

### Table of Contents

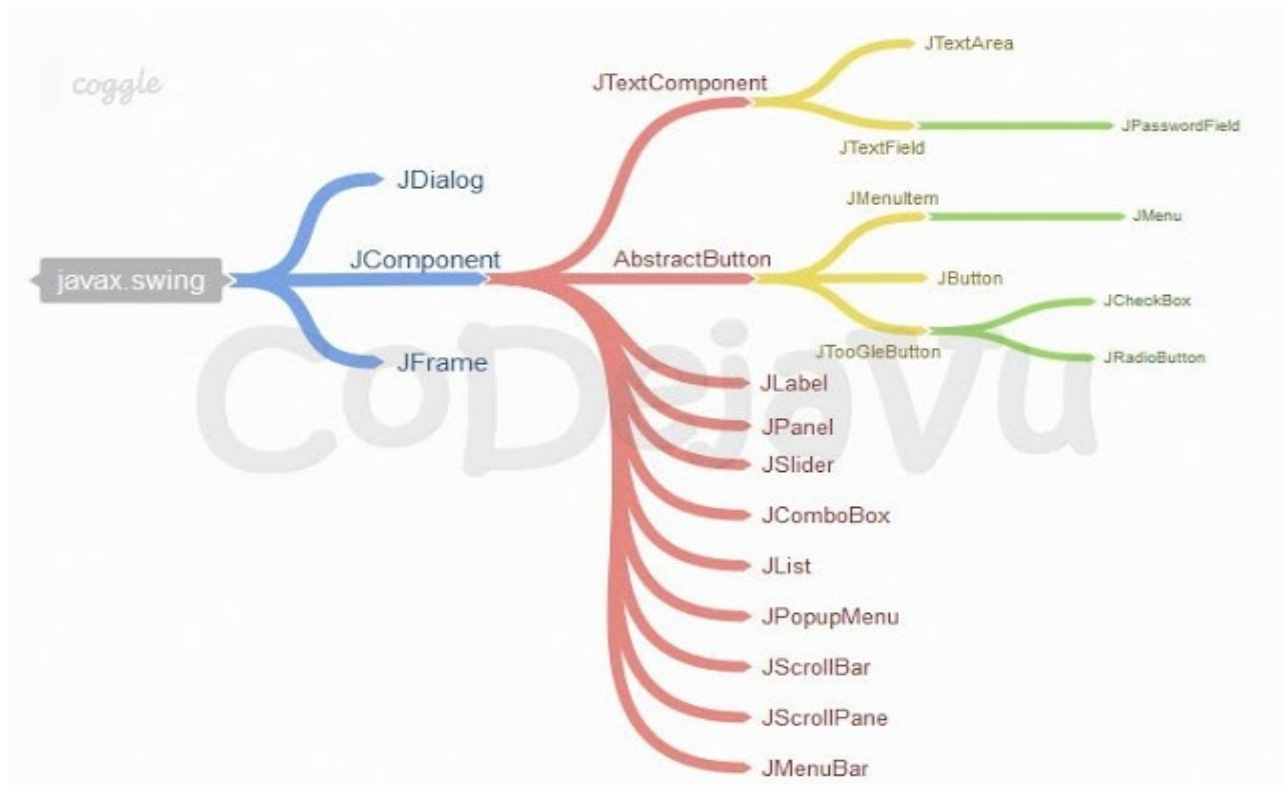
Swing.....	2
Pero Que es Java Swing?.....	2
JFrame y JDialog.....	3
JFrame o JDialog?.....	3
Componentes Java Swing.....	7
Que son Los Componentes Graficos?.....	7
Categorias.....	7
Contenedores.....	7
Componentes de Texto.....	8
Componentes de Menus.....	9
Componentes Atómicos Java Swing.....	10
Componentes De Texto Java Swing.....	18
Contenedores Java Swing.....	24

# Swing.

## Pero Que es Java Swing?

Es un paquete que hace parte de la Java Foundation Classes o mas conocida como JFC, la cual provee herramientas o facilidades para la construcción de **GUI's** o interfaces Graficas de Usuario (graphical user interface).

Podemos decir que **Swing** es la evolución del **AWT** (Abstract Window Toolkit), la cual al igual que **Swing** es un conjunto de librerías enfocadas a la construcción de interfaces, solo que con esta se presentaron algunos problemas en cuanto a portabilidad principalmente cuando se desarrollaban aplicaciones para diferentes sistemas operativos, pues el comportamiento de los componentes graficos en ocasiones podían variar..... bueno esa es otra historia, el punto es que a partir de **AWT** nace **Swing** y con el mejoras no solo en aspectos visuales sino también en portabilidad y comportamiento.....el siguiente es un ejemplo de la Jerarquía de esta librería (se muestran algunos de los principales componentes...)



Como vemos todos los componentes de **Swing** heredan del paquete **javax.swing** (obvio no?) a diferencia de los componentes **AWT** los **Swing** se reconocen porque anteponen la letra **J** antes del nombre, por ejemplo un botón en **AWT** se llama Button, mientras que en Java **Swing** es JButton.....

Cuando vamos a construir aplicaciones utilizando Java **Swing** debemos tener al menos un contenedor que será la base para nuestra aplicación, es decir, sera el lienzo donde pintaremos los demás componentes.

Normalmente podemos utilizar un JFrame o JDialog, estos serán la base para nuestra ventana y en ellos pintar los paneles, botones, cajas de texto, áreas entre otros.....

Con **Swing** le daremos vida a nuestro sistema, ya que se crearán las vistas de la aplicación, por medio de las cuales el Usuario interactuará con el sistema, veremos que se tiene una gran cantidad de posibilidades para estructurar nuestros desarrollos, se pueden manejar los eventos de cada componente dependiendo de nuestras necesidades, así como utilizar look & feel para modificar el aspecto visual de nuestras interfaces.

### JFrame y JDialog



Veremos simplemente como construir ventanas usando los componentes **JFrame** y **JDialog**, conoceremos en términos generales en que se diferencian y cual es la forma de crearlos....

Cuando vamos a construir aplicaciones con Java **Swing** debemos tener al menos un contenedor, este sera el tapiz donde pintaremos el conjunto de componentes que arman las interfaces, para eso podemos usar un **JFrame** o un **JDialog** (Aunque también se pueden usar Applets).

### JFrame o JDialog?

Estos componentes hacen parte del paquete **javax.swing**, básicamente nos permiten crear Ventanas para nuestras aplicaciones y en ellas alojar otros componentes para darle cuerpo a la interfaz de usuario, en si tienen

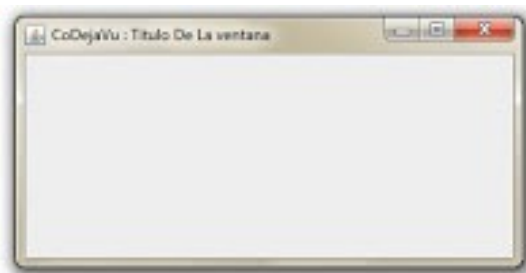
comportamientos similares y a simple vista pueden parecer iguales, pero son componentes distintos.

Un **JFrame** permite crear una ventana con ciertas características, por ejemplo podemos visualizarla en nuestra barra de tareas, caso contrario de los **JDialog**, ya que estos últimos son Ventanas de Dialogo con un comportamiento diferente, no se puede ver la ventana en la barra de tareas ni posee los botones comunes de maximizar o minimizar....

Los **JDialog** pueden ser hijos de **JFrames** o de otros **JDialog** mientras que los **JFrame** no (Como así Hijos?), es decir, si tenemos claros conceptos de programación Orientada a Objetos podemos relacionar esto de Hijos con el concepto de Herencia (Aunque no directamente, es mas a nivel conceptual), con estos componentes podemos hacer que una Ventana sea Padre de otra Ventana de tipo **JDialog**, asignándole algún tipo de comportamiento o dejando la ventana padre como Principal.

Es recomendable cuando trabajemos con **GUI's** crear un único **JFrame** para toda la aplicación y el resto de ventanas podemos trabajarlas como **JDialog** ya que de esa manera evitamos que se creen varias ventanas independientes y en vez de eso otras dependientes de la principal. Podemos encontrar muchas consideraciones sobre estos componentes pero lo anterior es básicamente lo que necesitamos saber en términos generales.....veamos su implementación.

## JFrame.



Como se mencionó anteriormente debemos tener un contenedor, para esto podemos utilizar la clase **Container**, esta será la encargada de "alojar" los componentes que queremos mostrar en la ventana (botones, labels, cajas de texto etc...), digo que "podemos" ya que nada nos impide alojar los

componentes directamente en el frame sin necesidad de usar el Container, sin embargo se recomienda el uso de este elemento para aprovechar las propiedades y métodos que nos puede brindar, y no dejar simplemente nuestros componentes regados....

Creamos la clase VentanaPrincipal, esta Hereda de **JFrame**, posteriormente declaramos el Objeto Contenedor y luego lo instanciamos mediante el método **getContentPane();**

```

1
2
3 public class ClaseFrame extends JFrame {
4     private Container contenedor;
5
6     public VentanaPrincipal()//constructor
7     {
8         contenedor=getContentPane();
9         contenedor.setLayout(null);
10        //Asigna un titulo a la barra de titulo
11        setTitle("Titulo De La ventana");
12        //tamaño de la ventana
13        setSize(400,200);
14        //pone la ventana en el Centrada en la pantalla
15        setLocationRelativeTo(null);
16    }
17 }

```



La clase anterior crea una Ventana Vacía, es importante tener muy claro que JFrame también es una clase de la cual se debe heredar para crear nuestras ventanas.....

## JDialog.

Un JDialog se construye de la misma manera que un JFrame la única diferencia es que hereda de la Clase JDialog..... como se mencionó anteriormente básicamente son ventanas pero tienen ciertas diferencias, la mas notoria es

que no posee los botones básicos de las ventanas ni pueden ser visualizados en la barra de tareas...

```

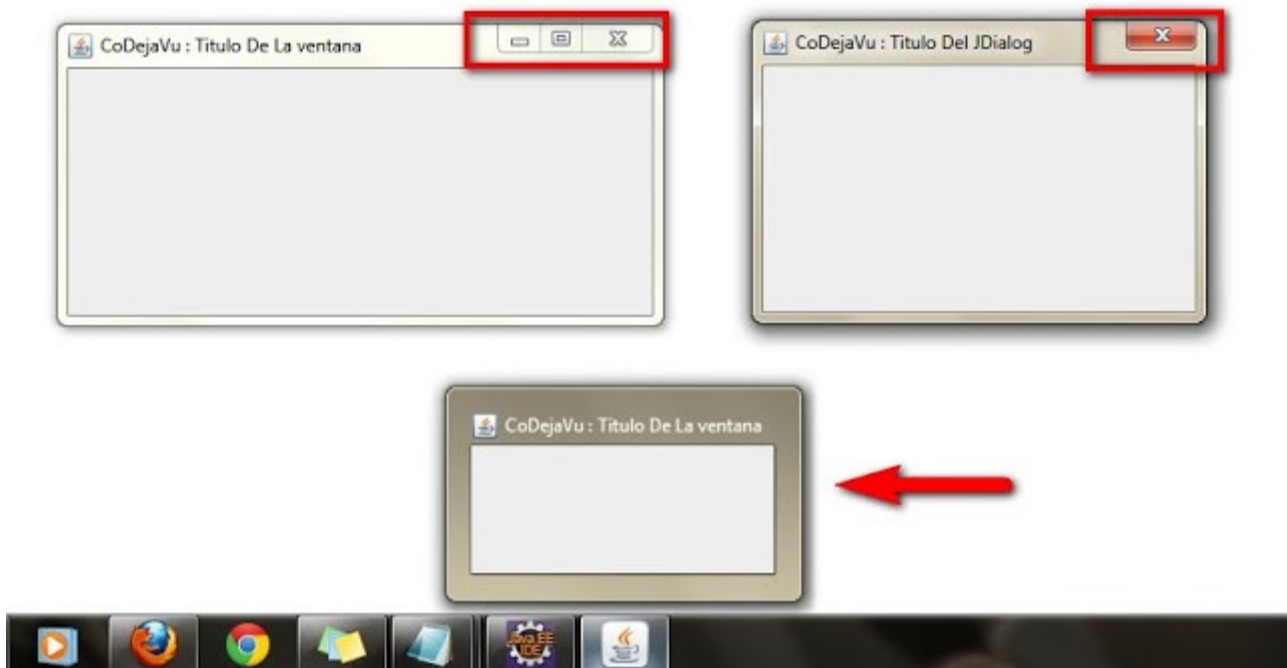
1
2
3 public class VentanaConfirmacion extends JDialog {
4     private Container contenedor;
5     public VentanaConfirmacion(){
6         contenedor=getContentPane();
7         contenedor.setLayout(null);
8         //Asigna un titulo a la barra de titulo
9         setTitle("Titulo Del JDialog");
10        //tamaño de la ventana
11        setSize(300,200);
12        //pone la ventana en el Centro de la pantalla
13        setLocationRelativeTo(null);
14    }
15}

```

Igual que El anterior, este código crea una ventana de dialogo Vacía, se llaman de dialogo por las características mencionadas que la diferencian de una ventana normal.

## Conclusiones.

Si ejecutamos las 2 clases veremos estas diferencias.



Como vemos lo principal en estos momentos es que la ventana que hereda de

**JFrame** tiene los botones básicos de toda ventana, a demás permite su visualización en la barra de tareas, mientras que la ventana que hereda de **JDialog** no posee estas características.

Y Listo!!!, la intención de esta entrada es Mostrar lo que necesitamos saber sobre estos componentes, en un próximo articulo veremos un ejemplo simple del como usarlos, donde las diferencias mencionadas serán mas notorias ;)

## Componentes Java Swing

### Que son Los Componentes Graficos?

Como se ha mencionado, los componentes gráficos son estos elementos que permiten brindar una interacción con el usuario del sistema..... Cada componente corresponde a una clase en Java, por esta razón cuando desarrollamos y queremos vincular uno de estos elementos simplemente instanciamos la clase que necesitamos, es decir, si queremos un Área de texto debemos crear un objeto de la clase JTextArea....

### Categorías...

En la introducción sobre Swing vimos un pequeño árbol de herencia, sin embargo este no enmarca todos los componentes Gráficos de la librería sino solo algunos de los principales, a continuación vamos a ampliar el numero de esos componentes agrupándolos en categorías dependiendo de su funcionalidad....

## Contenedores

Como vimos en entradas anteriores, un contenedor es el tapiz donde pintaremos nuestros componentes graficos, existen contenedores principales, entre estos se encuentran JFrame y JDialog pero también existen otros contenedores incluidos dentro de los mencionados...

- **JFrame** - Es la Ventana de aplicación, el contenedor principal

- **JDialog** – Una ventana de tipo Ventana de diálogo, también puede ser un contenedor principal.
- **JPanel** – Permite la creación de paneles independientes donde se almacenan otros componentes.
- **JScrollPane** – permite la vinculación de barras de desplazamiento en un contenedor.
- **JSplitPane** – permite la creación de un contenedor dividido en 2 secciones.
- **JTabbedPane** – Permite la creación de pestañas, cada pestaña representa un contenedor independiente.
- **JDesktopPane** – Permite crear ventanas dentro de una ventana principal
- **JToolBar** – Permite introducir una Barra de herramientas

## Componentes Atómicos

Los componentes atómicos son los elementos que no pueden almacenar otros objetos o componentes gráficos, por ejemplo, un JPanel no es Atómico, ya que en él podemos almacenar JButtons, JTextField entre otros...

- **JLabel** – Permite Vincular Etiquetas, tanto de texto como de imágenes
- **JButton** – Permite vincular Botones simples.
- **JCheckBox** – Son Casilla de verificación, ideal para selección múltiples.
- **JRadioButton** – Permite presentar opciones de selección similares a las checkbox, solo que el enfoque de estas es de única selección.
- **JToggleButton** – Botón que al oprimirlo se quedará presionado hasta que se ejecute otro evento.
- **JComboBox** – Permite mostrar una lista de elementos como un combo de selección.
- **JScrollBar** – Permite mostrar una barra de desplazamiento, regularmente usada en Áreas de texto o paneles donde el contenido es mayor que el tamaño del componente.
- **JSeparator** – Permite separar opciones, es una barra simple.
- **JSlider** - Permite vincular un Deslizador en nuestra ventana.
- **JSpinner** – permite vincular una caja de texto con botones integrados para seleccionar algún valor.
- **JProgressBar** – Establece una barra de progreso.

## Componentes de Texto.



Son todos aquellos que nos permiten procesar cadenas de texto, sea como entrada o salida de información.

- **TextField** – Permite introducir un campo de texto simple.
- **FormattedTextField** – Permite introducir un campo de texto con formato, (si definimos que solo recibe números no permitirá letras...)
- **PasswordField** – Campo de texto que oculta los caracteres ingresados.
- **TextArea** – Permite vincular un área de texto donde el usuario ingresara información o simplemente para presentar cadenas de texto.
- **EditorPane** – Permite vincular un área de texto con propiedades de formato.
- **TextPane** – Similar al anterior, permitiendo otras opciones de formato, colores, iconos entre otros.

## Componentes de Menus.

Estos componentes permiten vincular opciones de menú en nuestras ventanas, tipo menú principal, como por ejemplo el conocido Inicio, Archivo, Edición etc..

- **MenuBar** – Permite vincular una barra de menús.
- **Menu**– Permite vincular botones o enlaces que al ser pulsados despliegan un menú principal.
- **MenuItem** – Botón u opción que se encuentra en un menú.
- **CheckboxMenuItem**– Elemento del menú como opciones de checkbox.
- **RadioButtonMenuItem**– Elemento del menú como botón de selección.
- **PopupMenu**– Opciones de menú emergentes.

## Componentes Complejos

Estos son componentes un poco mas avanzados, cumplen con funciones mas enfocadas a procesos especificos y complejos, como por ejemplo obtener gran cantidad de información de una base de datos, trabajo con nodos, colores entre otros.

- **Table** – Permite vincular una tabla de datos con sus respectivas filas y columnas.
- **Tree** – Carga un árbol donde se establece cierta jerarquía visual, tipo directorio.
- **List** – Permite cargar una lista de elementos, dependiendo de las propiedades puede tenerse una lista de selección múltiple.

- **JFileChooser\_** - Es un componente que permite la búsqueda y selección de ficheros entre otras.
- **JColorChooser\_** - Componente que permite cargar un panel selector de color
- **JOptionPane** - No es algo complejo sino mas un componente independiente que permite mostrar un cuadro de diálogo personalizable.

### Componentes Atómicos Java Swing

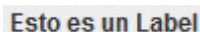
## Que Son?

Como se ha mencionado, Los componentes atómicos son los elementos que no pueden almacenar otros objetos o componentes gráficos, podríamos relacionarlos como componentes simples, pues su función esta bien definida en lo que ellos deben hacer...

## Cuales Son?

Veamos una pequeña descripción y la forma de crear los elementos de esta categoría, la idea es mostrar la forma básica de instanciarlos, dejaré también un enlace al Api de Java de cada componente donde se pueden evidenciar todos los metodos y documentación asociada, realmente es muy recomendable entenderla...

### Jlabel.



Son etiquetas de texto, sin embargo podemos usar sus propiedades para vincular imágenes por lo regular las utilizamos para títulos, nombres o información puntual que queremos mostrar.

```
1 JLabel miLabel;  
2 miLabel= new JLabel();  
3 miLabel.setText("Esto es un Label");
```

### JButton.



Esta clase permite la creación de botones simples, es uno de los elementos mas comunes y usados en las GUI's, trabajan gracias a eventos que se deben implementar a las clases que los usen, igual que los JLabels, pueden vincular imágenes o iconos.

```
1 JButton miBoton;
```

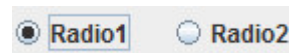
### JCheckBox



Son Casilla de verificación permite al usuario seleccionar una o mas de las opciones propuestas, ideales en aplicaciones con preguntas de selección múltiple con múltiple respuestas.

```
1 JCheckBox miCheckbox;  
2 miCheckbox = new JCheckBox();  
3 miCheckbox.setText("Check1");
```

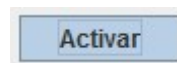
### JRadioButton



Permite presentar opciones de selección similares a las checkbox, solo que el enfoque de estas es de única selección, para trabajar con los RadioButtons se debe hacer uso de un ButtonGroup para determinar la selección única, ideales en aplicaciones con preguntas de selección múltiple con única respuesta.

```
1 JRadioButton miRadioButton;  
2 miRadioButton = new JRadioButton();  
3 miRadioButton.setText("Radio1");
```

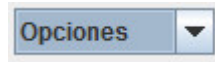
### JToggleButton



Esta clase provee un botón que al oprimirlo se quedará presionado hasta que se oprima nuevamente, ideal para aplicaciones donde se quiera simular un botón de activación, tipo interruptor.

```
1 JToggleButton miToggleButton;  
2 miToggleButton = new JToggleButton();  
3 miToggleButton.setText("Activar");
```

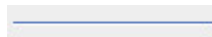
### JcomboBox



Clase que permite mostrar una lista de elementos como un combo de selección, ideal para gran cantidad de opciones de selección única.

```
1 JComboBox miCombo;  
2 miCombo = new JComboBox();  
3 miCombo.addItem("Opciones");  
4 miCombo.addItem("Opcion1");  
5 miCombo.addItem("Opcion2");  
6 miCombo.addItem("Opcion3");  
7 miCombo.addItem("Opcion4");
```

### JSeparator



Esta clase permite dibujar una barra simple en la ventana (o simplemente un raya), se puede crear de forma horizontal o vertical, por lo regular es usada como separador de items en una barra de menú (Archivo|Edición|Ver|Insertar...)

```
1 JSeparator separadorHorizontal;  
2 separadorHorizontal = new JSeparator();  
3 separadorHorizontal.setBounds(430, 92, 100, 5);
```

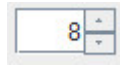
### JSlider



Permite vincular un Deslizador en nuestra ventana, un JSlider es una barra deslizador que permite al usuario definir un valor entre un mínimo o máximo definido con solo arrastrarlo.

```
1JSlider miDeslizado;
2miDeslizador = new JSlider(JSlider.HORIZONTAL, 0, 100, 30);
3miDeslizador.setBounds(430, 140, 100, 30);
4miDeslizador.setValue(0);
```

## JSpinner



Esta clase permite vincular una caja de texto con botones integrados para seleccionar algún valor específico, recorriendo los valores del rango definido.

```
1JSpinner miSpinner;
2miSpinner = new JSpinner();
```

## JProgressBar



Esta clase permite crear una barra de progreso en nuestra aplicación, dicha barra define de forma gráfica el porcentaje de avance de un proceso cualquiera, por lo regular es usada en el trabajo con hilos o temporizadores.

```
1JProgressBar miBarra;
2miBarra = new JProgressBar();
3miBarra.setBounds(450, 180, 110, 20);
```

## El Ejemplo.

Ejemplo de uso



El ejemplo fue creado todo desde cero (a pedal), todos los componentes son creados directamente desde Eclipse y no se utiliza un layout por defecto (mas adelante trabajaremos los layouts, básicamente estos definen como se muestran los componentes en la ventana), por esta razón los objetos se crean con coordenadas definidas gracias al método **setBounds(int x, int y, int a, int b);** donde "x" y "y" definen la posición, "a" y "b" definen el ancho y alto...

Se pueden evidenciar los eventos para los componentes que ejecutan alguna acción, para esto se implementan las interfaces necesarias (Ojo, son interfaces no GUI's) usando implements...

```
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JProgressBar;
import javax.swing.JRadioButton;
import javax.swing.JSeparator;
import javax.swing.JSlider;
import javax.swing.JSpinner;
import javax.swing.JToggleButton;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;

public class VentanaPrincipal extends JFrame implements
ActionListener,ChangeListener {

    private Container contenedor;/**declaramos el contenedor*/
    JLabel labelTitulo;/**declaramos el objeto Label*/
    /**declaramos los objetos JLabels*/
    JLabel etiquetaLabel;
    JLabel etiquetaBoton;
    JLabel etiquetaToogleButton;
    JLabel etiquetaCheckbox;
    JLabel etiquetaRadioButton;
    JLabel etiquetaCombo;
    JLabel etiquetaSeparator;
    JLabel etiquetaSpinner;
    JLabel etiquetaDeslizador;
    JLabel etiquetaBarra;
    JButton boton;/**declaramos el objeto Boton*/
    JCheckBox checkbox1,checkboxbox2;/**declaramos los objetos checkbox*/
    ButtonGroup grupoRadios;/**declaramos un grupo de radioButtons*/
```

```
JRadioButton radio1,radio2;/**declaramos los objetos radioButtons*/
JToggleButton toggleButton;/**declaramos el objeto ToggleButton*/
JComboBox combo;/**declaramos el objeto Combo*/
JSeparator separadorVertical, separadorHorizontal;/**declaramos los
    separadores*/
JSpinner spinner;/**declaramos el objeto spinner*/
JSlider deslizador;/**declaramos el objeto deslizador*/
JProgressBar barra;/**declaramos el objeto barra*/

public VentanaPrincipal(){
    /*permite iniciar las propiedades de los componentes*/
    iniciarComponentes();
    /*Asigna un titulo a la barra de titulo*/
    setTitle("CoDejaVu : JFrame Componentes Atomicos");
    /*tamaño de la ventana*/
    setSize(630, 250);
    /*pone la ventana en el Centro de la pantalla*/
    setLocationRelativeTo(null);
    setResizable(false);
}

private void iniciarComponentes() {
    contenedor=getContentPane();/*instanciamos el contenedor*/
    /*con esto definimos nosotros mismos los tamaños y posicion
    * de los componentes*/
    contenedor.setLayout(null);

    /*Definimos las propiedades de los componentes*/
    labelTitulo= new JLabel();
    labelTitulo.setText("Componentes Atomicos");
    labelTitulo.setFont(new java.awt.Font("Tahoma", 1, 20));
    labelTitulo.setBounds(180, 5, 380, 40);

    etiquetaLabel= new JLabel();
    etiquetaLabel.setText("JLabel :      Esto es un Label o Etiqueta");
    etiquetaLabel.setBounds(20, 50, 280, 23);

    etiquetaBoton= new JLabel();
    etiquetaBoton.setText("JButton : ");
    etiquetaBoton.setBounds(20, 80, 80, 23);

    boton= new JButton();
    boton.setText("Boton");
    boton.setBounds(80, 80, 80, 23);
    boton.addActionListener(this);

    etiquetaCheckbox= new JLabel();
    etiquetaCheckbox.setText("JCheckBox : ");
    etiquetaCheckbox.setBounds(20, 110, 80, 23);

    checkbox1 = new JCheckBox();
    checkbox1.setText("Check1");
    checkbox1.setBounds(100, 110, 80, 23);

    checkbox2 = new JCheckBox();
    checkbox2.setText("Check2");
    checkbox2.setBounds(180, 110, 80, 23);

    etiquetaRadioButton= new JLabel();
```

```
etiquetaRadioButton.setText("JRadioButton : ");
etiquetaRadioButton.setBounds(20, 140, 100, 23);

grupoRadios = new ButtonGroup();
radio1 = new JRadioButton();
radio1.setText("Radio1");
radio1.setBounds(110, 140, 80, 23);

radio2 = new JRadioButton();
radio2.setText("Radio2");
radio2.setBounds(190, 140, 80, 23);

grupoRadios.add(radio1);
grupoRadios.add(radio2);

etiquetaToggleButton= new JLabel();
etiquetaToggleButton.setText("JToggleButton : ");
etiquetaToggleButton.setBounds(20, 180, 100, 23);

toggleButton = new JToggleButton();
toggleButton.setText("Activar");
toggleButton.setBounds(120, 180, 80, 23);

etiquetaCombo= new JLabel();
etiquetaCombo.setText("JComboBox : ");
etiquetaCombo.setBounds(350, 50, 100, 23);

combo = new JComboBox();
combo.addItem("Opciones");
combo.addItem("Opcion1");
combo.addItem("Opcion2");
combo.addItem("Opcion3");
combo.addItem("Opcion4");
combo.setBounds(430, 50, 100, 23);
combo.setSelectedIndex(0);

separadorVertical = new JSeparator();

separadorVertical.setOrientation(javax.swing.SwingConstants.VERTICAL);
separadorVertical.setBounds(300, 60, 10, 200);

etiquetaSeparator= new JLabel();
etiquetaSeparator.setText("JSeparator : ");
etiquetaSeparator.setBounds(350, 80, 100, 23);

separadorHorizontal = new JSeparator();
separadorHorizontal.setBounds(430, 92, 100, 5);

etiquetaSpinner= new JLabel();
etiquetaSpinner.setText("JSpinner : ");
etiquetaSpinner.setBounds(350, 110, 100, 23);

spinner = new JSpinner();
spinner.setBounds(430, 110, 50, 23);
spinner.addChangeListener(this);

etiquetaDeslizador= new JLabel();
etiquetaDeslizador.setText("JSlider : ");
etiquetaDeslizador.setBounds(350, 140, 100, 23);

deslizador = new JSlider(JSlider.HORIZONTAL, 0, 100, 30);
```



```

deslizador.setBounds(430, 140, 100, 30);
deslizador.setPaintTicks(true);
deslizador.setMajorTickSpacing(50);
deslizador.setMinorTickSpacing(5);
deslizador.setBorder(new TitledBorder(""));
deslizador.setValue(0);
deslizador.addChangeListener(this);

etiquetaBarra= new JLabel();
etiquetaBarra.setText("JProgressBar : ");
etiquetaBarra.setBounds(350, 180, 100, 23);

barra = new JProgressBar();
barra.setBounds(450, 180, 110, 20);

/*Agregamos los componentes al Contenedor*/
contenedor.add(labelTitulo);contenedor.add(etiquetaLabel);
contenedor.add(etiquetaBoton);
contenedor.add(etiquetaCheckbox);
contenedor.add(checkbox1);
contenedor.add(checkbox2);
contenedor.add(etiquetaRadioButton);
contenedor.add(radio1);
contenedor.add(radio2);
contenedor.add(etiquetaToogleButton);
contenedor.add(toggleButton);
contenedor.add(etiquetaCombo);
contenedor.add(separadorVertical);
contenedor.add(etiquetaSeparator);
contenedor.add(separadorHorizontal);
contenedor.add(etiquetaSpinner);
contenedor.add(spinner);
contenedor.add(etiquetaDeslizador);
contenedor.add(deslizador);
contenedor.add(etiquetaBarra);
contenedor.add(barra);
contenedor.add(combo);
contenedor.add(boton);
}

/**Agregamos los eventos de presionar*/
@Override
public void actionPerformed(ActionEvent evento) {
    /**Evento cuando presiona el boton*/
    if (evento.getSource()==boton)
    {
        String salida="";
        salida=validaEventos();
        JOptionPane.showMessageDialog(null, salida);
    }
}

/**permite validar cuando se selecciona un check
 * un radioButton, una opción del combo o se
 * presiona el ToogleButton y se */
private String validaEventos() {
    String cad="Selecciona : \n";
    if (checkboxbox1.isSelected()) {
        cad+="check1\n";
    }
}

```

```

        if (checkbox2.isSelected()) {
            cad+="check2\n";
        }
        if (radio1.isSelected()) {
            cad+="radio1\n";
        }
        if (radio2.isSelected()) {
            cad+="radio2\n";
        }
        if (toggleButton.isSelected()) {
            cad+="Toogle Activo\n";
        }else{
            cad+="Toogle InActivo\n";
        }
        cad+=combo.getSelectedItem()+"\n";
        return cad;
    }

    /**Permite definir los eventos del deslizador y el spinner*/
    @Override
    public void stateChanged(ChangeEvent evento) {
        int valor;
        if (evento.getSource() == deslizador) {
            valor = deslizador.getValue();
            barra.setValue(valor);
            spinner.setValue(valor);
        }

        if (evento.getSource() == spinner) {
            valor = (Integer) spinner.getValue();
            deslizador.setValue(valor);
            barra.setValue(valor);
        }
    }
}

```

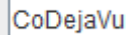
### Componentes De Texto Java Swing

Los componentes de Texto son los que nos permiten procesar datos de tipo cadena, sea como entrada o salida de información, todos los sistemas necesitan procesar datos, tener un mecanismo de entrada y salida disponible para el usuario, este tipo de componentes son obligados en casi todos los desarrollos...

### Cuales Son?

Existen diferentes componentes y formas de procesar texto, podemos hacerlo por consola o por medio de ventanas de Dialogo... veamos los que nos provee java swing...

#### **JtextField.**



Es uno de los componentes mas comunes, permite introducir un campo de texto simple en nuestra ventana, ideal para ingresar o mostrar datos puntuales en un formulario.

```
1 cajaDeTexto = new JTextField();
2 cajaDeTexto.setText("CoDejaVu");
3 cajaDeTexto.setBounds(90, 60, 90, 23);
```

### JformattedTextField.



Permite introducir un campo de texto con formato, (si definimos que solo recibe números no permitirá letras, para esto se requiere definir la mascara a utilizar...) es muy útil al momento de hacer validaciones en nuestros formularios, tambien muy comun al trabajar con fechas.

```
1 /**Creamos la mascara con la que trabajaremos*/
2 mascara= new MaskFormatter("#####");
3 /**Le enviamos la mascara a el componente*/
4 cajaDeTextoConFormato= new JFormattedTextField(mascara);
5 /**Importante definir el foco del componente, para que almacene el valor*/
6 cajaDeTextoConFormato.setFocusLostBehaviorcajaDeTextoConFormato.COMMIT);
7 cajaDeTextoConFormato.setBounds(330, 60, 80, 23);
```

### JpasswordField



Al verlo es un campo simple, sin embargo es un campo de texto especial que oculta los caracteres ingresados, su uso se centra en ventanas de login o ingreso de contraseñas.

```
1 campoContraseña = new JPasswordField();
2 campoContraseña.setBounds(530, 60, 80, 23);
```

### JTextArea

```
JTextField = CoDejaVu
JFormattedTextField = 12345678
JPasswordField = Qaz158*&
JPasswordField Encriptado= [C@b8deef
check1 seleccionado
```

Permite vincular un área de texto donde el usuario ingresará información o simplemente para presentar cadenas de texto, obviamente permite procesar mucha mas cantidad de información que los componentes anteriores.

```
1areaDeTexto = new JTextArea();
2areaDeTexto.setText(CadenaConElTexto);
3areaDeTexto.setBounds(90, 90, 520, 103);
```

### JEditorPane

**Texto En Negrilla y etiqueta H1**

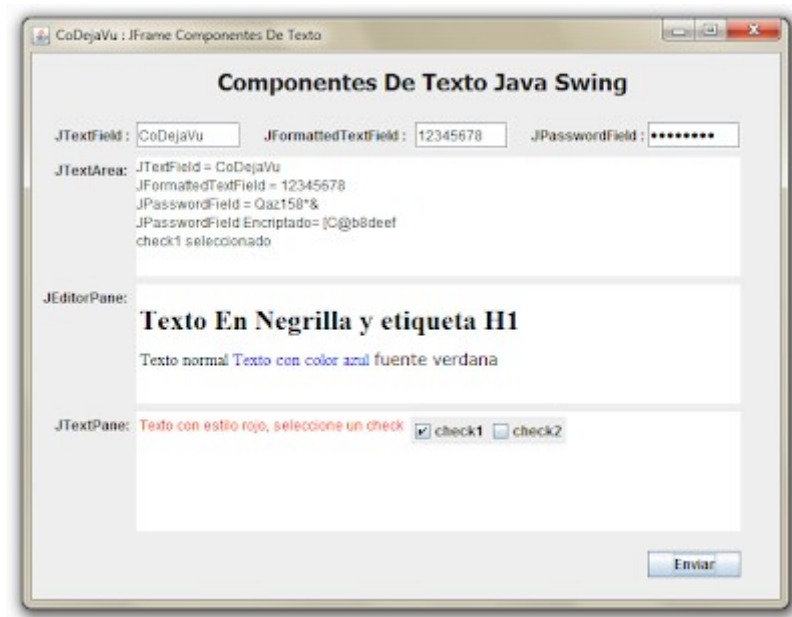
Texto normal Texto con color azul fuente verdana

Es un componente similar al anterior, sin embargo permite vincular un área de texto con propiedades de formato, es decir, por ejemplo, podemos darle formato HTML a nuestro texto usando etiquetas, modificando el tamaño, color y hasta vinculando imagenes....

```
1areaEditorPane = new JEditorPane();
2areaEditorPane.setBounds(90, 200, 520, 103);
3/**Definimos el tipo de texto que utiliza*/
4areaEditorPane.setContentType("text/html");
5areaEditorPane.setText(CadenaConTextoHTML);
```

### El Ejemplo.

Ejemplo de uso:



```
import java.awt.Color;
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.text.ParseException;
import javax.swing.ButtonGroup;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JEditorPane;
import javax.swing.JFormattedTextField;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPasswordField;
import javax.swing.JProgressBar;
import javax.swing.JRadioButton;
import javax.swing.JSeparator;
import javax.swing.JSlider;
import javax.swing.JSpinner;
import javax.swing.JTextArea;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.JToggleButton;
import javax.swing.border.TitledBorder;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import javax.swing.text.BadLocationException;
import javax.swing.text.DefaultStyledDocument;
import javax.swing.text.MaskFormatter;
import javax.swing.text.SimpleAttributeSet;
import javax.swing.text.Style;
import javax.swing.text.StyleConstants;
import javax.swing.text.StyleContext;

public class VentanaPrincipal extends JFrame implements ActionListener {
```

```
private Container contenedor;/**declaramos el contenedor*/
JLabel labelTitulo;/**declaramos el objeto Label*/
/**declaramos los objetos JLabels*/
JLabel etiquetaTextField;
JLabel etiquetaFormattedTextField;
JLabel etiquetaCampoContraseña;
JLabel etiquetaRadioButton;
JLabel etiquetaCombo;
JLabel etiquetaJTextArea;
JLabel etiquetaJEditorPane;
JLabel etiquetaJTextPane;

JTextField cajaDeTexto;
JFormattedTextField cajaDeTextoConFormato;
MaskFormatter mascara;
JPasswordField campoContraseña;

JTextArea areaDeTexto;
JEditorPane areaEditorPane;
JTextPane areaTextPane;
JCheckBox check1, check2;

JButton boton;

public VentanaPrincipal(){
    /*permite iniciar las propiedades de los componentes*/
    iniciarComponentes();
    /*Asigna un titulo a la barra de titulo*/
    setTitle("CoDejaVu : JFrame Componentes De Texto");
    /*tamaño de la ventana (x,y)*/
    setSize(660, 510);
    /*pone la ventana en el Centro de la pantalla*/
    setLocationRelativeTo(null);
    /*impide que la ventana cambie de tamaño*/
    setResizable(false);
}

private void iniciarComponentes() {
    contenedor=getContentPane();/*instanciamos el contenedor*/
    /*con esto definimos nosotros mismos los tamaños y posicion
    * de los componentes*/
    contenedor.setLayout(null);

    /*Definimos las propiedades de los componentes*/
    labelTitulo= new JLabel();
    labelTitulo.setText("Componentes De Texto Java Swing");
    labelTitulo.setFont(new java.awt.Font("Tahoma", 1, 20));
    labelTitulo.setBounds(160, 5, 380, 40);

    etiquetaTextField= new JLabel();
    etiquetaTextField.setText("JTextField : ");
    etiquetaTextField.setBounds(20, 60, 280, 23);

    cajaDeTexto = new JTextField();
    cajaDeTexto.setBounds(90, 60, 90, 23);

    etiquetaFormattedTextField= new JLabel();
    etiquetaFormattedTextField.setText("JFormattedTextField : ");
    etiquetaFormattedTextField.setBounds(200, 60, 140, 23);
```

```

try {
    mascara= new MaskFormatter("#####");
    cajaDeTextoConFormato= new JFormattedTextField(mascara);
    /*Importante definir el foco del componente, para que almacene
el valor*/

    cajaDeTextoConFormato.setFocusLostBehavior(cajaDeTextoConFormato.COMMIT);
    cajaDeTextoConFormato.setBounds(330, 60, 80, 23);

} catch (ParseException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}

etiquetaCampoContraseña= new JLabel();
etiquetaCampoContraseña.setText("JPasswordField : ");
etiquetaCampoContraseña.setBounds(430, 60, 120, 23);

campoContraseña = new JPasswordField();
campoContraseña.setBounds(530, 60, 80, 23);

etiquetaJTextArea= new JLabel();
etiquetaJTextArea.setText("JTextArea: ");
etiquetaJTextArea.setBounds(20, 90, 100, 23);

areaDeTexto = new JTextArea();
areaDeTexto.setBounds(90, 90, 520, 103);

etiquetaJEditorPane= new JLabel();
etiquetaJEditorPane.setText("JEditorPane: ");
etiquetaJEditorPane.setBounds(10, 200, 100, 23);

areaEditorPane = new JEditorPane();
areaEditorPane.setBounds(90, 200, 520, 103);
/*Definimos el tipo de texto que utiliza*/
areaEditorPane.setContentType("text/html");
areaEditorPane.setText("<h1><b>Texto En Negrilla y etiqueta
H1</b></h1>" +
                                " Texto normal" +
                                "<font color=\"blue\"> Texto con color
azul</font>" +
                                "<font face=\"verdana\"> fuente
verdana</font>");

etiquetaJTextPane= new JLabel();
etiquetaJTextPane.setText("JTextPane: ");
etiquetaJTextPane.setBounds(20, 310, 100, 23);

/*usamos StyleContext para definir el estilo a usar*/
StyleContext estilo = new StyleContext();
/*creamos el estilo, no definimos nombre ni estilo padre...*/
Style estiloRojo = estilo.addStyle(null, null);
StyleConstants.setForeground( estiloRojo, Color.red );
/*definimos el estilo a usar*/
DefaultStyledDocument estiloPorDefecto = new
DefaultStyledDocument( estilo );
areaTextPane = new JTextPane(estiloPorDefecto);
areaTextPane.setCharacterAttributes(estiloRojo, false);
areaTextPane.setBounds(90, 310, 520, 103);

```

```
// Se inserta
try {
    areaTextPane.getStyledDocument().insertString(
        areaTextPane.getStyledDocument().getLength(),
        "Texto con estilo rojo, seleccione un check ", estiloRojo);
    } catch (BadLocationException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }

    /*Definimos el Foco del componente, para que se inserte de ultimo en
    el area*/

    areaTextPane.setCaretPosition(areaTextPane.getStyledDocument().getLength());
    check1 = new JCheckBox("check1");
    areaTextPane.insertComponent(check1);

    areaTextPane.setCaretPosition(areaTextPane.getStyledDocument().getLength());
    check2 = new JCheckBox("check2");
    areaTextPane.insertComponent(check2);

    boton = new JButton();
    boton.setText("Enviar");
    boton.setBounds(530, 430, 80, 23);
    boton.addActionListener(this);

    /*Agregamos los componentes al Contenedor*/
    contenedor.add(labelTitulo);
    contenedor.add(etiquetaTextField);
    contenedor.add(etiquetaFormattedTextField);
    contenedor.add(etiquetaJTextArea);

    contenedor.add(boton);
    contenedor.add(cajaDeTexto);
    contenedor.add(cajaDeTextoConFormato);
    contenedor.add(etiquetaCampoContraseña);
    contenedor.add(campoContraseña);
    contenedor.add(areaDeTexto);
    contenedor.add(etiquetaJEditorPane);
    contenedor.add(areaEditorPane);
    contenedor.add(etiquetaJTextPane);
    contenedor.add(areaTextPane);

}

/**Agregamos los eventos de presionar*/
@Override
public void actionPerformed(ActionEvent evento) {
    String retorno="";

    retorno+="JTextField = "+cajaDeTexto.getText()+"\n";
    retorno+="JFormattedTextField = "+cajaDeTextoConFormato.getText()+"\n";

    /*getText() es deprecated, sin embargo podemos usarlo */
    retorno+="JPasswordField = "+campoContraseña.getText()+"\n";
}
```



```

        /*getpassword permite obtener el valor encriptado, para
desencriptarlo se maneja
        * con tipos de datos char*/
        retorno+="JPasswordField Encriptado= "+campoContraseña.getPassword()
+"\\n";

        /*Evento cuando presiona el boton*/
        if (evento.getSource()==boton) {
            if (check1.isSelected()) {
                retorno+="check1 seleccionado\\n";
            }
            if (check2.isSelected()) {
                retorno+="check2 seleccionado\\n";
            }

            JOptionPane.showMessageDialog(null, retorno);
            /*Enviamos el valor de retorno al JTextArea*/
            areaDeTexto.setText(retorno);
        }
    }
}

```

### Contenedores Java Swing

Los contenedores son componentes que permiten almacenar, alojar o contener otros elementos gráficos.....nuevamente mencionamos que es el Tapiz donde vamos a pintar.....

### Cuales Son?

Java Swing provee algunos contenedores útiles para diferentes casos, así cuando desarrollamos una Ventana podemos decidir de que manera presentar nuestros elementos, como serán alojados y de que forma serán presentados al usuario.....veamos....

#### JFrame



Este contenedor es uno de los principales y mas usados (ya lo hemos visto anteriormente), representa la ventana Principal de nuestra aplicación, en el podemos alojar otros contenedores.

#### JDialog



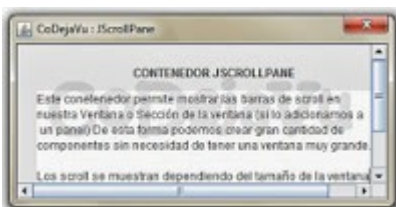
Este contenedor representa una ventana de tipo Ventana de diálogo, también puede ser un contenedor principal aunque es mas recomendable dadas sus propiedades, que sea usada como ventana secundaria, es decir, un **JFrame** como ventana Principal y el resto de ventanas como un **JDialog**.

### JPanel



Este contenedor es uno de los mas simples, permite la creación de paneles independientes donde se almacenan otros componentes, de esta manera decidimos que elementos se alojan en que paneles y dado el caso podemos usar sus propiedades para ocultar, mover o delimitar secciones... cuando alojamos elementos en un panel, los cambios mencionados se aplican a todo su conjunto...es decir, si nuestro panel tiene 5 botones y ocultamos solo el panel, los botones también se ocultan....

### JScrollPane



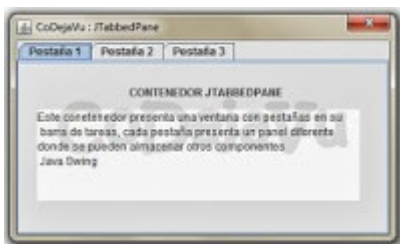
Este contenedor permite vincular barras de scroll o desplazamiento en nuestra aplicación, puede ser utilizado tanto en paneles como en otros componentes como un **JTextArea**, hay que tener en cuenta que no es simplemente poner un scroll, es alojar el componente (en este caso panel o área de texto) en el **JScrollPane**....

### JsplitPane



Este componente permite la creación de un contenedor dividido en 2 secciones, muchas veces usado en aplicaciones donde una sección presenta una lista de propiedades y otra sección presenta el elemento al que le aplicamos dicha lista....cada sección puede ser manipulada por aparte y redimensionar sus componentes (Mas utilizado cuando se trabaja con layouts...).

### JtabbedPane



Este tal vez sea otro de los componentes mas usados, permite la creación de una pestañas en nuestra ventana, cada pestaña representa un contenedor independiente donde podemos alojar paneles u otros elementos.

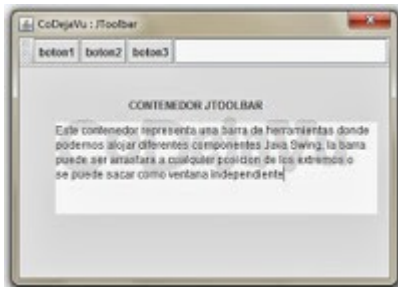
### JdesktopPane



Este contenedor aloja componentes de tipo **JInternalFrame**, estos representan ventanas internas, permitiendo así crear ventanas dentro de una ventana principal, al momento de su creación podemos manipular sus propiedades para definir si queremos redimensionarlas, cerrarlas, ocultarlas entre otras....

También podemos definir una posición inicial de cada ventana interna, sin embargo después de presentadas podemos moverlas por toda la ventana Principal donde se encuentran alojadas.

### JToolBar



Este contenedor representa una Barra de herramientas dentro de nuestra aplicación, en el podemos alojar diferentes componentes que consideremos útiles, botones, check, radios, campos entre otros.....esta barra de herramientas puede ser manipulada permitiendo cambiar su ubicación con tan solo arrastrarla al extremo que queramos, o sacarla de la ventana para que nuestras opciones se encuentren como una ventana independiente.

## El Ejemplo.

Ejemplo de uso



```
import java.awt.Container;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JMenu;
import javax.swing.JMenuBar;
import javax.swing.JMenuItem;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JSplitPane;
import javax.swing.JTabbedPane;
import javax.swing.JToolBar;

public class VentanaPrincipal extends JFrame implements ActionListener {

    private Container contenedor; /*declaramos el contenedor*/
    /*declaramos los objetos Boton*/
    JButton
    botonJFrame, botonJDialog, botonJPanel, botonJScrollPane, botonJSplitPane, botonJTabb
    edPane, botonJDesktopPane, botonJToolBar;
    JLabel labelTitulo; /*declaramos el objeto Label*/
    private VentanaPrincipal miVentanaPrincipal;
    private JMenuBar barraMenu;
    private JMenu menuAcercaDe;
    /*items del menu Acerca De...*/
    private JMenuItem menuItemAplicacion, menuItemBlog;
    String informacionAplicacion="";
    String informacionCoDejaVu="";

    public VentanaPrincipal(){

        /*Inicializamos el Mensaje del menu Acerca de...*/
        informacionAplicacion="Esta es una aplicacion simple con el fin de
exponer \n";
        informacionAplicacion+="de forma sencilla los diferentes
contenedores \n";
        informacionAplicacion+="Java Swing.\n\n";
        informacionAplicacion+="Autor: Cristian David Henao H.\n\n\n";

        informacionCoDejaVu="CoDejaVu es un blog personal sobre lecciones
Aprendidas\n";
        informacionCoDejaVu+="en torno a Ingenieria de Software, presentando
conceptos\n";
        informacionCoDejaVu+="basicos y ejemplos sencillos de programación.\n
n\n";
        informacionCoDejaVu+="codejavu.blogspot.com\n\n\n";

        /**/
        /*permite iniciar las propiedades de los componentes*/
        iniciarComponentes();
        /*Asigna un titulo a la barra de titulo*/
        setTitle("CoDejaVu : JFrame VentanaPrincipal");
        /*tamaño de la ventana*/
```

```

setSize(740,300);
/*pone la ventana en el Centro de la pantalla*/
setLocationRelativeTo(null);
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

/**
 * @param miVentana
 * Enviamos una instancia de la ventana principal
 */
public void setVentanaPrincipal(VentanaPrincipal miVentana) {
    miVentanaPrincipal=miVentana;
}

private void iniciarComponentes() {
    contenedor=getContentPane();/*instanciamos el contenedor*/
    /*con esto definimos nosotros mismos los tamaños y posicion
    * de los componentes*/
    contenedor.setLayout(null);
    barraMenu = new JMenuBar();
    menuAcercaDe = new JMenu();
    menuItemAplicacion = new JMenuItem();
    menuItemBlog = new JMenuItem();

    /*Crea los items del menu Acerca De...*/
    menuItemAplicacion.setText("Aplicación");
    menuAcercaDe.add(menuItemAplicacion);
    menuItemAplicacion.addActionListener(this);

    menuItemBlog.setText("CoDejaVu...");
    menuAcercaDe.add(menuItemBlog);
    menuItemBlog.addActionListener(this);

    menuAcercaDe.setText("Acerca de...");
    barraMenu.add(menuAcercaDe);

    setJMenuBar(barraMenu);

    /*Propiedades del boton, lo instanciamos, posicionamos y
    * activamos los eventos*/

    botonJFrame= new JButton();
    botonJFrame.setText("Ejemplo JFrame");
    botonJFrame.setBounds(40, 80, 200, 25);
    botonJFrame.addActionListener(this);

    botonJDialog= new JButton();
    botonJDialog.setText("Ejemplo JDialog");
    botonJDialog.setBounds(260, 80, 200, 25);
    botonJDialog.addActionListener(this);

    botonJPanel= new JButton();
    botonJPanel.setText("Ejemplo JPanel");
    botonJPanel.setBounds(480, 80, 200, 25);
    botonJPanel.addActionListener(this);

    botonJScrollPane= new JButton();
    botonJScrollPane.setText("Ejemplo JScrollPane");
    botonJScrollPane.setBounds(40, 120, 200, 25);
    botonJScrollPane.addActionListener(this);

```

```

    botonJSplitPane= new JButton();
    botonJSplitPane.setText("Ejemplo JSplitPane");
    botonJSplitPane.setBounds(260, 120, 200, 25);
    botonJSplitPane.addActionListener(this);

    botonJTabbedPane= new JButton();
    botonJTabbedPane.setText("Ejemplo JTabbedPane");
    botonJTabbedPane.setBounds(480, 120, 200, 25);
    botonJTabbedPane.addActionListener(this);

    botonJDesktopPane= new JButton();
    botonJDesktopPane.setText("Ejemplo JDesktopPane");
    botonJDesktopPane.setBounds(150, 160, 200, 25);
    botonJDesktopPane.addActionListener(this);

    botonJToolBar= new JButton();
    botonJToolBar.setText("Ejemplo JToolBar");
    botonJToolBar.setBounds(370, 160, 200, 25);
    botonJToolBar.addActionListener(this);
    //////////////////////////////////////

    /*Propiedades del Label, lo instanciamos, posicionamos y
    * activamos los eventos*/
    labelTitulo= new JLabel();
    labelTitulo.setFont(new java.awt.Font("Tahoma", 0, 28));

    labelTitulo.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
    labelTitulo.setText("CONTENEDORES JAVA SWING");

    labelTitulo.setBorder(javax.swing.BorderFactory.createBevelBorder(javax.swing.bo
rder.BevelBorder.LOWERED));
    labelTitulo.setBounds(110, 10, 500, 40);

    /*Agregamos los componentes al Contenedor*/
    contenedor.add(labelTitulo);
    contenedor.add(botonJFrame);
    contenedor.add(botonJDialog);
    contenedor.add(botonJPanel);
    contenedor.add(botonJScrollPane);
    contenedor.add(botonJSplitPane);
    contenedor.add(botonJTabbedPane);
    contenedor.add(botonJDesktopPane);
    contenedor.add(botonJToolBar);
}

/*Agregamos el evento al momento de llamar la otra ventana*/
@Override
public void actionPerformed(ActionEvent evento) {
    if (evento.getSource()==botonJFrame)
    {
        ClaseFrame miClaseJFrame= new ClaseFrame();
        miClaseJFrame.setVisible(true);
    }
    if (evento.getSource()==botonJDialog)
    {
        ClaseJDialog miClaseJDialog=new
ClaseJDialog(miVentanaPrincipal,true);
        miClaseJDialog.setVisible(true);
    }
    if (evento.getSource()==botonJPanel)

```

```

        {
            ClaseJPanel miClaseJPanel= new
ClaseJPanel(miVentanaPrincipal,true);
            miClaseJPanel.setVisible(true);
        }
        if (evento.getSource()==botonJScrollPane)
        {
            ClaseJScrollPane miClaseJScrollPane = new
ClaseJScrollPane(miVentanaPrincipal,true);
            miClaseJScrollPane.setVisible(true);
        }
        if (evento.getSource()==botonJTabbedPane)
        {
            ClaseJTabbedPane miClaseJTabbedPane = new
ClaseJTabbedPane(miVentanaPrincipal,true);
            miClaseJTabbedPane.setVisible(true);
        }
        if (evento.getSource()==botonJDesktopPane)
        {
            ClaseJDesktopPane miClaseJDesktopPane = new
ClaseJDesktopPane(miVentanaPrincipal,true);
            miClaseJDesktopPane.setVisible(true);
        }
        if (evento.getSource()==botonJToolBar)
        {
            ClaseJToolBar miClaseJToolBar = new
ClaseJToolBar(miVentanaPrincipal,true);
            miClaseJToolBar.setVisible(true);
        }
        if (evento.getSource()==botonJSplitPane)
        {
            ClaseJSplitPane miClaseJSplitPane = new
ClaseJSplitPane(miVentanaPrincipal,true);
            miClaseJSplitPane.setVisible(true);
        }
        /*Desde aqui tambien trabajamos algunos eventos simples*/
        if (evento.getSource()==menuItemAplicacion) {
            JOptionPane.showMessageDialog(null,
informacionAplicacion,"INFORMACION",JOptionPane.INFORMATION_MESSAGE);
        }

        if (evento.getSource()==menuItemBlog) {
            JOptionPane.showMessageDialog(null,
informacionCoDejaVu,"CoDejaVu!!!",JOptionPane.WARNING_MESSAGE);
        }
    }
}

```