Universidad de Alcalá

Departamamento Ciencias de la Computación

Programación Orientada a Objetos

Tema 6:

Desarrollo de interfaces gráficas de usuario

Tema 6-1: Conceptos básicos de SWING

Universidad de Alcalá

Departamento Ciencias de la Computación

Contenidos

- Tema 6-1: Conceptos básicos de SWING

INTRODUCCIÓN

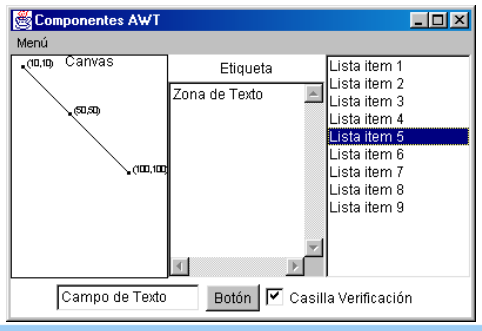
- **Cuando se desarrolla un sistema informático**, no solo se espera que funcione bien y que se adapte a la funcionalidad que el usuario requería, sino que la forma en que éste interactúa con el sistema sea lo más sencilla y cómoda posible.
- Para lograr este objetivo, buena parte del desarrollo del sistema se centra en la construcción de la ***interfaz de usuario*** que es la parte del programa que permite a éste interactuar con el usuario.
- Los lenguajes de programación usados hoy en día utilizan la **programación orientada a objetos** a la hora de diseñar y programar una interfaz gráfica de usuario.
- Los **componentes** de la interfaz (botones, listas, barras de menú, cajas de texto, etc.) se consideran objetos que permiten ser manejados a través de sus métodos.

COMPONENTES Y CONTENEDORES

- Los Contenedores contienen Componentes, que son los controles básicos.
- Si no se usan posiciones fijas de los Componentes, éstos se sitúan a través de una disposición controlada (layouts) en el Contenedor.
- Podemos controlar eventos de todo tipo acercándonos incluso al teclado o al ratón.
- Hay que tratar de separar lo más posible la interfaz gráfica de la lógica de negocio.
- En Java existen dos bibliotecas de clases para realizar aplicaciones gráficas:
  - AWT (más antigua).
  - Swing.

COMPONENTES Y CONTENEDORES

- Los **Componentes** son una serie de elementos gráficos básicos, por ejemplo: botones, etiquetas, listas, casillas de verificación, campos de texto o barras de desplazamiento.
- En AWT y Swing, todos los Componentes de la interface de usuario son instancias de la clase **Component** o uno de sus subtipos.
- Estos Componentes no se encuentran aislados, sino que deben agruparse dentro de **Contenedores**. En AWT y Swing, todos los Contenedores son instancias de la clase **Container** o uno de sus subtipos.



java.lang.Object
└ java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent

SWING

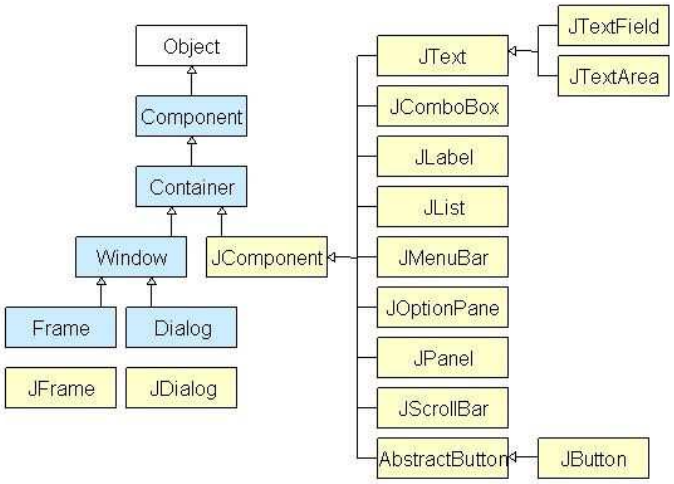
- Las clases que ofrece el API **Swing** nos permite desarrollar interfaces gráficas de usuario mucho más potentes y aparentes que con el **AWT**, debido a la cantidad y calidad de los controles gráficos que ofrece.

- **El API Swing pertenece a las JFC** (Java Foundation Classes), es decir, un conjunto de APIs que proporcionan al programador una serie de herramientas para el desarrollo de interfaces gráficas de usuario, estas herramientas son las siguientes:

  – Los componentes Swing.
  – Look & Feel (Windows, Motif, Macintosh, Java o Metal).
  – Accesibilidad.
  – Java 2D.
  – Drag and Drop.

- La clase **javax.swing.JComponent** es la clase base de la jerarquía de componentes Swing, es una subclase de la clase java.awt.Container y por lo tanto es a la vez un componente y un contenedor en el sentido del AWT.

java.awt.Component
└ java.awt.Container
└ javax.swing.JComponent



- Los eventos dependen del tipo de **componente** que los generen, pero no hay un único tipo de **evento** para cada componente, sino que estos son compartidos por muchos tipos de componentes.

- Cada tipo de evento tiene asociado un **oyente** (listener) y cada oyente requiere de un **manipulador** (handler) dotado de uno o más **métodos** que serán completados con nuestro código.

- Ejemplo:

EVENTO	OYENTE	MÉTODO	COMPONENTES
ActionEvent	**ActionListener**	**actionPerformed**	**JButton** **JComboBox** **JMenuItem** **JTextField**

COMPONENTES BÁSICOS

- **Etiquetas:**
  - Las etiquetas en Swing se implementan con la clase **JLabel**, permiten la incorporación de iconos y la alineación.

```
JLabel etiq1 = new JLabel(); etiq1.setText( "Etiqueta1" );
JLabel etiq2 = new JLabel( "Etiqueta2" );
etiq2.setFont( new Font( "Helvetica", Font.BOLD, 18 ) );
Icon imagen = new ImageIcon( "javalogo.gif" );
JLabel etiq3 = new JLabel( "Etiqueta3", imagen, SwingConstants.CENTER );
etiq3.setVerticalTextPosition( SwingConstants.TOP );
JLabel etiq4 = new JLabel( "Etiqueta4",SwingConstants.RIGHT );
```

Universidad de Alcalá
Departamento de Ciencias de la Computación

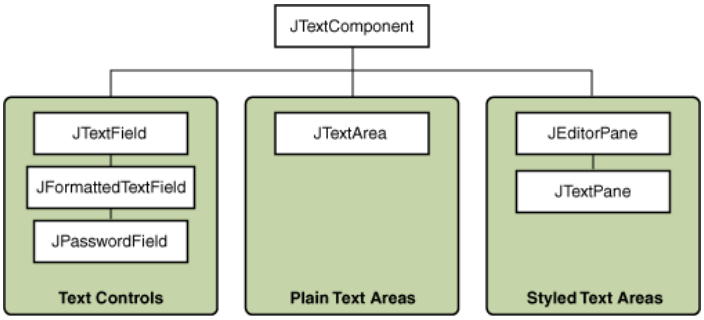
COMPONENTES BÁSICOS

- **Botones de Pulsación:**
  - Los botones se construyen con la clase **JButton**. Pueden tener una etiqueta y también pueden contener un icono o ambas cosas.
  - Los eventos que producen son del tipo ActionEvent y se tratan implementando la interfaz ActionListener dentro del método actionPerformed().
  - Para que un botón produzca eventos debemos indicarlo mediante el método addActionListener().

```
Icon icono1 = new ImageIcon( "pto-roj.gif" );
JButton boton1 = new JButton("Uno",icono1);
boton1.addActionListener(this);
```

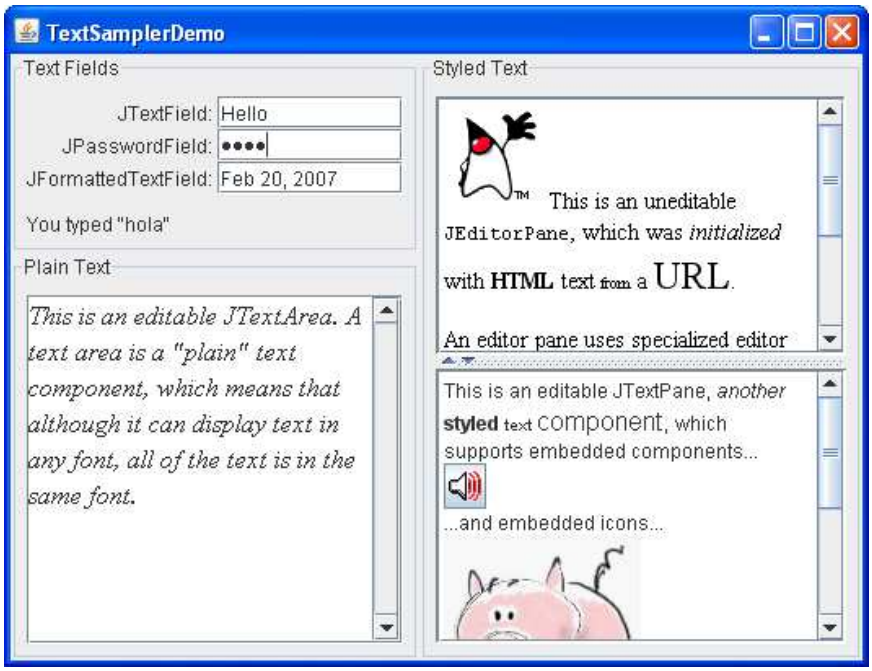
COMPONENTES BÁSICOS

- **Campos de Texto y Áreas de Texto :**
  - Los campos de texto se implementan mediante la clase **JTextField**. **JFormattedTextField** sirve para poner texto con formato, por ejemplo: fechas, números, monedas, etc. Para poner un campo de contraseña tenemos que hacer uso de la clase **JPasswordField**.
    - `JTextField textField = new JTextField();`
    - `JPasswordField textPasw = new JPasswordField(10);`
    - `textPasw.setEchoChar('*');`
  - Para capturar el contenido de una caja de texto se utiliza el método getText() y para escribir, el método setText(cadena).
  - Para implementar áreas de texto mediante Swing se utiliza la clase **JTextArea**.



COMPONENTES BÁSICOS

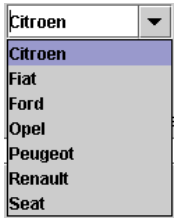
- **Ejemplo Campos de Texto y Áreas de Texto:**



COMPONENTES BÁSICOS

- **Listas Desplegables:**
  - Las listas desplegables se deben crear mediante la clase **JComboBox**.
  - Para crearlas utilizamos su constructor pasándole como argumento las distintas opciones que contendrá mediante un array de cadenas.
  - Produce eventos de tipo ActionEvent y el tratamiento es igual que el de los botones (ActionListener, addActionListener(), actionPerformed()).
  - Se pueden incorporar más opciones a la lista (en ejecución), para ello tenemos que hacer uso del método setEditable(true).

```
String[] datos_marcas =
 {"Citroen","Fiat","Ford","Opel","Peugeot","Renault","Seat"};
JComboBox marcas = new JComboBox(datos_marcas);
marcas.setEditable(true);
marcas.addActionListener(this);
```



Universidad de Alcalá
Departamento de Ciencias de la Computación

COMPONENTES BÁSICOS

- **Listas:**
  - Utilizamos objetos de tipo **JList** y les pasamos como argumento las distintas opciones que contendrá. Debemos seleccionar el modo de selección que le vamos a aplicar mediante el método *setSelectionMode(selectionMode)*. Los distintos modos de selección son los siguientes:
    - SINGLE_SELECTION
    - SINGLE_INTERVAL_SELECTION
    - MULTIPLE_INTERVAL_SELECTION

  - Los eventos que producen son del tipo ListSelectionEvent y se tratan implementando la interfaz ListSelectionListener dentro del método valueChanged().
  - Para que una lista produzca eventos debemos indicarlo mediante el método addListSelectionListener().
  - Para proporcionar scroll automáticamente a una lista hay que asociarla a un JScrollPane.

```
String[] datos_equipo = {"ABS","Aire Ac.","Air-bag",
                         "Dir. Asistida","EE","Pintura Met.","Radio"};
JList equipo = new JList(datos_equipo);
equipo.setSelectionMode(ListSelectionModel.MULTIPLE_INTERVAL_SELECTION);
JScrollPane scrollPane = new JScrollPane(equipo);
equipo.addListSelectionListener(this);
```

COMPONENTES BÁSICOS

- **Casillas de verificación:**
  - Para construir casillas de verificación se utiliza la clase **JCheckBox**.
  - El constructor más completo permite poner un texto, un icono y si se trata de una opción que se quiera tener seleccionada ponerla a true. Por ejemplo:

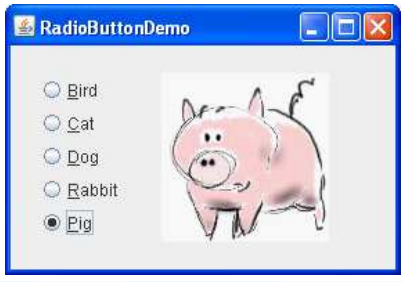
```
Icon icono = new ImageIcon("icono.gif");
JCheckBox cboton1 = new JCheckBox("opción 1", icono, true);
cboton1.addItemListener(this);
```

  - Los eventos que producen son del tipo ItemEvent y se tratan implementando la interfaz ItemListener dentro del método itemStateChanged().
  - Para que una casilla de verificación produzca eventos debemos indicarlo mediante el método addItemListener().

COMPONENTES BÁSICOS

- **Botones de Radio:**
  - Los botones de radio o de comprobación se implementan a través de la clase **JRadioButton** que al igual que los JCheckBox nos permiten agregar iconos en las distintas opciones.
  - Una peculiaridad de este tipo de botones es la necesidad de agruparlos de forma que siempre haya un único botón activo. Para realizarlo tan solo tenemos que crear un objeto de tipo **ButtonGroup** y agregarle cada una de las opciones que tengamos.
  - El tratamiento de los eventos es el mismo que el de los botones.

```
JRadioButton rboton1 = new JRadioButton("opción 1");
rboton1.addActionListener(this);
JRadioButton rboton2 = new JRadioButton("opción 2");
rboton2.addActionListener(this);
ButtonGroup grouporb = new ButtonGroup();
grouporb.add(rboton1);
grouporb.add(rboton2);
```

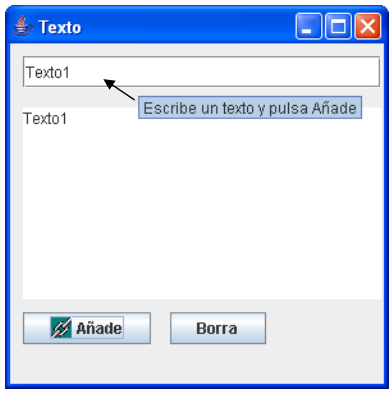


COMPONENTES BÁSICOS

- **Ayuda Emergente:**
  - En Swing se puede asignar a los componentes gráficos una ayuda emergente de forma que cuando el usuario pase el ratón por el componente se le presente un texto indicativo de lo que puede hacer con ese componente.

  - Para realizarlo utilizamos el método **setToolTipText("texto ayuda")**.

```
JTextField textField = new JTextField();
textField.setToolTipText("Escribe un texto y pulsa Añade");
```



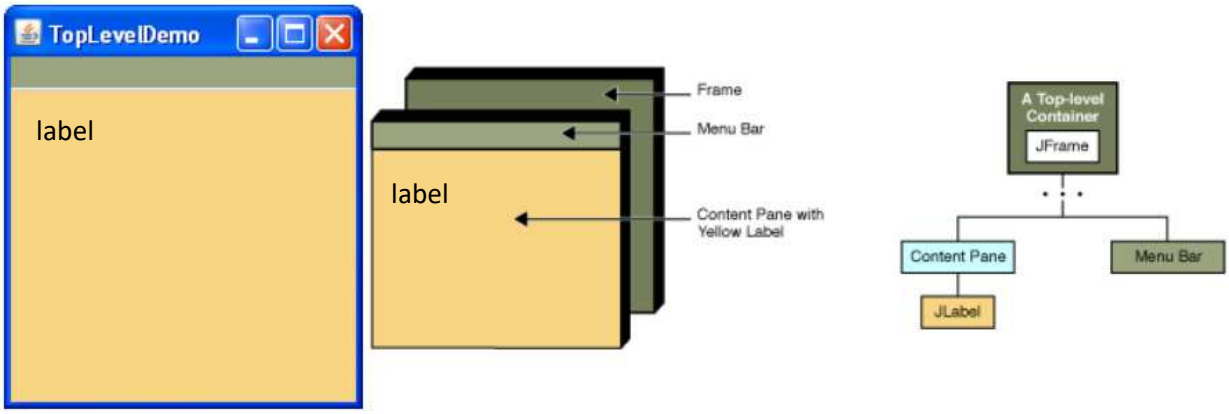
CONTENEDORES

- Los componentes deben situarse en un contenedor, Swing proporciona diversos contenedores como:
  - **JWindow:** Es una superficie de pantalla de alto nivel (una ventana). Una instancia de esta clase no tiene ni título ni borde. Contiene métodos generales para el manejo de ventanas.
  - **JFrame:** Un JFrame o marco es una **ventana con borde y título**. Una instancia de esta clase puede tener una **barra de menú**.
  - **JDialog:** Es una ventana con borde y título pero con la característica especial de que una instancia de esta clase no puede existir sin una instancia asociada de la clase JFrame.
  - **JApplet:** Permite crear un applet swing.
  - **JPanel:** Es un contenedor de nivel intermedio que corresponde con el área de trabajo de la ventana. Es un Contenedor genérico de Componentes.

CONTENEDORES

- Cada contenedor de nivel superior tiene de forma predeterminada un panel raíz al que se puede acceder por medio del método getContentPane().
- Para añadir a este panel los diversos componentes se utiliza el método add().

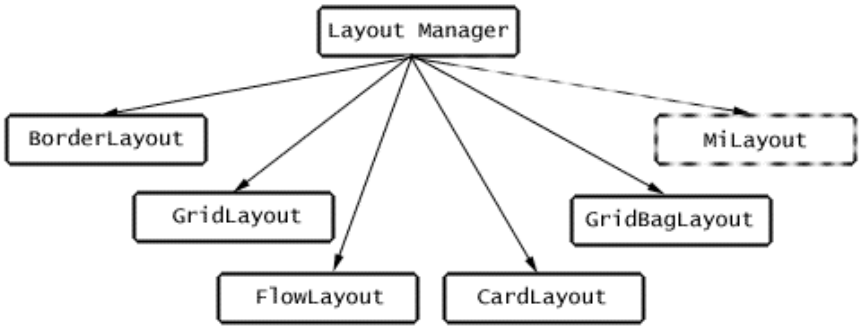
```
this.getContentPane().add(label);          //una forma
Container contenedor = getContentPane();   //otra forma
contenedor.add(label);
```



LAYOUTS

- Los *layout managers* o **gestores de disposición** ayudan a adaptar los diversos Componentes que se desean incorporar a un JPanel. Es decir, especifican la disposición que tendrán los Componentes a la hora de colocarlos sobre un Contenedor.

- Java proporciona estos esquemas predefinidos de disposición de componentes para su correcta visualización en todas las plataformas, y en todas las configuraciones de pantalla que los usuarios pudieran tener.



- **FlowLayout**:
  - Es el más simple y el que se utiliza por defecto en todos los paneles, si no se fuerza el uso de alguno de los otros.
  - Los componentes añadidos a un contenedor con FlowLayout se encadenan en forma de lista, es decir, formando una cadena horizontal, de izquierda a derecha y de arriba abajo. Nos permite seleccionar el espaciado entre cada uno de los componentes.

- **FlowLayout:**

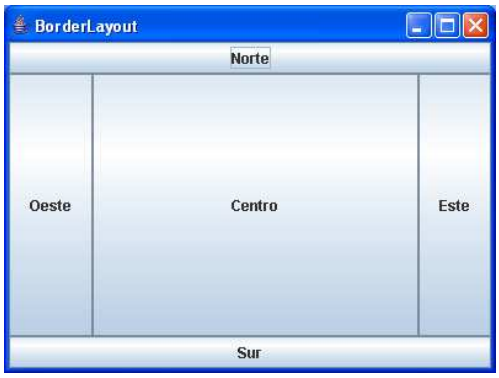

```
import java.awt.*;
import javax.swing.*;

public class FlowL extends JFrame {
  //componentes
  JLabel etiqueta = new JLabel("FlowLayout");
  JButton boton1 = new JButton("Uno");
  JButton boton2 = new JButton("Dos");
  JButton boton3 = new JButton("Tres");

  public static void main(String args[]) {
         FlowL fl = new FlowL();
  }

  public FlowL() {
         super("FlowLayout");
         this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         this.setup();
  }
```

- **FlowLayout:**


```
public void setup() {
        Container contenedor = getContentPane();
        contenedor.setLayout(new FlowLayout());
        contenedor.add(etiqueta);
        contenedor.add(boton1);
        contenedor.add(boton2);
        contenedor.add(boton3);

        this.setSize( 300,100 );
        this.setVisible( true );
    }
}
```


- **BorderLayout:**
  - El gestor de disposición BorderLayout proporciona un esquema más complejo de colocación de los componentes.
  - La composición utiliza cinco zonas para colocar los componentes sobre ellas: North, South, East, West, y Center (Norte, Sur, Este, Oeste y Centro). Es el layout que se utilizan por defecto en las clases JFrame y JDialog.

- **BorderLayout:**

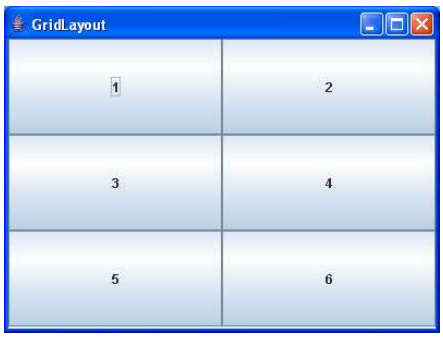



```
import java.awt.*;
import javax.swing.*;

public class BorderL extends JFrame {
   //componentes
   JButton botonN = new JButton( "Norte" );
   JButton botonS = new JButton( "Sur" );
   JButton botonE = new JButton( "Este" );
   JButton botonO = new JButton( "Oeste" );
   JButton botonC = new JButton( "Centro" );

   public static void main(String args[]) {
            BorderL bl = new BorderL();
   }

   public BorderL() {
            super("BorderLayout");
   this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setup();
   }
```

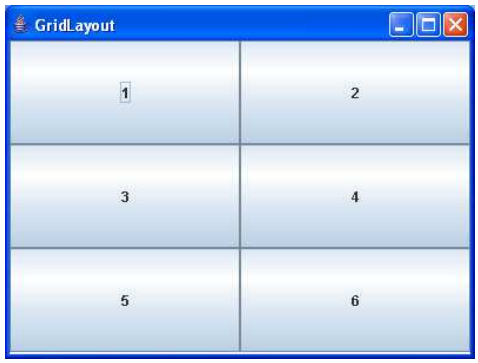

- **BorderLayout:**

```
   public void setup() {
            Container contenedor = getContentPane();
            contenedor.setLayout(new BorderLayout());
            contenedor.add( "North",botonN );
            contenedor.add( "South",botonS );
            contenedor.add( "East",botonE );
            contenedor.add( "West",botonO );
            contenedor.add( "Center",botonC );

            this.setSize( 400,300 );
            this.setVisible( true );
   }
}
```

- **GridLayout:**
  - Es uno de los gestores de disposición que proporciona más flexibilidad para situar componentes. Este layout se crea mediante la utilización de una tabla para situar cada uno de los componentes en una celda determinada por su número de fila y columna. Los componentes tienen el mismo tamaño.
  - En la siguiente figura se muestra una ventana que usa este tipo de disposición para posicionar seis botones en su interior, con tres filas y dos columnas que crearán las seis celdas necesarias para albergar los botones:



- **GridLayout:**



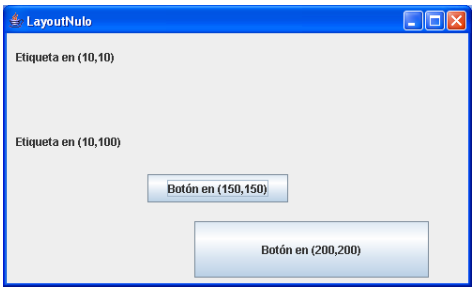

```
import java.awt.*;
import javax.swing.*;

public class GridL extends JFrame {
   //componentes
   JButton boton1 = new JButton( "1" );
   JButton boton2 = new JButton( "2" );
   JButton boton3 = new JButton( "3" );
   JButton boton4 = new JButton( "4" );
   JButton boton5 = new JButton( "5" );
   JButton boton6 = new JButton( "6" );

   public static void main(String args[]) {
            GridL gl = new GridL();
   }

   public GridL() {
            super("GridLayout");
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setup();
   }
```

- **GridLayout:**

```
public void setup()  {

        Container contenedor = getContentPane();
        contenedor.setLayout(new GridLayout(3,2));
        contenedor.add( boton1 );
        contenedor.add( boton2 );
        contenedor.add( boton3 );
        contenedor.add( boton4 );
        contenedor.add( boton5 );
        contenedor.add( boton6 );

        this.setSize( 400,300 );
        this.setVisible( true );
    }
}
```

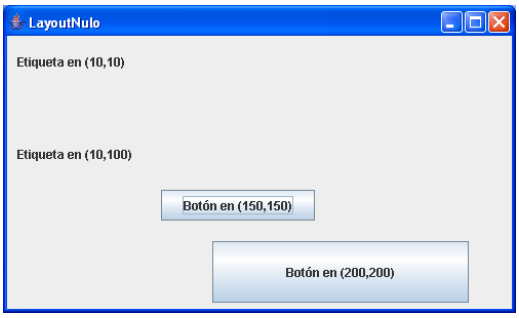
- **Layout Nulo:**

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

public class LayoutNulo extends JFrame {
   //componentes
   JLabel etiqueta1 = new JLabel("Etiqueta en (10,10)");
   JLabel etiqueta2 = new JLabel("Etiqueta en (10,100)");
   JButton boton1 = new JButton("Botón en (150,150)");
   JButton boton2 = new JButton("Botón en (200,200)");

   public static void main(String args[])  {
            LayoutNulo fl = new LayoutNulo();
   }

   public LayoutNulo()  {
            super("LayoutNulo");
            this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
            this.setup();
   }
```

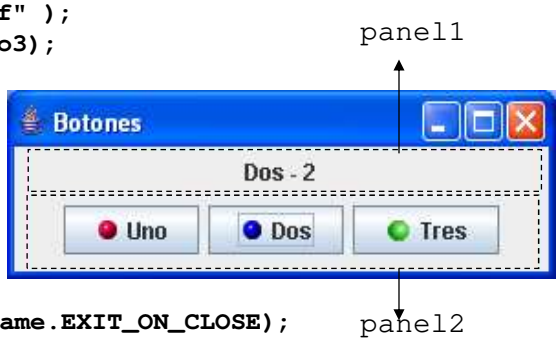


- **Layout Nulo:**

```
public void setup() {
        Container contenedor = getContentPane();
        contenedor.setLayout(null);
        etiqueta1.setBounds(10,10,200,30);
        etiqueta2.setBounds(10,100,200,30);
        boton1.setBounds(150,150,150,30);
        boton2.setBounds(200,200,250,60);
        contenedor.add(etiqueta1);
        contenedor.add(etiqueta2);
        contenedor.add(boton1);
        contenedor.add(boton2);

        this.setSize( 500,300 );
        this.setVisible( true );
  }
}
```




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

public class JBotones extends JFrame implements ActionListener {
  //Componentes
  JLabel etiqueta = new JLabel("Etiqueta Inicial");
  Icon icono1 = new ImageIcon( "pto-roj.gif" );
  JButton boton1 = new JButton("Uno",icono1);
  Icon icono2 = new ImageIcon( "pto-azu.gif" );
  JButton boton2 = new JButton("Dos",icono2);
  Icon icono3 = new ImageIcon( "pto-ver.gif" );
  JButton boton3 = new JButton("Tres",icono3);
  JPanel panel1 = new JPanel();
  JPanel panel2 = new JPanel();

  public static void main(String args[]) {
        JBotones jb = new JBotones();
  }
  public JBotones() {
        super("Botones");
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.setup();
  }
```



EJEMPLO APLICACIÓN

```
public void setup() {
        //situamos los componentes
        panel1.add(etiqueta);
        panel2.add(boton1);
        panel2.add(boton2);
        panel2.add(boton3);

        //activamos los eventos
        boton1.addActionListener(this);
        boton2.addActionListener(this);
        boton3.addActionListener(this);

        Container contenedor = getContentPane();
        contenedor.setLayout(new BorderLayout());
        contenedor.add("North",panel1);
        contenedor.add("Center",panel2);
}

//tratamos los eventos
public void actionPerformed(ActionEvent e)
{
        String s = e.getActionCommand(); //captura la etiqueta del botón

        if ( e.getSource() == boton1 ) etiqueta.setText(s+" - 1");
        if ( e.getSource() == boton2 ) etiqueta.setText(s+" - 2");
        if ( e.getSource() == boton3 ) etiqueta.setText(s+" - 3");
}
}
```