Interfaces de Programación e Interfaces Gráficas

Patricio.Inostroza@dcc.uchile.cl

1

Problema

- Años atrás...
- Un zapatero medía el pies a cada cliente y fabricaba el zapato
- Problema:
  - Las medidas de uno no sirven a otro
  - Imposible llevar la fabricación a escala industrial

2

Problema

- Hoy...
- Juan sale a comprar zapatos
- A priori, no conoce cuál será el modelo que comprará
- Pero:
  - Él cliente (Juan) sabe que será N^{ro} 41
  - El vendedor fabrica zapatos N^{ro} 41
- ¿Por qué?
- ¿Cómo se llegó a este acuerdo?

3

La solución

- Los fabricantes de zapato han definido un estándar de medidas para que el cliente pueda elegir el modelo
- El zapatero debe conocer este estándar
- El cliente debe conocer este estándar
- Este estándar permite que el zapatero y el cliente “conversen”
- El zapatero no conoce al cliente
- El cliente no conoce al zapatero

4

Interfaz

- Interfaz:
  - Medio que permite la comunicación de dos o más usuarios
- En computación existen las interfaces de programación
  - Permiten la comunicación de dos o más objetos sin que estos se conozcan

Interfaz de programación

- Un interfaz de programación es una clase donde los métodos que NO tienen implementación
  - No tiene constructor
  - No se puede instanciar (hacer “new”)
- ¿Par qué sirven?

Problema

- Varios clientes van a comprar.
- El vendedor les informa que el producto no ha llegado.
- Los clientes le piden al vendedor que les "avise" cuando el producto este disponible

Solución 1

- Modificar el código por cada cliente

```
class Vendedor {
    Producto p;
  // ...
    public void nuevoStock(int s) {
        if (s > 0)
            avisar();
    }

   public void avisar() {
      pedro.llego(p);
   }
}
```

```
class Vendedor {
    Producto p;
  // ...
    public void nuevoStock(int s) {
        if (s > 0)
            avisar();
    }

   public void avisar() {
        pedro.llego(p);
        juan.nuevoProducto(p);
   }
}
```

Problemas de la solución

```
class Vendedor {
    Producto p;
  // ...
    public void nuevoStock(int s) {
        if (s > 0)
            avisar();
    }

  public void avisar() {
    pedro.llego(p);
  }
}
```

```
    public void avisar() {
        pedro.llego(p);
        juan.nuevoProducto(p);
    }
}
```

- Es necesario conocer el código del vendedor
- Es necesario conocer el código del cliente
- Hay que modificar el código del vendedor
- Cada cliente tiene un método distinto
- No es fácilmente extensible

Solución 2: con Interfaz

- Definir una interfaz para la comunicación
- Cada cliente será llamado a través de esta interfaz
- El cliente deberá informar al Vendedor que está a la escucha del evento
- El cliente deberá reaccionar frente a la llamada

```
interface Listener {

    public void llego(Producto p);

}
```

1: El cliente se inscribe
2: El vendedor informa del suceso

```
interface Listener {

    public void llego(Producto p);

}
```

```
class Cliente implements Listener{
    // . . .
    public void algunMetodo( ) {
        // . . .
        vendedor.inscribir(this);
        // ..
    }

    public void llego( Producto p) {
        // este método será llamado
        // por el vendedor cuando
        // reciba el producto
        // . . .
    }

   // ... resto de los métodos
}
```

```
class Vendedor {
    Producto p;
    ListaDeListener lista;
    // ...
    public void inscribir(Listener l) {
        // Almacenar a cada Listener
        // “l” en la lista
    }

    public void avisar() {
        // para todos que están en la lista
        // de Listener, llamar el método
        // “llego( . . .)”
    }
    // . . .
}
```

1

2

Swing

Una mejora respecto a AWT

Swing

- Sun Microsystems , Netscape Communications, IBM y Lighthouse Design crean una nueva Graphical User Interface (GUI)
- Integrada en JDK 1.1.5+
- Es un estándar en Java 2
- Posee un mejor look and feel que AWT.
- El conjunto de API´s que lo componen se denomina Java Foundation Classes (JFC)

Swing

- JFC compuesto de 5 APIs:
  - AWT,
  - Java 2D,
  - Accessibility,
  - Drag and Drop,
  - Swing
- Las componentes de AWT son las que se encuentran en JDK versión 1.1.2 y posteriores.

Swing

- Paquetes:
  - javax.swing
  - javax.swing.border
  - javax.swing.colorchooser
  - javax.swing.event
  - javax.swing.filechooser
  - javax.swing.plaf.*
  - javax.swing.table
  - javax.swing.text
  - javax.swing.text.html.*
  - javax.swing.text.rtf
  - javax.swing.tree
  - javax.swing.undo
  - javax.accessibility

Swing

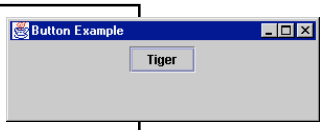
- JPanel
- JLabels
- JButton
- JCheckBox
- JRadioButton
- JScrollPane
- JPasswordField
- JEditorPane
- JSlider
- JProgressBar
- JComboBox
- JList
- Borbers
- Menus
- JSeparator
- JPopupMenu
- JFrame
- Toolbars
- JTabbedPane
- JSplitPane
- Swing Layouts
- ScrollPaneLayout
- ViewportLayout

JPanel / JLabel

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

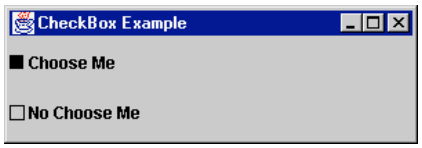
JButton

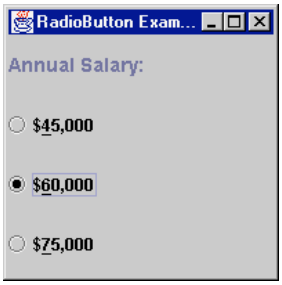
```
public class ButtonPanel extends JPanel {
  public ButtonPanel () {
    JButton myButton = new JButton("Tiger");
    add(myButton);
  }
}
```




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

JCheckBox / JRadioButton



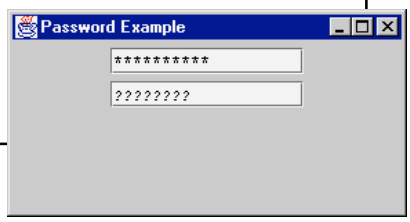


JScrollPane

```
public class ScrollPanel extends JPanel {

  public ScrollPanel() {
    setLayout(new BorderLayout());
    Icon bigTiger = new ImageIcon("BigTiger.gif");
    JLabel tigerLabel = new JLabel(bigTiger);
    JScrollPane scrollPane =
      new JScrollPane(tigerLabel);
    add(scrollPane, BorderLayout.CENTER);
  }
}
```

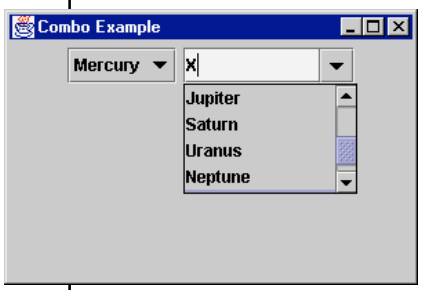
JPasswordField

```
class PasswordPanel extends JPanel {
  PasswordPanel() {
    JPasswordField pass1 = new JPasswordField(20);
    JPasswordField pass2 = new JPasswordField(20);
    pass2.setEchoChar ('?');
    add(pass1);
    add(pass2);
  }
}
```



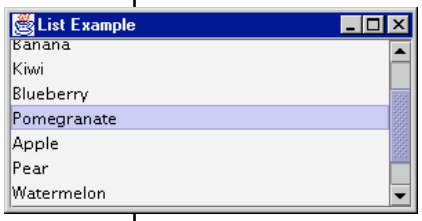
JComboBox

```
public class ComboPanel extends JPanel {
  String choices[] = {
    "Mercury", "Venus", "Earth", "Mars",
    "Jupiter", "Saturn", "Uranus","Neptune", "Pluto"};

  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);
  }
}
```

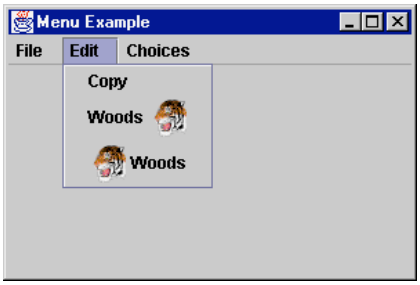


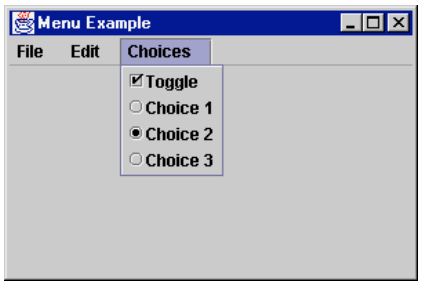
JList

```
public class ListPanel extends JPanel {
  String label [] = {"Cranberry", "Orange",
    "Banana", "Kiwi", "Blueberry",
    "Pomegranate", "Apple", "Pear",
    "Watermelon", "Raspberry", "Snozberry"
  };

  public ListPanel() {
    setLayout (new BorderLayout());
    JList list = new JList(label);
    JScrollPane pane = new JScrollPane(list);
    add(pane, BorderLayout.CENTER);
  }
}
```

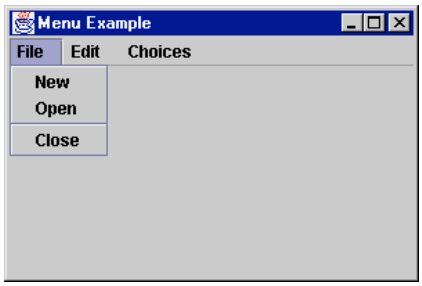


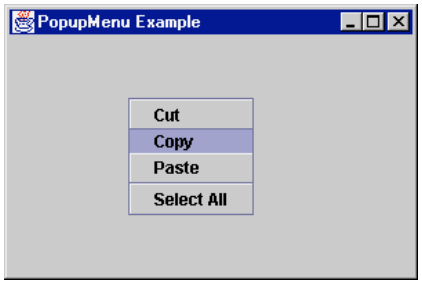
JMenuBar / Jmenu / JMenuItem





JSeparator / JPopupMenu





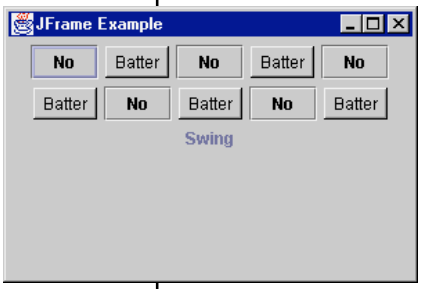
JSeparator / JPopupMenu

```
public class PopupPanel extends JPanel {
  JPopupMenu popup = new JPopupMenu ();
  public PopupPanel() {
    JMenuItem item;
    popup.add (item = new JMenuItem ("Cut"));
    popup.add (item = new JMenuItem ("Copy"));
    popup.add (item = new JMenuItem ("Paste"));
    popup.addSeparator();
    popup.add (item = new JMenuItem ("Select All"));
    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());
        }
      }
    });
  }
}
```

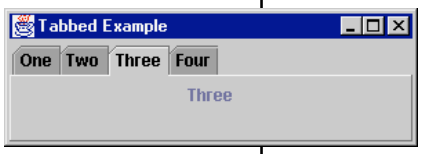

JFrame

```
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"));
    f.setSize (300, 200);
    f.show();
  }
}
```

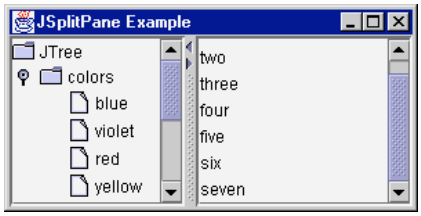


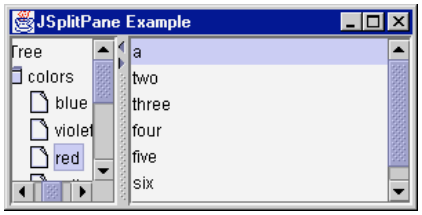
JTabbedPane

```
public class TabbedPanel extends JPanel {
  String tabs[] = {"One", "Two", "Three", "Four"};
  public JTabbedPane tabbedPane = new JTabbedPane();
  public TabbedPanel() {
    setLayout (new BorderLayout());
    for (int i=0;i<tabs.length;i++)
      tabbedPane.addTab (tabs[i], null,
                         createPane (tabs[i]));
    tabbedPane.setSelectedIndex(0);
    add (tabbedPane, BorderLayout.CENTER);
  }

  JPanel createPane(String s) {
    JPanel p = new JPanel();
    p.add(new JLabel(s));
    return p;
  }
}
```



JSplitPane





Swing

- Ejemplos tomados de:

  http://developer.java.sun.com/developer/onlineTraining/GUI/Swing1/shortcourse.html

Relación entre las interfaces

- Interfaces de Programación
  - Interface + Implemments
- Interfaces Gráficas
  - Awt + Swing
- ¿Dónde está la relación?

Ejemplo

- Una aplicación crea una ventana un panel
- El panel contiene un botón
- Una clase escucha los click sobre el botón
- La aplicación escucha lo que le pasa a la ventana

o Una aplicación crea una ventana con un boton
o Una clase escucha los click sobre el botón
o La aplicación escucha lo que le pasa a la ventana

```
public class Logo2D implements WindowListener {

    public static void main(String[] args)   {
          Logo2D logo = new Logo2D();
    }

    public Logo2D() {
        createUI();
    }

    private void createUI( )   {
        JFrame frame = new JFrame("Este es un ejemplo");
        Container container = frame.getContentPane();
        container.setLayout (new FlowLayout());

        MyButtonListener bl = new MyButtonListener();
        Button b1 = new JButtonl("Primer boton");
        b1.addButtonListerner( bl);
        container.add(b1);

        frame.addWindowListener(this);
        frame.setSize(300, 200);
        frame.show();
    }
// continua al lado...
```

```
// continuación...
    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
    public void windowActivated(WindowEvent windowEvent) {   }
    public void windowClosed(WindowEvent windowEvent) {    }
    public void windowDeactivated(WindowEvent windowEvent) {    }
    public void windowDeiconified(WindowEvent windowEvent) {    }
    public void windowIconified(WindowEvent windowEvent) {    }
    public void windowOpened(WindowEvent windowEvent) {    }
} // Fin clase Logo2D
```

- Una aplicación crea una ventana con un boton
- **Una clase escucha los click sobre el botón**
- La aplicación escucha lo que le pasa a la ventana

```
public class Logo2D implements WindowListener {

    public static void main(String[] args)   {
          Logo2D logo = new Logo2D();
    }

    public Logo2D() {
        createUI();
    }

    private void createUI( )   {
        JFrame frame = new JFrame("Este es un ejemplo");
        Container container = frame.getContentPane();
        container.setLayout (new FlowLayout());

        MyButtonListener bl = new MyButtonListener();
        Button b1 = new JButtonl("Primer boton");
        b1.addMouseListerner( bl);
        container.add(b1);

        frame.addWindowListener(this);
        frame.setSize(300, 200);
        frame.show();
    }
// continua al lado...
```

```
// continuación...

    public void windowClosing(WindowEvent windowEvent) {
        System.exit(0);
    }
    public void windowActivated(WindowEvent windowEvent) {   }
    public void windowClosed(WindowEvent windowEvent) {    }
    pu                                                                  }
    pu                                                                  }
    pu
    pu
} // F
```

```
public class MyButtonListener implements
MouseListener {

  void mouseClicked(MouseEvent e) {
      System.out.println(e.getX() + " " + e.getY());
  }

  void mouseEntered(MouseEvent e) { }

  void mouseExited(MouseEvent e) { }

  void mousePressed(MouseEvent e) { }

  void mouseReleased(MouseEvent e) { }
}
```


Preguntas