Curso de programación con AWT y Swing

(Versión 5, JDK 1.5.0)

Manual del alumno

Autor: Cédric Simon

Ing. Cedric Simon – Tel: 2268 0974 – Cel: 8888 2387 – Email: cedric@solucionjava.com – Web: www.solucionjava.com

Índice

1 Introducción al curso

1.1 Objetivo de este curso

En este curso vamos a ver cómo diseñar entorno gráfico para aplicaciones desarrolladas en lenguaje Java, utilizando el paquete AWT del Java.

1.2 Manual del alumno

Este manual del alumno es una ayuda para el alumno, para tenga un recuerdo del curso. Este manual contiene un resumen de las materias que se van a estudiar durante el curso, pero el alumno debería de tomar notas personales para completas este manual.

1.3 Ejercicios prácticos

Para captar mejor la teoría, se harán muchos ejercicios con los alumnos, para probar la teoría y verificar la integración de la materia.

También, el alumno podrá copiar sus códigos en un disquete al fin del curso para llevarse, con fin de seguir la práctica en su hogar.

1.4 Requisitos para atender a este curso

Una iniciación al lenguaje Java es requerida para seguir este curso. La creación y el manejo de objetos Java está considerada cómo asimilado antes de empezar este curso.

Si el alumno tiene dificultades en un u otro capitulo, el debe sentirse libre de pedir explicaciones adicionales al profesor.

Pero si aparece que el alumno no posee los requisitos mínimos para este curso, por respeto a los otros alumnos que ya poseen esta materia, el alumno podría ser traslado para otro curso en el futuro, cuando el cumplirá con los requisitos.

2 Introducción al AWT

2.1 Objetivo del capitulo

Al fin de este capitulo, el alumno tendrá una vista general del paquete AWT de Java, así como de sus características generales.

2.2 Que es el AWT

El AWT (Abstract Windows Toolkit) es la parte de Java que se ocupa de construir interfaces gráficas de usuario. Existen también otros modelos de componentes distintos como Swing.

2.3 Las partes de una Interface Gráfica de Usuario

La construcción de interfaces gráficas de usuario en Java se descompone en tres partes:

- El contenedor (container), que es la ventana (o parte de la ventana) donde se situarán los componentes.
- Los componentes, que son los menús, botones, áreas de texto etc...
- El modelo de eventos. Cada acción producida por el usuario (con el ratón o el teclado), produce un evento que puede ser captado por Java, para permitir ejecutar un código en reacción a ese evento, si es necesario. Por defecto, Java no capta ninguno evento sobre ningún objeto. Hay que registrar el evento sobre el objeto para que sea captado por el Java.

3 El contenedor

El contenedor es la ventana que va a contener todos los componentes de la aplicación.

Cuando se crea la ventana, se debe de acompañarla de la creación del evento de cierro de ventana. Si se le olvida, solo podrá cerrar la ventana matando al proceso Java. El evento es un escuchador de ventana, que está atento al clic sobre el botón de cierra de ventana.

La creación de un contenedor de hace creando una nueva clase extendiendo la clase `Frame` del AWT.

Se necesita también importar el paquete `java.awt.*` y el paquete `java.awt.event.*`.

Por defecto, la ventana no es visible, así hay que ponerla visible utilizando el método `setVisible` con parámetro verdadero.

A la ventana, se le puede definir el título que aparecerá en la barra de título.

También se puede definir muchas propiedades cómo su tamaño por defecto, si está maximizada, etc... Para más información sobre los métodos disponible, ver en la documentación de Java.

Ejemplo:

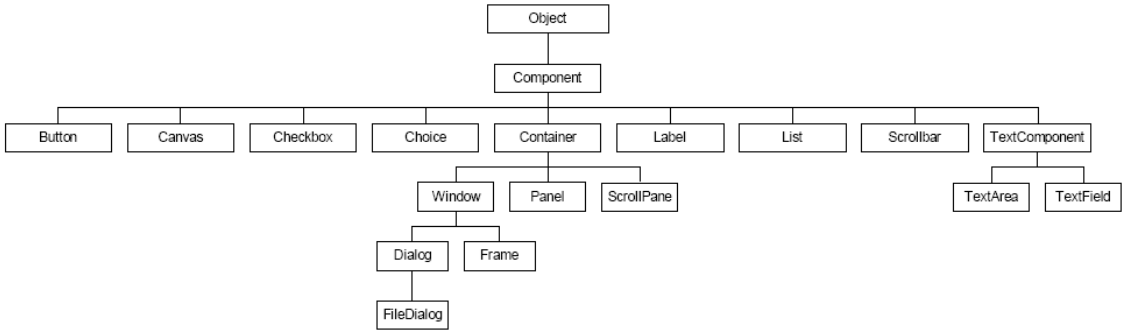
```
import java.awt.*;
import java.awt.event.*;

public class Ventana extends Frame {
    public Ventana() {
        try {
          this.addWindowListener(new WindowAdapter() { // Opens addWindowListener method
                public void windowClosing(WindowEvent e) { // Opens windowClosing method
                    System.exit(0);
                } // Closes windowClosing method
            }); // Closes addWindowListener method
            this.setTitle("Mi primera ventana en Java");
          //  this.setSize(800, 570); // ventana de 800x570 pixeles
          //  this.setExtendedState(this.MAXIMIZED_BOTH); // Maximizar la ventana
            this.setVisible(true);

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
    public static void main(String[] args) {
        new Ventana();
    }
}
```


4 Los componentes

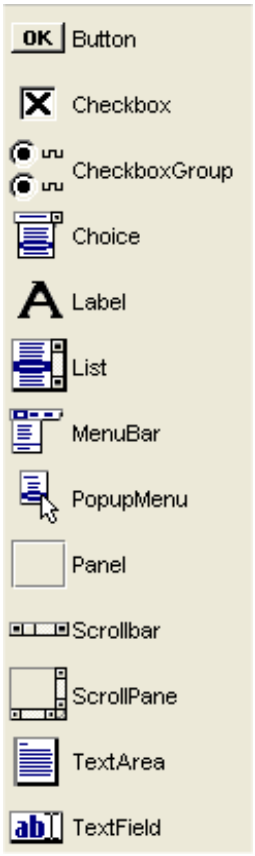
Los componentes son los botones, etiquetas, áreas de textos, etc... que van a llenar la ventana, y permitirnos de enseñar la información y interaccionar con el usuario.

Todos los componentes tienen métodos comunes, herencia de la clase `Component`, cómo por ejemplo el tamaño, el color, la fuente, etc... Pero cada componente tiene también sus propios métodos para sus acciones especificas. Para más información sobre los métodos disponible, ver en la documentación de Java.

Los componentes se pueden incluir en la ventana principal, o en otro componente de tipo Container.



Jerarquía de clase para los componentes AWT



4.1 Etiquetas

La etiquetas (`Label`) son textos que no se pueden cambiar directamente por el usuario (sólo lectura).

4.2 Textos

Existen dos tipos de componentes de texto diferente. Los dos permiten al usuario de modificar su valor. El primero, llamado `TextField`, se usa para contener textos de una sola línea. El secundo, `TextArea`, se usa para textos de varias líneas.

4.3 Botones

Los botones (`Button`) se utilizan para empezar acciones.

4.4 Casillas

Las casillas (`CheckBox`) se utilizan para marcar (o no) opciones. La casilla incluye un texto como una etiqueta.

Cuando se utiliza un grupo de opciones (`CheckBoxGroup`), al contrario de las `CheckBox`, sólo una de las opciones propuestas se puede elegir. La elección de una borra elecciones previas.

4.5 Elecciones

Las elecciones (`Choice`) permite al usuario elegir un valor entre varias dentro de una lista.

4.6 Listas

La lista (`List`) permite al usuario de elegir una o varias valores dentro de una lista.

4.7 Canvas

El `Canvas` crea un cuadro blanco en el cual se puede dibujar objetos gráficos.

4.8 Scrollbar

Un `Scrollbar` permite elegir valores de un rango de valores moviendo el cursor del `ScrollBar`.

4.9 Barra de menú

La barra de menú (`MenuBar`) permite crear una barra de menú con menús de varios niveles.

4.10PopupMenu

Un `PopupMenu` permite sacar un menú de acciones de desde un componente.

4.11Contenedores

4.11.1 Panel

Es el contenedor más simple. Provee un espacio a donde se pueden poner otros componentes.

4.11.2 ScrollPane

Es un contenedor parecido al Panel, si no que tiene `ScrollBar` para moverse horizontalmente y verticalmente en el en caso que su tamaño es mas grande que el de la ventana.

4.12Diseño del código

Para facilitar la lectura del código, es mejor de dividir el código en varias partes.

No existe una manera estándar de dividir el código, y cada entorno gráfico de desarrollo tiene su propio formato.

Si no utilizan un entorno de diseño gráfico de GUI, les propongo el formato siguiente:

- declarar los componentes a nivel de la clase
- crear a lo menos dos métodos: una para inicializar los componentes, y otro para los eventos
- si hay varios paneles, se puede crear un método para manejar los `layouts`.
- utilizar el constructor para llamar a los métodos creadas arriba, y crear la ventana.

Ejemplo:

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

public class VentanaCompleta extends Frame {
        // Mis componentes de clase
        Panel miPanel1 = new Panel();
        Label miEtiqueta1 = new Label();
        TextField miTexto1 = new TextField();
        TextArea miTexto2 = new TextArea();
        Button miBotón1 = new Button();
        Checkbox miCasilla1 = new Checkbox();
        Checkbox miCasilla2 = new Checkbox();
```



```
CheckboxGroup miGrupo1 = new CheckboxGroup();
Checkbox miCasilla3 = new Checkbox();
Checkbox miCasilla4 = new Checkbox();
Choice miElección1 = new Choice();
List miLista1 = new List();
Scrollbar miScrollBar1 =   new Scrollbar();
ScrollPane miPanel2 = new ScrollPane();
MenuBar miMenu = new MenuBar();
Menu menuArchivos = new Menu("Archivos");
MenuItem guardar = new MenuItem("Guardar");
MenuItem salir = new MenuItem("Salir");

Ventana () { // constructor
        initComponentes();
        initEvents();
        testTextArea();
        this.add(miPanel1);
        this.setMenuBar(miMenu);
        this.setSize(500,600);
        this.setVisible(true);
}

void initComponentes (){
        miEtiqueta1.setText("Mi primera etiqueta");
        miTexto1.setText("mi primer texto");
        miTexto2.setText("Mi secundo texto");
        miBotón1.setLabel("Mi primer botón");
        miCasilla1.setLabel("Elección 1");
        miCasilla2.setLabel("Elección 2");
        miCasilla3.setLabel("Elección 3");
        miCasilla4.setLabel("Elección 4");
        miCasilla3.setCheckboxGroup(miGrupo1);
        miCasilla4.setCheckboxGroup(miGrupo1);
        miElección1.add("Uno");
        miElección1.add("Dos");
        miElección1.add("Tres");
        miLista1.add("Primero");
        miLista1.add("Secundo");
        miLista1.add("Tercero");
        miLista1.setMultipleMode(true);
        miScrollBar1.setValues(
                    0, // initial value is 0
                    10, // width of slider
                 -100, 100); // range -100 to 100)
        miScrollBar1.setOrientation(Scrollbar.HORIZONTAL);

        miPanel1.add(miEtiqueta1);
        miPanel1.add(miTexto1);
        miPanel1.add(miTexto2);
        miPanel1.add(miBotón1);
        miPanel1.add(miCasilla1);
        miPanel1.add(miCasilla2);
        miPanel1.add(miCasilla3);
        miPanel1.add(miCasilla4);
        miPanel1.add(miElección1);
        miPanel1.add(miLista1);
        miPanel1.add(miScrollBar1);

        miPanel2.add(new TextField ("Para texto muy largo"));
        miPanel1.add(miPanel2);

        menuArchivos.add(guardar);
        menuArchivos.add(salir);
        miMenu.add(menuArchivos);
}

void initEvents(){
        this.addWindowListener(new WindowAdapter() {
     public void windowClosing(WindowEvent e) {
         System.exit(0);
```



```
            }
        }); // Closes addWindowListener method
           miBotón1.addActionListener(new ActionListener() {
                   public void actionPerformed(ActionEvent e) {
                           try{
                                   System.out.println("Hola");
                                   }
                           catch (Exception ee) {
                                   System.out.println(ee.getStackTrace());
                                   }
                           }
                   });// Closes addActionListener method
           guardar.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               System.out.println("Guardar");
           }
        }); // Closes addActionListener method
           salir.addActionListener(new ActionListener() {
           public void actionPerformed(ActionEvent e) {
               System.exit(0);
           }
        }); // Closes addActionListener method
       }

       void testTextArea (){
           String s =
               "This is a very long message " +
               "It should wrap when there is " +
               "no horizontal scrollbar.";
           miPanel1.add(new TextArea (s, 4, 15,
                 TextArea.SCROLLBARS_NONE));
           miPanel1.add(new TextArea (s, 4, 15,
                 TextArea.SCROLLBARS_BOTH));
           miPanel1.add(new TextArea (s, 4, 15,
                 TextArea.SCROLLBARS_HORIZONTAL_ONLY));
           miPanel1.add(new TextArea (s, 4, 15,
                 TextArea.SCROLLBARS_VERTICAL_ONLY));
       }
}
```


5 Eventos

5.1 Principio de evento

En Java, para poder interaccionar con el usuario, el Java tiene que 'escuchar' a las acciones del usuario.

Por eso, hay de declarar escuchadores sobre los objetos (componentes) por los cuales esperamos una acción.

5.2 Objetos al origen de eventos

Existen diferentes objetos que pueden ser al origen de una acción. Cada objeto acepta un cierto tipo de escuchador. Sólo el objeto de tipo `Component` acepta varios escuchadores diferentes.

Eventos de bajo nivel	
Objeto	***Escuchador disponible***
Component	ComponentListener FocusListener KeyListener MouseListener MouseMotionListener
Container	ContainerListener
Window	WindowListener

Eventos de alto nivel	
Objeto	***Escuchador disponible***
Button List MenuItem TextField	ActionListener
Choice Checkbox Checkbox CheckboxMenuItem List	ItemListener
Scrollbar	AdjustmentListener
TextArea TextField	TextListener

5.3 Escuchadores

Cada escuchador (`Listener`) tiene uno o varios métodos posibles. Los escuchadores son interfaces o clase por los cuales los métodos no están implementadas, así que hay que implementarlas.

Lista de métodos de cada escuchador

Interface	*Method(s)*
ActionListener	`actionPerformed(`ActionEvent `e)`
AdjustmentListener	`adjustmentValueChanged(`AdjustmentEvent `e)`
ComponentListener	`componentHidden(`ComponentEvent `e)`
	`componentMoved(`ComponentEvent `e)`
	`componentResized(`ComponentEvent `e)`
	`componentShown(`ComponentEvent `e)`
ContainerListen er	`componentAdded(`ContainerEvent `e)`
	`componentRemoved(`ContainerEvent `e)`
FocusListener	`focusGained(`FocusEvent `e)`
	`focusLost(`FocusEvent `e)`
ItemListener	`itemStateChanged(`ItemEvent `e)`
KeyListener	`keyPressed(`KeyEvent `e)`
	`keyReleased(`KeyEvent `e)`
	`keyTyped(`KeyEvent `e)`

MouseListener	mouseClicked(MouseEvent e)
	mouseEntered(MouseEvent e)
	mouseExited(MouseEvent e)
	mousePressed(MouseEvent e)
	mouseReleased(MouseEvent e)
MouseMotionListener	mouseDragged(MouseEvent e)
	mouseMoved(MouseEvent e)
TextListener	textValueChanged(TextEvent e)
WindowListener	windowActivated(WindowEvent e)
	windowClosed(WindowEvent e)
	windowClosing(WindowEvent e)
	windowDeactivated(WindowEvent e)
	windowDeiconified(WindowEvent e)
	windowIconified(WindowEvent e)
	windowOpened(WindowEvent e)

La implementación del escuchador se puede hacer en uno o dos pasos.

5.3.1 Implementación en un paso

Cuando se implementa en un solo paso, el métodos del escuchador se sobre define directamente a la declaración del escuchador.

Ejemplo:

```
this.addWindowListener(new WindowAdapter() {
            public void windowClosing(WindowEvent e) {
                System.exit(0);
            }
        }); // Closes addWindowListener method
```

5.3.2 Implementación en dos pasos

Cuando el escuchador se podría reutilizar para otros componentes, se implementa en una clase aparte.

Si el escuchador sólo se utilizara para este objeto, se puede definir utilizando una clase interna. Si tiene que ser disponible para otros objetos, es mejor ponerlo en una clase aparte.

Ejemplos:

```
class CierraVentana extends WindowAdapter {
         public void windowClosing(WindowEvent e) {
           // Action Command is not necessarily label
          System.exit(0);
         }
       }

class MyActionListener implements ActionListener {
         public void actionPerformed(ActionEvent e) {
           // Action Command is not necessarily label
           String s = e.getActionCommand();
           if (s.equals("Quit")) {
             System.exit(0);
           }
           else if (s.equals("Hello")) {
             System.out.println("Bon Jour");
           }
           else {
             System.out.println(s + " selected");
```

```
            }
          }
        }
```

Remplazar initEvents() con el código siguiente:

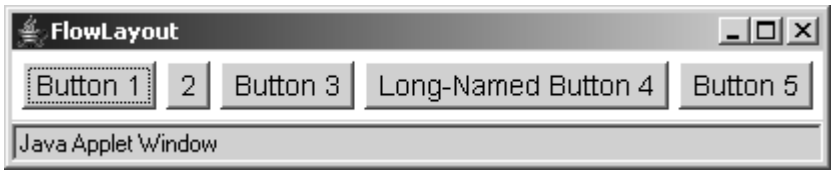
```
void initEvents(){
this.addWindowListener(new CierraVentana()); // Closes addWindowListener method
miBotón1.addActionListener(new MyActionListener());// Closes addActionListener method
guardar.addActionListener(new MyActionListener()); // Closes addActionListener method
salir.addActionListener(new MyActionListener()); // Closes addActionListener method
}
```

6 Layouts

Para manejar la presentación de los componentes, existen varios objetos que permiten presentar los componentes.

6.1 FlowLayout

El `FlowLayout` es el formato por defecto para los paneles. Es de lo más sencillo: el pone los componentes uno a lado de otro, pasando a la línea cuando es necesario (según el tamaño del panel).

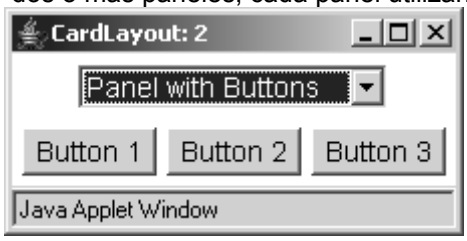


6.2 BorderLayout

El `BroderLayout` es el formato por defecto para las ventanas (`Frame`, `Dialog`). El está divido en cinco áreas: `North`, `South`, `East`, `West`, y `Center`.

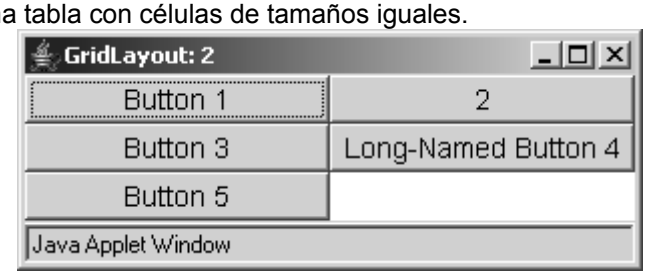
6.3 CardLayout

El `CardLayout` permite manejar dos o más paneles, cada panel utilizando su propio `Layout Manager`.



6.4 GridLayout

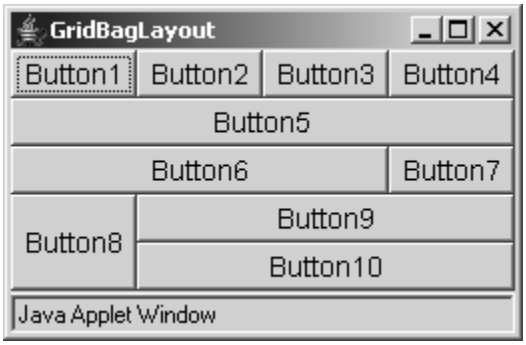
El `GridLayout` crea una tabla con células de tamaños iguales.



6.5 GridBagLayout

El `GridBagLayout` es el `Layout Manager` más poderoso del AWT. El permite manejar la disposición de los componentes cómo una tabla, el la cual la células pueden expandir sobre varias columnas, y las líneas pueden tener varias alturas, así cómo las columnas pueden tener varios tamaño de ancho.

El `GridBagLayout` utiliza un objeto `GridBagConstraint` que contiene el formato que se aplica a una celda. Antes de aplicar el formato, se modifica el formato según sus deseos (por defecto, tiene los valores del ultimo formato utilizado, o valores de defecto si es nuevo), y se aplica el formato al adjuntar el objeto en el contenedor.



Ejemplo:
```
Panel p = new Panel();
p.setLayout(GridBagLayout);
TextField txtF1 = new TextField(10);
GridBagConstraint c = new GridBagConstraint;
c.ipadx = 100;
c.anchor = GridBagConstraints.FIRST_LINE_START;
c.gridwidth = 2;
p.add(txtF1, c);
```

Para pasar a la línea (llenando las columnas que quedan) el atributo `gridwidth` debe tener el valor siguiente: `GridBagConstraints.REMAINDER`.

Cómo el `GridBagLayout` tiene muchas posibilidades de formato, aconsejo de crear una clase para manejarlo. Un ejemplo de clase para manejarlo está disponible aquí abajo.

Ejemplo de clase:
```
import java.awt.*;
public class BuildGrid {
       BuildGrid(){}
    void setLabel(Panel p, GridBagConstraints c, Label p_objeto, int p_ipadx,
                  int p_ipady,
                  int p_colwidth, int p_anchor) {
        c.gridx = GridBagConstraints.RELATIVE;
        c.gridy = GridBagConstraints.RELATIVE;
        c.ipadx = p_ipadx;
        c.ipady = p_ipady;
        c.anchor = p_anchor;
        c.gridwidth = p_colwidth;
        p.add(p_objeto, c);
    }
    void setTextField(Panel p, GridBagConstraints c, TextField p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
              c.gridy = GridBagConstraints.RELATIVE;
              c.ipadx = p_ipadx;
              c.ipady = p_ipady;
              c.anchor = p_anchor;
```

```
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setTextArea(Panel p, GridBagConstraints c, TextArea p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setButton(Panel p, GridBagConstraints c, Button p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setList(Panel p, GridBagConstraints c, List p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setChoice(Panel p, GridBagConstraints c, Choice p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setCheckbox(Panel p, GridBagConstraints c, Checkbox p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setScrollbar(Panel p, GridBagConstraints c, Scrollbar p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
       c.gridx = GridBagConstraints.RELATIVE;
            c.gridy = GridBagConstraints.RELATIVE;
            c.ipadx = p_ipadx;
            c.ipady = p_ipady;
            c.anchor = p_anchor;
            c.gridwidth = p_colwidth;
            p.add(p_objeto, c);
    }
    void setPanel(Panel p, GridBagConstraints c, Panel p_objeto, int p_ipadx,
            int p_ipady,
```

```
            int p_colwidth, int p_anchor) {
        c.gridx = GridBagConstraints.RELATIVE;
                c.gridy = GridBagConstraints.RELATIVE;
                c.ipadx = p_ipadx;
                c.ipady = p_ipady;
                c.anchor = p_anchor;
                c.gridwidth = p_colwidth;
                p.add(p_objeto, c);
    }
    void setScrollPane(Panel p, GridBagConstraints c, ScrollPane p_objeto, int p_ipadx,
            int p_ipady,
            int p_colwidth, int p_anchor) {
        c.gridx = GridBagConstraints.RELATIVE;
                c.gridy = GridBagConstraints.RELATIVE;
                c.ipadx = p_ipadx;
                c.ipady = p_ipady;
                c.anchor = p_anchor;
                c.gridwidth = p_colwidth;
                p.add(p_objeto, c);
    }
}
```


7 Componentes Swing

7.1 Versión SWING de los componentes AWT

La versión Swing de los componentes AWT tienen los mismos nombres, pero con el prefijo 'J'. Así existe el JtextArea, Jlabel, Jframe, etc...

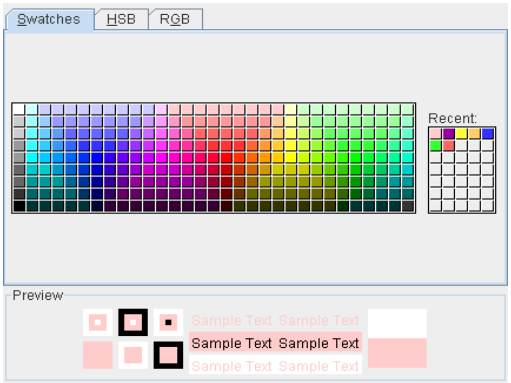
Una de las difenrecias entre Swing y AWT es que en los componenetes Swing se puede usar el HTML para formatear el contenido.

7.2 Nuevos componentes Swing

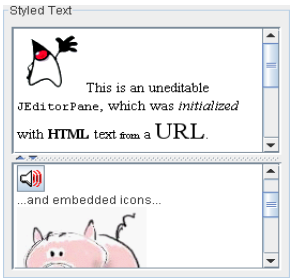
Además de los compenentes que ya existen en AWT, la biblioteca Swint agrega muchos otros componentes como:

Jslider

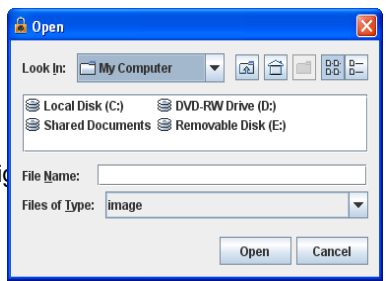
JSpinner



JColorChooser



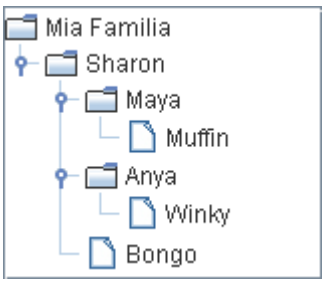
JEditorPane and JtextPane



JFileChooser

Host	User	Password	Last Modified
Biocca Games	Freddy	!#asf6Awwzb	Mar 16, 2006
zabble	ichabod	Tazb!34$fZ	Mar 6, 2006
Sun Developer	fraz@hotmail.co...	AasW541fbZ	Feb 22, 2006
Heirloom Seeds	shams@gmail....	bkz[ADF78!	Jul 29, 2005
Pacific Zoo Shop	seal@hotmail.c...	vbAf124%z	Feb 22, 2006

JTable

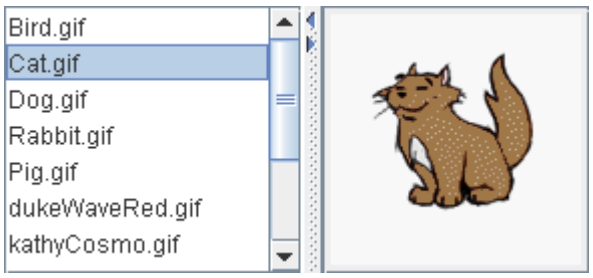


JTree

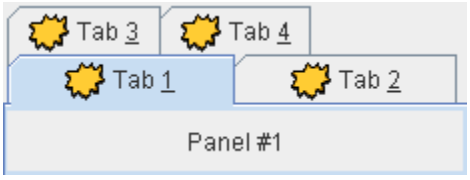
JProgressBar

JSeparator

JtoolTip



JSplitPane



JTabbedPane

JtoolBar

Para utilizar cada tipo de componente Swing, ver en la documentación Java en el CD del material del curso, así que en la documentación disponible en línea.

8 Ejercicios

1) Crear una ventana y probar los diferente tipos de componentes.

2) Crear un TextField, un botón, y un TextArea. Cuando se empuja el botón, el texto contenido en el TextField debe ser adjuntado al contenido del TextArea, y el valor del TextField vaciado.

3) Crear un programa para maneja una lista de compras. Empujando el botón, el articulo mencionado está adjuntado a la lista de compras del super y/o de la venta según la casilla marcada.

Ejercicio final:
Crear una aplicación sencilla pero funcional, que se conecta a la base de datos del curso, y permite crear, modificar, y borrar empleados usando los componentes de AWT y/o Swing.

9 Esquema de la BD 'curso'