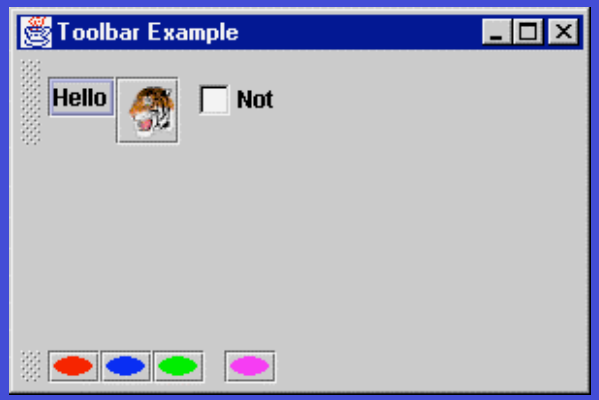
Tooltips

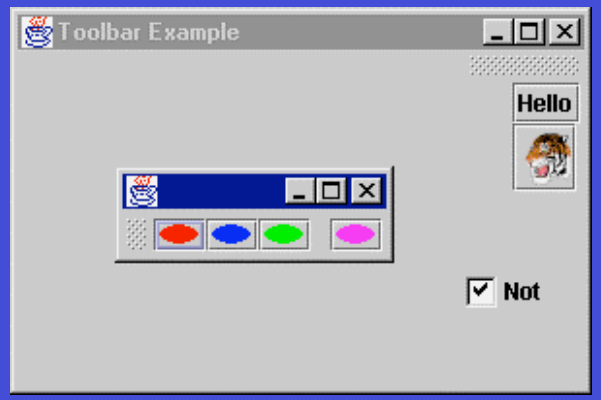
```
public class TooltipPanel extends JPanel {

  public TooltipPanel() {

    JButton myButton = new JButton("Hello");

    myButton.setToolTipText ("World");

    add(myButton);
  }
}
```

Toolbars

- JToolbar est un container qui permet d'afficher des toolbars déplaçables, éventuellement dans d'autres containers que celui d'origine. L'affichage du toolbar passe de vertical à horizontal suivant son emplacement.
- On peut désactiver la possibilité de rendre les toolbars flottantes.

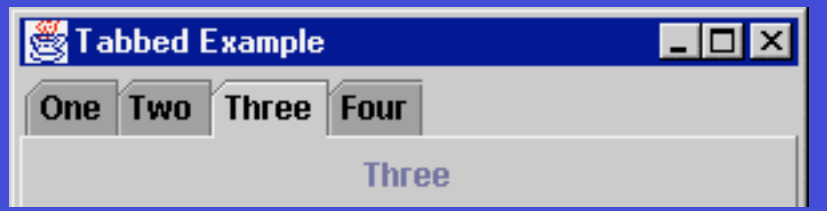
aToolBar.setFloatable (false);





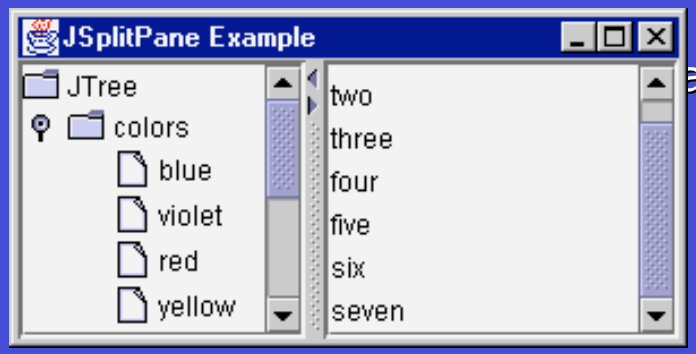
JTabbedPane

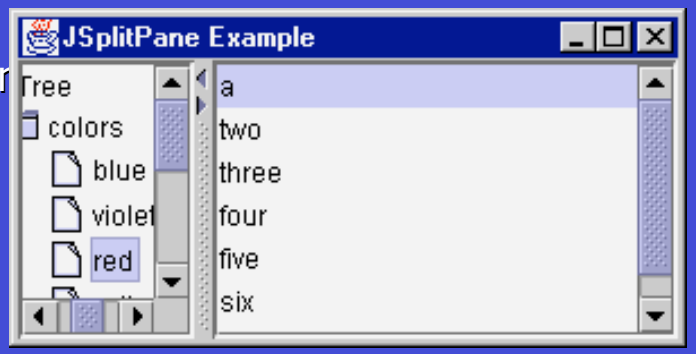
- JTabbedPane permet de réaliser des interfaces à onglets.
- On ajoute les onglets (des "cards") avec addTab(). Une des versions permet l'affichage d'un tooltip
- N'importe quel Component peut être affiché dans un onglet
  - addTab(String title, Component component)
  - addTab(String title, Icon icon, Component component)
  - addTab(String title, Icon icon, Component component, String tip)



JSplitPane

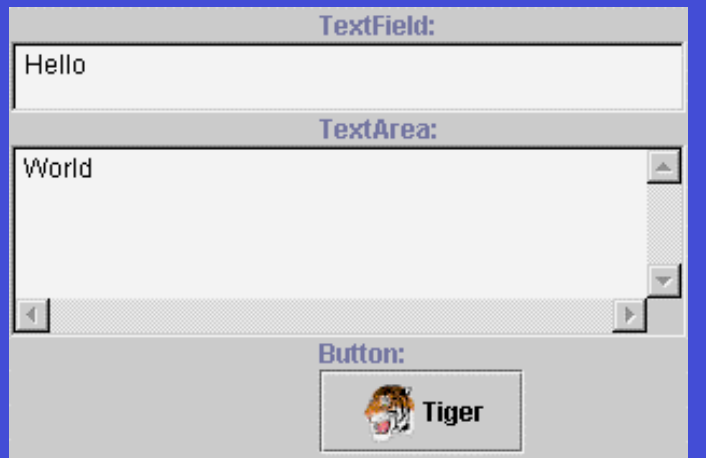
- JSplitPane permet le redimensionnement réciproque de deux fenêtres
- On peut placer un JSplitPane dans un autre : c'est un moyen de faire des interfaces compliquées sans gérer les layouts
- setContinuousLayout permet de voir le redimensionnement en direct





BoxLayout

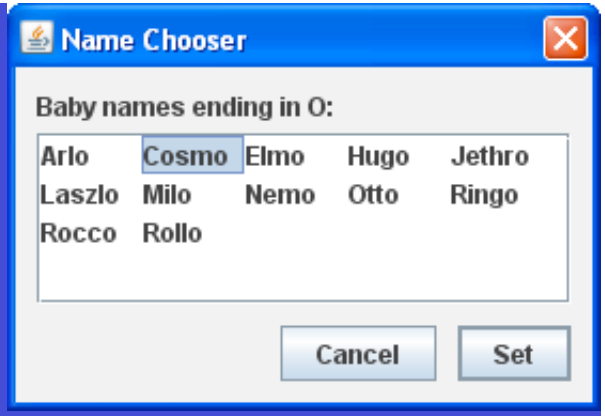
- Le BoxLayout layout arrange les composants selon l'axe horizontal ou vertical, mais plus intelligemment que le grid layout : les épaisseurs ou largeurs peuvent varier
- Il centre les composants ne pouvant pas être redimensionnés
- setLayout(new BoxLayout(this, BoxLayout.Y_AXIS));
- Le premier paramètre spécifie le container (JPanel par ex), et le second l'axe du BoxLayout. On ajoute les composants comme d'habitude
- add(myComponent);



BoxLayout

- Par ailleurs, BoxLayout permet de lier relativement des composants consécutifs :
- En les « collant »,
- Et en introduisant des séparateurs invisibles

BoxLayout




```
//Alignement des boutons
JPanel buttonPane = new JPanel();
buttonPane.setLayout(new BoxLayout(buttonPane,
   BoxLayout.LINE_AXIS));
buttonPane.setBorder(BorderFactory.createEmptyBorder
   (0, 10, 10, 10));
buttonPane.add(Box.createHorizontalGlue());
buttonPane.add(cancelButton);
buttonPane.add(Box.createRigidArea(new Dimension(10,
   0)));
buttonPane.add(setButton);
```

Nouveautés majeures de Java 5

- cf
  http://java.sun.com/j2se/1.5.0/docs/guide/swing/1.5/index.html
- look and feel par défaut amélioré
- introduction d'un nouveau look and feel skinnable sans changer le code: "synth"
- gestion directe des popup menus (avec héritage)
- support de l'impression sur JTable

Nouveautés de Java5 - Texte

- JTextArea ne scrolle plus automatiquement quand on ajoute du texte
- HTMLEditorKit a été amélioré pour intercepter les évènements des formulaires avant que les paramètres "post" ne soient transmis
- Interrogation du caractère "actif" du curseur ("visible" ne convient pas s'il clignote...)
- Gestion de la couleur de fond améliorée dans LabelView

Conclusion

- C'était un point de départ pour Javax Swing
- De la pratique maintenant.
- Doc en ligne (référence et exemples) sur http://www.javasoft.com