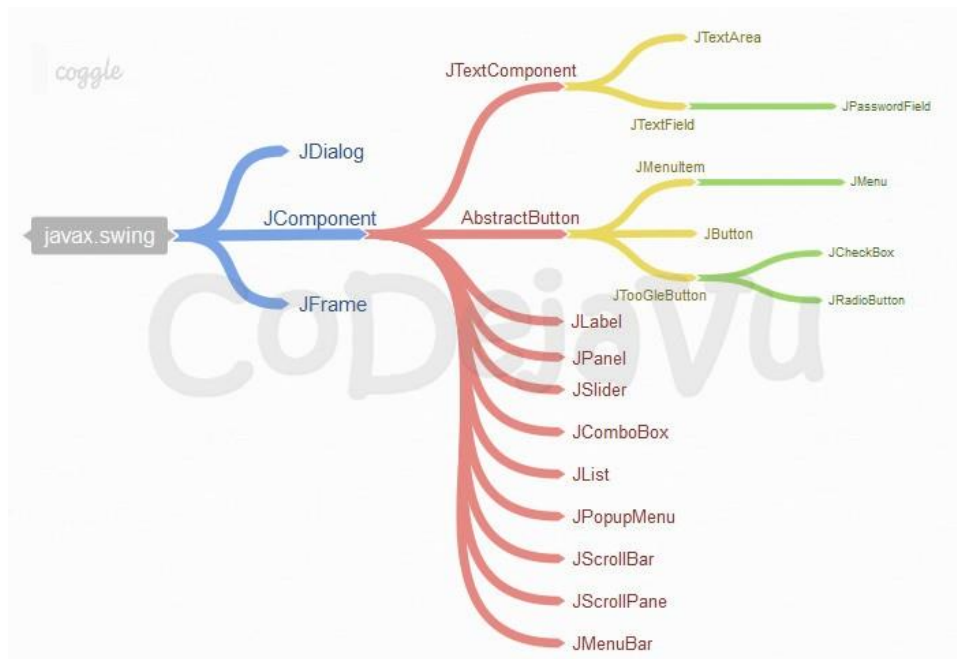


## Componentes Swing



### Componentes Atómicos Java Swing

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.

#### JLabel.

**Esto es un Label** 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 ([Api de Java](#)).

```
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 ([Api de Java](#)).

```
1 JButton miBoton;  
2 miBoton= new JButton();  
3 miBoton.setText("Boton");
```

#### 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 multiple respuestas ([Api de Java](#)).

```
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 ([Api de Java](#)).

```
1 ButtonGroup grupoRadios = new ButtonGroup();
```

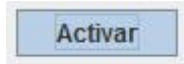
```
2 JRadioButton miRadioButton;
```

```
3 miRadioButton = new RadioButton();
```

```
4 miRadioButton.setText("Radio1");
```

```
5 grupoRadios.add(miRadioButton);
```

### 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 ([Api de Java](#)).

```
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 ([Api de Java](#)).

```
1 String[] opciones = {"Opcion1", "Opcion2", "Opcion3", "Opcion4", "Opcion5"};
```

```
2 JComboBox miCombo;
```

```
3 miCombo = new JComboBox(opciones);
```

### 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...) ([Api de Java](#)).

```
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 deslizadora que permite al usuario definir un valor entre un mínimo o máximo definido con solo arrastrarlo ([Api de Java](#)).

```
1 JSlider miDeslizado;
```

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

```
3 miDeslizador.setBounds(430, 140, 100, 30);
```

```
4 miDeslizador.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 ([Api de Java](#)).

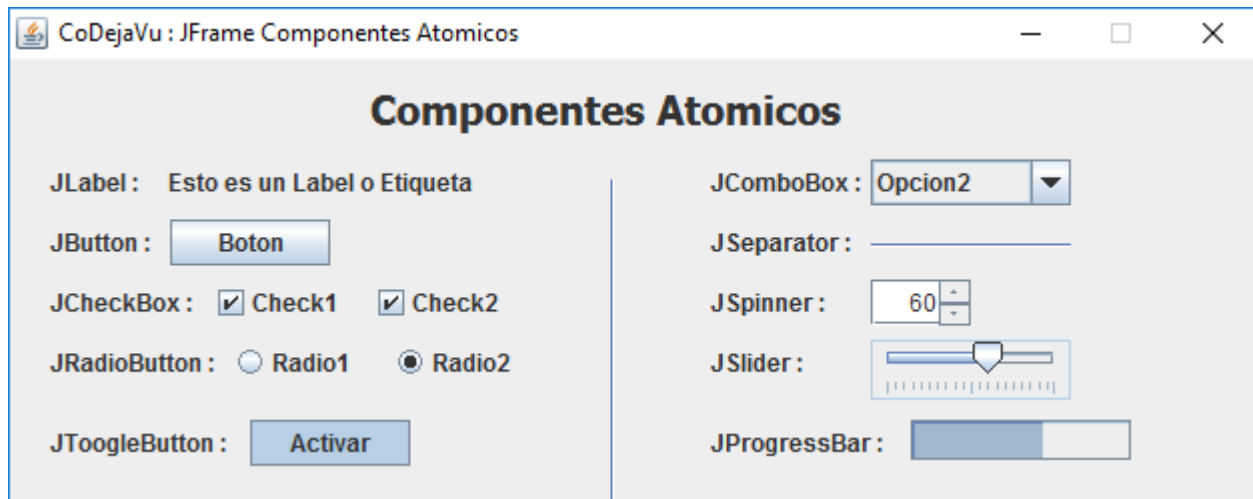
```
1 JSpinner miSpinner;  
2 miSpinner = new JSpinner();
```

## 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 ([Api de Java](#)).

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



java.awt

### Class Container

java.lang.Object  
java.awt.Component  
java.awt.Container

#### All Implemented Interfaces:

ImageObserver, MenuContainer, Serializable

#### Direct Known Subclasses:

BasicSplitPaneDivider, CellRendererPane, DefaultTreeCellEditor.EditorContainer, JComponent, Panel, ScrollPane, Window

```
public class Container  
extends Component
```

A generic Abstract Window Toolkit(AWT) container object is a component that can contain other AWT components.

Components added to a container are tracked in a list. The order of the list will define the components' front-to-back stacking order within the container. If no index is specified when adding a component to a container, it will be added to the end of the list (and hence to the bottom of the stacking order).

**Note:** For details on the focus subsystem, see *How to Use the Focus Subsystem*, a section in *The Java Tutorial*, and the *Focus Specification* for more information.

Since:

JDK1.0

See Also:

`add(java.awt.Component, int)`, `getComponent(int)`, `LayoutManager`, `Serialized Form`

javax.swing

### Class JFrame

## getContentPane

```
public Container getContentPane()
```

Returns the `contentPane` object for this frame.

Specified by:

`getContentPane` in interface `RootPaneContainer`

Returns:

the `contentPane` property

See Also:

```
setContentPane(java.awt.Container), RootPaneContainer.getContentPane()
```

javax.swing

## Class JPanel

```
java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.JPanel
```



javax.swing.event

## Class ChangeEvent

```
java.lang.Object
  java.util.EventObject
    javax.swing.event.ChangeEvent
```

All Implemented Interfaces:

Serializable

```
public class ChangeEvent
extends EventObject
```

`ChangeEvent` is used to notify interested parties that state has changed in the event source.

**Warning:** Serialized objects of this class will not be compatible with future Swing releases. The current serialization support is appropriate for short term storage or RMI between applications running the same version of Swing. As of 1.4, support for long term storage of all JavaBeans™ has been added to the `java.beans` package. Please see `XMLEncoder`.

EventoCambio se utiliza para notificar a las partes interesadas que el estado ha cambiado en el evento origen.

## Constructors

### Constructor and Description

**`ChangeEvent(Object source)`**  
Constructs a `ChangeEvent` object.

## Method Summary

### Methods inherited from class java.util.EventObject

`getSource`, `toString`

### Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

## Interface ChangeListener

All Superinterfaces:

`EventListener`

All Known Implementing Classes:

`AbstractButton.ButtonChangeListener`, `BasicButtonListener`, `BasicMenuUI.ChangeHandler`, `BasicProgressBarUI.ChangeHandler`, `BasicScrollBarUI.ModelListener`, `BasicScrollPaneUI.HSBChangeListener`, `BasicScrollPaneUI.ViewportChangeHandler`, `BasicScrollPaneUI.VSBChangeListener`, `BasicSliderUI.ChangeHandler`, `BasicTabbedPaneUI.TabSelectionHandler`, `JCheckBoxMenuItem.AccessibleJCheckBoxMenuItem`, `JMenuItem.AccessibleJMenuItem`, `JMenuItem.AccessibleJMenuItem`, `JRadioButtonMenuItem.AccessibleJRadioButtonMenuItem`, `JScrollPane.AccessibleJScrollPane`, `JSpinner.AccessibleJSpinner`, `JSpinner.DateEditor`, `JSpinner.DefaultEditor`, `JSpinner.ListEditor`, `JSpinner.NumberEditor`, `JTabbedPane.AccessibleJTabbedPane`, `JTabbedPane.ModelListener`, `ProgressMonitor.AccessibleProgressMonitor`

```
public interface ChangeListener
extends EventListener
```

Defines an object which listens for `ChangeEvents`.

## Method Summary

Methods	
Modifier and Type	Method and Description
void	<b><code>stateChanged(ChangeEvent e)</code></b> Invoked when the target of the listener has changed its state.

```

package ComponentesAtomicos.ComponentesAtomicos.src.principal;

public class Principal {

    /**
     * @param args
     */
    public static void main(String[] args) {
        VentanaPrincipal miVentanaPrincipal= new VentanaPrincipal();
        miVentanaPrincipal.setVisible(true);
    }

}

```

```

package ComponentesAtomicos.ComponentesAtomicos.src.principal;

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

/**
 * @author HENAO
 */
public class VentanaPrincipal extends JFrame implements ActionListener,ChangeListener {

    private JPanel contenedor;/**declaramos el contenedor*/
    private JLabel labelTitulo;/**declaramos el objeto JLabel*/
    /**declaramos los objetos JLabels*/
    private JLabel etiquetaLabel;
    private JLabel etiquetaBoton;
    private JLabel etiquetaToggleButton;
    private JLabel etiquetaCheckbox;
    private JLabel etiquetaRadioButton;
    private JLabel etiquetaCombo;
    private JLabel etiquetaSeparator;
    private JLabel etiquetaSpinner;
    private JLabel etiquetaDeslizador;
    private JLabel etiquetaBarra;
    private JButton boton;/**declaramos el objeto Boton*/
    private JCheckBox checkbox1,checkbox2;/**declaramos los objetos checkbox*/
    private ButtonGroup grupoRadios;/**declaramos un grupo de radioButtons*/
    private JRadioButton radio1,radio2;/**declaramos los objetos radioButtons*/
    private JToggleButton toggleButton;/**declaramos el objeto ToggleButton*/
    private JComboBox combo;/**declaramos el objeto Combo*/
    private JSeparator separadorVertical, separadorHorizontal;/**declaramos los
separadores*/
    private JSpinner spinner;/**declaramos el objeto spinner*/
    private JSlider deslizador;/**declaramos el objeto deslizador*/
    private 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=new JPanel();/*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);

String[] opciones = {"Opciones", "Opcion1", "Opcion2", "Opcion3", "Opcion4",
"Opcion5"};

combo = new JComboBox(opciones);
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(etiquetaToggleButton);
        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);
        add(contenedor);
    }

    /**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 (checkbox1.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);
        }
    }

```



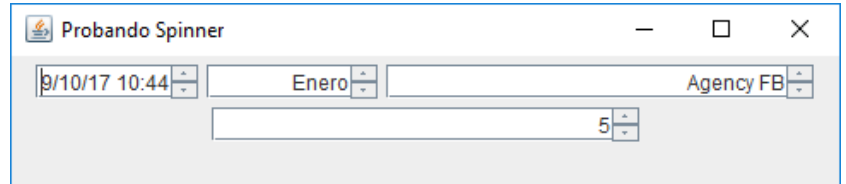
```
        spinner.setValue(valor);
    }

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

```

package graficos;
import javax.swing.*.*;
import java.awt.*.*;
import java.awt.event.*;

```



```

public class PruebaSpinner {

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        MarcoSpinner miMarco=new MarcoSpinner();
        miMarco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        miMarco.setVisible(true);
    }
}

class MarcoSpinner extends JFrame{
    public MarcoSpinner(){
        setTitle("Probando Spinner");
        setBounds(500,300,500,350);
        add(new laminaSpinner());
    }
}

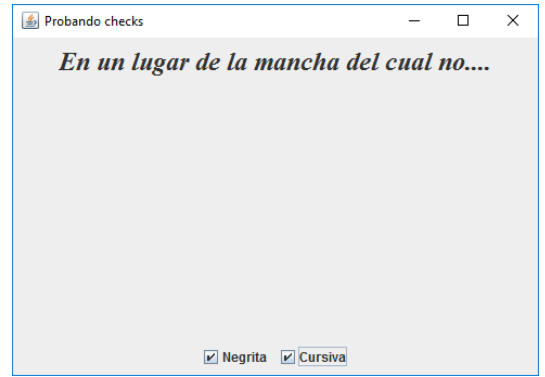
class laminaSpinner extends JPanel{
    public laminaSpinner(){
        //por defecto
        //JSpinner control=new JSpinner();
        //Sineer de fechas
        JSpinner control=new JSpinner(new SpinnerDateModel());
        //Spinner con listas
        String lista[]={"Enero","Febrero","Marzo"};
        JSpinner control1=new JSpinner(new SpinnerListModel(lista));
        //para cambiar el tamaño del spinner
        Dimension d=new Dimension(100,20);
        Dimension d2=new Dimension(250,20);
        Dimension d3=new Dimension(50,20);
        control1.setPreferredSize(d);
        //Todas las fuentes que estan disponibles en nuestro equipo
        String
        listaF[]=GraphicsEnvironment.getLocalGraphicsEnvironment().getAvailableFontFamilyNames();
        JSpinner control2=new JSpinner(new SpinnerListModel(listaF));
        control2.setPreferredSize(d2);
        //acotar los valores numericos
        //empieza en 5, min 0, max 10 y aumenta en 1
        JSpinner control3=new JSpinner(new SpinnerNumberModel(5,0,10,1));
        control3.setPreferredSize(d2);
        add(control);
        add(control1);
        add(control2);
        add(control3);
    }
}

```

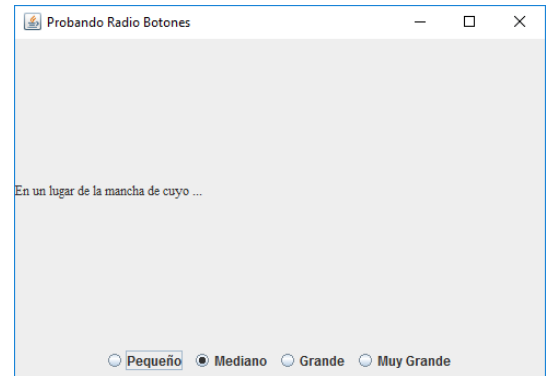
## EJERCICIOS COMPONENTES SWING



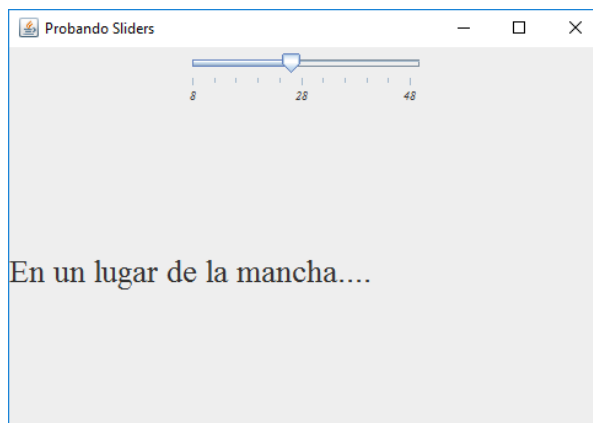
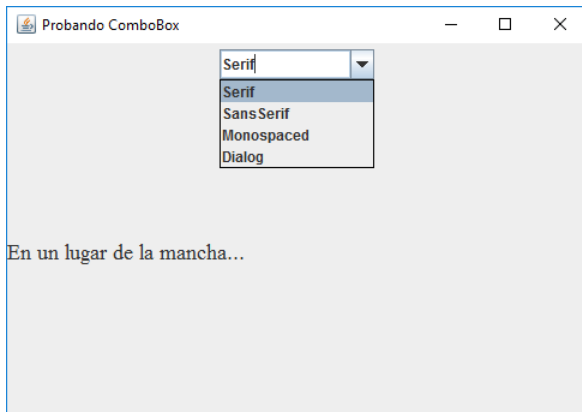
- 1** PruebaChecks.java El texto(Utiliza una JLabel) escrito en el panel se pondra en negrita si pulsamos la casilla de verificación “Negrita” y en cursiva si pulsamos en la casilla “cursiva”.



- 2** EjemploRadio2.java . Por defecto aparece seleccionado el radio boton “Mediano”, ira cambiando el tamaño de la letra según el radio boton que seleccionemos.



- 3** PruebaCombo.java



- 4** PruebaSliders2.java

## 5. VariosSpinners.java

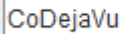
Crea un programa en el que incluyas 5 tipos de Spinners y una barra de progreso. Programa que el Spinner de tipo numérico cuando cambie aumente o disminuya la barra de progreso.

## 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...

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...

### TextField.



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 ([Api de Java](#)).

```
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 ([Api de Java](#)).

```
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);
```

The following table shows the characters that you can use in the formatting mask:

Character	Description
#	Any valid number (Character.isDigit).
' (single quote)	Escape character, used to escape any of the special formatting characters.
U	Any character (Character.isLetter). All lowercase letters are mapped to uppercase.
L	Any character (Character.isLetter). All uppercase letters are mapped to lowercase.
A	Any character or number (Character.isLetter or Character.isDigit).
?	Any character (Character.isLetter).
*	Anything.
H	Any hex character (0-9, a-f or A-F).

## 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 ([Api de Java](#)).

```
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 ([Api de Java](#)).

```
1 areaDeTexto = new JTextArea();
2 areaDeTexto.setText(CadenaConElTexto);
3 areaDeTexto.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... ([Api de Java](#)).

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

## JTextPane

Texto con estilo rojo, seleccione un check ☒ check1 ☐ check2

Es practicamente igual al anterior, básicamente posee

las misma funciones, por lo cual podemos decir que es una mejora del JEditorPane permitiendo otras opciones de formato, colores, iconos, trabajo con estilos, componentes entre otros ([Api de Java](#)).

```
1      /**usamos StyleContext para definir el estilo a usar*/
2      StyleContext estilo = new StyleContext();
3      /**creamos el estilo, no definimos nombre ni estilo padre...*/
4      Style estiloRojo = estilo.addStyle(null, null);
5      StyleConstants.setForeground( estiloRojo, Color.red );
6      /**definimos el estilo a usar*/
7      DefaultStyledDocument estiloPorDefecto=new DefaultStyledDocument(estilo);
8      areaTextPane = new JTextPane(estiloPorDefecto);
9      areaTextPane.setCharacterAttributes(estiloRojo, false);
10     areaTextPane.setBounds(90, 310, 520, 103);
```

```
package ComponentesDeTexto.ComponentesDeTexto.src.principal;
```

```
public class Principal {
```

```
    /**
```

```
     * @param args
```

```
    */
```

```
    public static void main(String[] args) {
        VentanaPrincipal miVentanaPrincipal= new VentanaPrincipal();
        miVentanaPrincipal.setVisible(true);
    }
```

```
}
```

```
package ComponentesDeTexto.ComponentesDeTexto.src.principal;
```

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

```
/**
```

```
 * @author HENAO
```

```
 * Calse que permite presentar la ventana con los componentes de texto
```

```
 * en funcionamiento
```

```
 * codejavu.blogspot.com
```

```
 *
```

```
 */
```

```
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);
        }
    }
}

```



## 6. ComponetesTexto1PracticaPropuesta.java

Los campos deben cumplir las siguientes restrincciones:

- En el campo usuario solo se pueden introducir maximo 6 caracteres alfanumericos.
- En el telefono maximo 9 digitos.
- La fecha de alta debe aparecer por defecto la fecha actual con este formato.
- Al presionar en el boton enviar, se deben copiar todos los datos al area de texto.

Usuario:  Maximo 6 alfanuméricos

Contraseña:

Telefono:  Nueve digitos

Fecha de Alta:

Usuario = isa123  
Contraseña encriptada = [C@3397cee9  
Telefono = 655123456  
Fecha de Alta = 07/10/2017

## Menús Con Java Swing

Java Swing nos provee ciertos componentes para crear una barra de Menú, en ella podemos combinar diferentes elementos con un mismo fin, proveer las opciones necesarias para trabajar con el sistema.

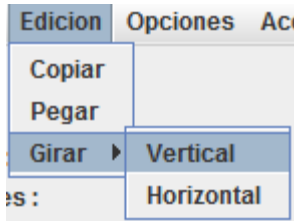
### JMenuBar.



Es el elemento principal cuando vamos a crear menús, ya que provee la barra donde se alojaran cada uno de los items u opciones deseadas.

```
1 barraMenu = new JMenuBar();
2 barraMenu.add(menuArchivo);
3 setJMenuBar(barraMenu);
```

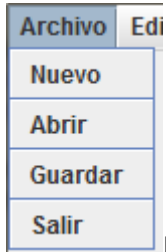
### JMenu.



Si bien el elemento anterior permite crear una Barra donde alojar el resto de componentes, el **JMenu** es quien contiene dichos componentes, a este se le agregan el resto de opciones, podemos asociarlo con un contenedor el cual aloja otros elementos como botones, etiquetas, campos entre otros..... el **JMenu** permite agregar los elementos o items correspondientes, así como otros **JMenus**....

```
1 menuArchivo = new JMenu();
2 menuArchivo.add(menuItemNuevo);
3 menuArchivo.addSeparator();
```

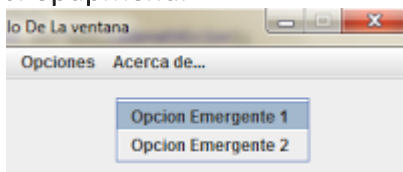
### JMenuItem.



Representan Items u opciones del menú, cuando creamos un **JMenu** decimos que tipo de opciones puede contener y al crear un **JMenuItem** decimos cuales son las opciones para ese menú en especifico, por ejemplo el Menú "Archivo", contendrá los items "Abrir", "Guardar", "Nuevo", "Principal", etc....

```
1 menuItemNuevo = new JMenuItem();
2 menuItemNuevo.setText("Nuevo");
3 menuArchivo.add(menuItemNuevo);
```

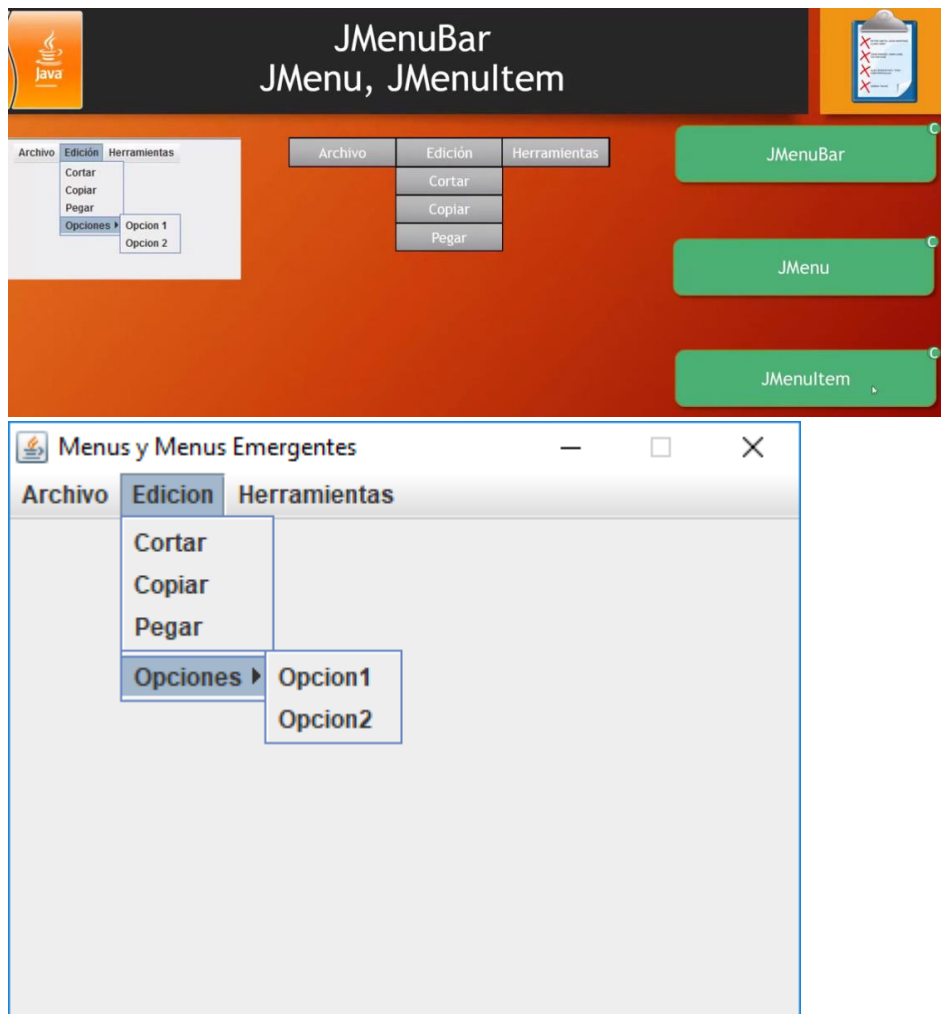
### JPopupMenu.



Por ultimo tenemos el **JPopupMenu**, a diferencia de los anteriores, este componente no es contenido en la Barra de Menú, sino que se asocia al contenedor principal,

permite brindar opciones emergentes o popup con tan solo dar click derecho sobre algún área del panel..... el JPopupMenu funciona también como un contenedor similar al JMenu....

```
1 menuEmergente = new JPopupMenu();
2 itemEmergente1.setText("Opcion Emergente 1");
3 menuEmergente.add(itemEmergente1)
```



```
package pruebaMenu;
import javax.swing.JFrame;
public class Principal {
    public static void main(String[] args) {
        VentanaPrincipal miFrame= new VentanaPrincipal();
        miFrame.setVisible(true);
        miFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
}
```

```
package pruebaMenu;
import javax.swing.*;

public class VentanaPrincipal extends JFrame{
    private JMenuBar miBarra;
    private JMenu archivo, edicion, herramientas, opciones;
    private JMenuItem guardar, guardarComo, cortar, copiar, pegar, opcion1, opcion2,
    generales, itemEmergente1, itemEmergente2;
    private JPopupMenu menuEmergente;
    private JPanel miPanel;
    public VentanaPrincipal() {
        iniciarComponentes();
        //Asigna un titulo a la barra de titulo
    }
}
```

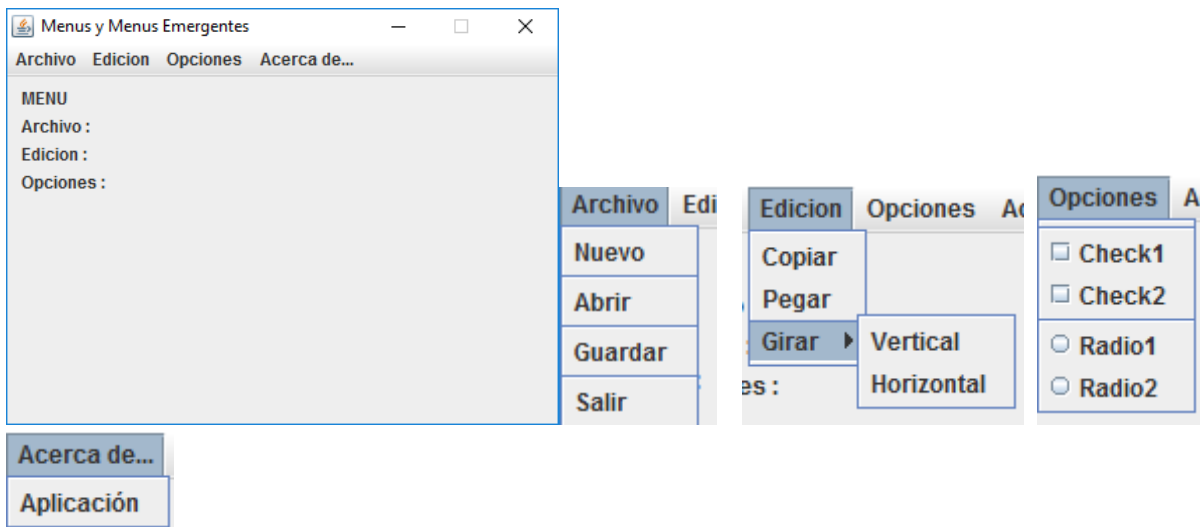
```

setTitle("Menús y Menús Emergentes");
//tamaño de la ventana
setSize(400,300);
//pone la ventana en el Centro de la pantalla
setLocationRelativeTo(null);
//impide que la ventana cambie de tamaño*/
setResizable(false);
}
private void iniciarComponentes() {
    //lo más habitual es colocarla en la parte superior
    //creamos la barra en sí
    miBarra=new JMenuBar();
    //Creamos las opciones de la barra
    archivo=new JMenu("Archivo");
    edicion=new JMenu("Edición");
    herramientas=new JMenu("Herramientas");
    opciones=new JMenu("Opciones");
    //la opciones de las fichas
    guardar=new JMenuItem("Guardar");
    guardarComo=new JMenuItem("Guardar Como");
    cortar=new JMenuItem("Cortar");
    copiar=new JMenuItem("Copiar");
    pegar=new JMenuItem("Pegar");
    opcion1=new JMenuItem("Opcion1");
    opcion2=new JMenuItem("Opcion2");
    generales=new JMenuItem("Generales");
    //agregamos las opciones al menú
    archivo.add(guardar);
    archivo.add(guardarComo);
    edicion.add(cortar);
    edicion.add(copiar);
    edicion.add(pegar);
    //incluimos un separador
    edicion.addSeparator();
    herramientas.add(generales);
    opciones.add(opcion1);
    opciones.add(opcion2);
    //especificamos que opciones no cuelga de la barra general si no de edición
    edicion.add(opciones);
    miBarra.add(archivo);
    miBarra.add(edicion);
    miBarra.add(herramientas);
    //añadimos la barra a la ventana
    setJMenuBar(miBarra);

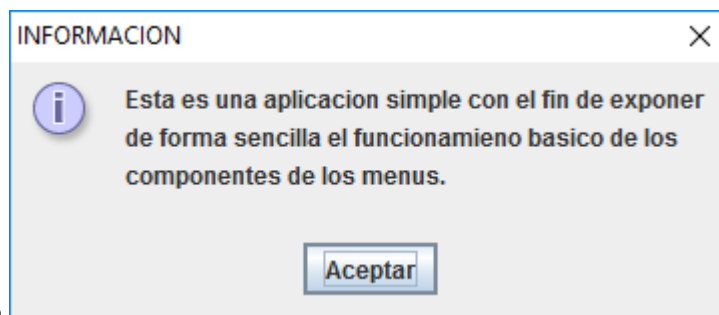
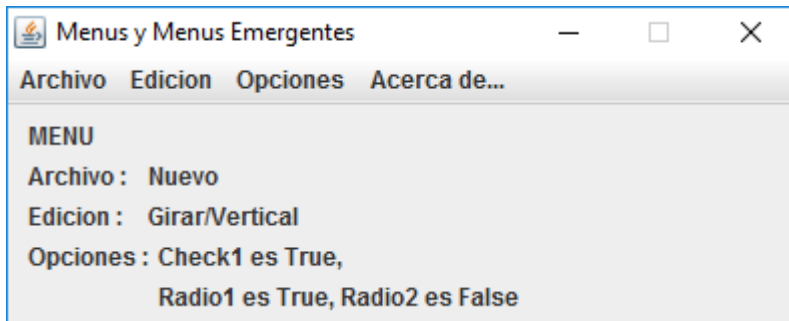
    //las opciones del menú emergente
    miPanel=new JPanel();
    miPanel.setLayout(null);
    itemEmergente1=new JMenuItem("Opción Emergente 1");
    itemEmergente2=new JMenuItem("Opción Emergente 2");
    menuEmergente = new JPopupMenu();
    //añadimos las opciones al menú emergente
    menuEmergente.add(itemEmergente1);
    menuEmergente.add(itemEmergente2);
    //añadimos el menú emergente al panel
    miPanel.setComponentPopupMenu(menuEmergente);
    //añadimos el panel
    add(miPanel);
}
}

```

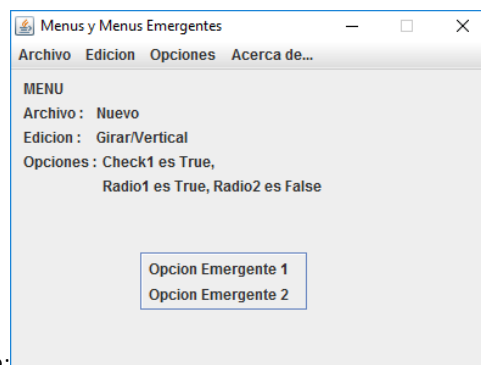
## 7. ComponentesMenu1PracticaPropuesta.java



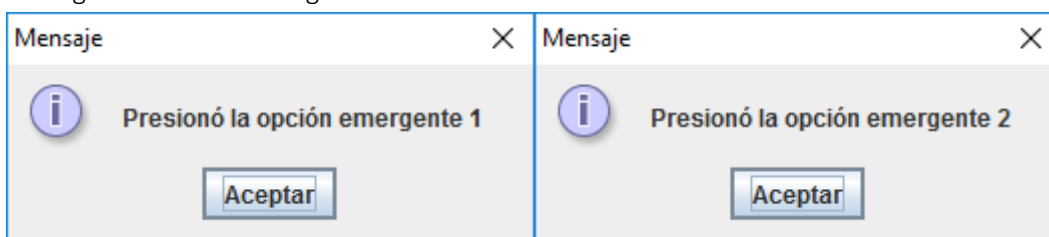
Al Seleccionar cada una de las opciones aparecera su nombre en las JLabel del panel central:



Acerca de... - Aplicación



Y el siguiente menu emergente:



## 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.....

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, 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...después lo veremos).

## 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.

## Ejemplo JFrame y JDialog

(Diálogos modales y no modales tema1, pag25)

([http://chuwiki.chuidiang.org/index.php?title=JFrame\\_y\\_JDialog](http://chuwiki.chuidiang.org/index.php?title=JFrame_y_JDialog))

Este ejemplo será muy básico, simplemente vamos a crear 2 ventanas y mediante un botón estableceremos comunicación entre ellas, tendremos la **VentanaPrincipal** que será un **JFrame** y la **VentanaConfirmacion** que será un **JDialog**...

Vamos a ver de forma simple como relacionar estas ventanas, pero para eso necesitamos como mínimo un botón que permita establecer el evento de llamado.

El ejemplo se divide en 3 clases, la Clase **VentanaPrincipal** que corresponde a un **JFrame**, la Clase **VentanaConfirmacion** que es un formulario y la clase **Principal** que contiene el método main para iniciar el sistema.

### VentanaPrincipal

Esta ventana será el lugar donde llamamos la ventana de confirmación, aquí crearemos un contenedor donde se agregará una etiqueta de texto y un botón para generar el evento.

También cabe resaltar que se envió una instancia de la **VentanaPrincipal** a la **VentanaConfirmacion** esto con el fin de que el **JDialog** sea **hijo** del **JFrame**.

### VentanaConfirmacion.

La **VentanaConfirmacion** tan solo la crearemos para diferenciar nuestra ventana de Dialogo de nuestra ventana principal, extiende de **JDialog** y como se menciona en el punto anterior, vamos a hacer que sea hija del **JFrame**, de esta manera evitamos que se creen muchos objetos **miVentanaConfirmacion**, sino, solo uno dependiente de la principal.

Como vemos el constructor trae 2 parámetros, el Frame padre y la propiedad que determina si es hija o no, con esto vamos a evitar que el usuario ingrese a la ventanaPrincipal y trabaje en ella, bloqueándola solo hasta que se cierre la ventana de dialogo.

### Principal.

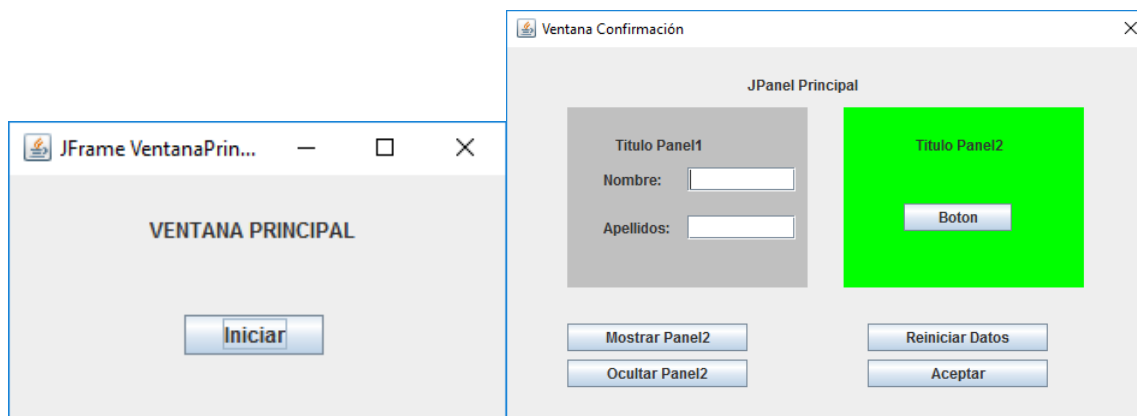
Tenemos una clase **Principal** donde inicia todo el proceso, esta clase contiene el método main que contiene la lógica para la creación y cargue de la ventana principal, como vemos creamos una instancia de la VentanaPrincipal y la enviamos como parámetro, esto para trabajar siempre con la ventanaPrincipal y no crear muchas instancias de la misma, sino trabajar siempre con la original.

Al final solo bastará con ejecutar esta clase y nuestra aplicación iniciará, para esto usamos el método setVisible permitiendo cargar la ventanaPrincipal.

## El Resultado.

Al ejecutar la aplicación vemos que se carga la ventanaPrincipal con los componentes mencionados, si presionamos el botón se carga la ventanaConfirmacion dependiente de la ventanaPrincipal, podemos notar las diferencias entre estas 2 ventanas, así como la propiedad que impide acceder a la principal mientras se encuentra activa la ventana de Dialogo, esto será muy útil cuando queremos hacer ventanas que deberían ser únicas por usuario, por ejemplo ventanas de registro o información, de modo que siempre tengamos una sola por cada evento.

Tenemos nuestra aplicación en funcionamiento, es algo simple pero la base para construir aplicaciones completas.





Principal.java

```
package JFrameJDialog;

public class Principal {
    public static void main(String[] args) {
        /* Declaramos el objeto */
        VentanaPrincipal miVentanaPrincipal;
        /* Instanciamos el objeto */
        miVentanaPrincipal = new VentanaPrincipal();
        //Enviamos el objeto como parametro para que sea unico en toda la aplicación
        miVentanaPrincipal.setVentanaPrincipal(miVentanaPrincipal);
        /* Hacemos que se cargue la ventana */
        miVentanaPrincipal.setVisible(true);
    }
}
```



VentanaPrincipal.java

```
package JFrameJDialog;

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

public class VentanaPrincipal extends JFrame implements ActionListener {
    private JPanel miPanel; /* declaramos el Panel */
    private JButton botonCambiar; /* declaramos el objeto Boton */
    private JLabel labelTitulo; /* declaramos el objeto Label */
    private VentanaPrincipal miVentanaPrincipal;

    public VentanaPrincipal() {
        /* permite iniciar las propiedades de los componentes */
        iniciarComponentes();
        /* Asigna un titulo a la barra de titulo */
        setTitle("JFrame VentanaPrincipal");
        /* tamaño de la ventana */
        setSize(300, 180);
        /* pone la ventana en el Centro de la pantalla */
        setLocationRelativeTo(null);
    }

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

    private void iniciarComponentes() {
        miPanel = new JPanel(); /* instanciamos el Panel */
        /*
         * con esto definimos nosotros mismos los tamaños y posicion de los componentes
         */
        miPanel.setLayout(null);
        /*
         * Propiedades del boton, lo instanciamos, posicionamos y activamos los eventos
         */
    }
}
```

```

        botonCambiar = new JButton();
        botonCambiar.setText("Iniciar");
        botonCambiar.setBounds(100, 80, 80, 23);
        botonCambiar.addActionListener(this);
        /*
         * Propiedades del Label, lo instanciamos, posicionamos y activamos los eventos
         */
        labelTitulo = new JLabel();
        labelTitulo.setText("VENTANA PRINCIPAL");
        labelTitulo.setBounds(80, 20, 180, 23);
        /* Agregamos los componentes al Contenedor */
        miPanel.add(labelTitulo);
        miPanel.add(botonCambiar);
        add(miPanel);

    }

    /* Agregamos el evento al momento de llamar la otra ventana */
    @Override
    public void actionPerformed(ActionEvent evento) {
        if (evento.getSource() == botonCambiar) {
            VentanaConfirmacion miVentanaConfirmacion = new
VentanaConfirmacion(miVentanaPrincipal, true);
            miVentanaConfirmacion.setVisible(true);
        }
    }
}

```

6

```

package JFrame_JDialog.src.ventanas;
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class VentanaPrincipal extends JFrame implements ActionListener {

    private Container contenedor; /*declaramos el contenedor*/
    private JButton botonCambiar; /*declaramos el objeto Boton*/
    private JLabel labelTitulo; /*declaramos el objeto Label*/
    private VentanaPrincipal miVentanaPrincipal;

    public VentanaPrincipal(){
        /*permite iniciar las propiedades de los componentes*/
        iniciarComponentes();
        /*Asigna un titulo a la barra de titulo*/
        setTitle("JFrame VentanaPrincipal");
        /*tamaño de la ventana*/
        setSize(300,180);
        /*pone la ventana en el Centro de la pantalla*/
        setLocationRelativeTo(null);
    }

    /**
     * @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);

        /*Propiedades del boton, lo instanciamos, posicionamos y
        * activamos los eventos*/
        botonCambiar= new JButton();
        botonCambiar.setText("Iniciar");
        botonCambiar.setBounds(100, 80, 80, 23);
        botonCambiar.addActionListener(this);

        /*Propiedades del Label, lo instanciamos, posicionamos y
        * activamos los eventos*/
        labelTitulo= new JLabel();
        labelTitulo.setText("VENTANA PRINCIPAL");
        labelTitulo.setBounds(80, 20, 180, 23);

        /*Agregamos los componentes al Contenedor*/
        contenedor.add(labelTitulo);
        contenedor.add(botonCambiar);
    }

    /*Agregamos el evento al momento de llamar la otra ventana*/
    @Override
    public void actionPerformed(ActionEvent evento) {
        if (evento.getSource()==botonCambiar)
        {
            VentanaConfirmacion miVentanaConfirmacion=new
VentanaConfirmacion(miVentanaPrincipal,true);
            miVentanaConfirmacion.setVisible(true);
        }
    }
}

```



VentanaConfirmacion.java

```

package JFrame_JDialog.src.ventanas;
import java.awt.*;
import java.awt.event.*;
import java.text.ParseException;
import java.util.Calendar;
import java.util.Date;
import javax.swing.*;

public class VentanaConfirmacion extends JDialog implements ActionListener{
    private JLabel labelTitulo, tituloPanel1, tituloPanel2;/*declaramos el objeto Label*/
    private JPanel miPanelPrincipal;/*contenedor de los componentes
    private PanelUno panel1; /*panel de ejemplo
    private JPanel panel2;/*panel de ejemplo
    private JButton miBotonPanel2;
    private JButton botonOcultar, botonMostrar, botonReiniciar, botonAceptar;
    private JTextField nombre, apellidos;
    private JLabel nombrelLabel, apellidosLabel;

    public VentanaConfirmacion(VentanaPrincipal miVentanaPrincipal, boolean
modal)/*constructor
    {
        super(miVentanaPrincipal, modal);
        iniciaComponentes();
    }
}

```

```

//Asigna un titulo a la barra de titulo
setTitle("Ventana Confirmación");
//tamaño de la ventana
setSize(550,350);
//pone la ventana en el Centro de la pantalla
setLocationRelativeTo(null);
}

private void iniciaComponentes() {
    miPanelPrincipal = new JPanel();
    miPanelPrincipal.setLayout(null);

    /*Propiedades del Label, lo instanciamos, posicionamos y
    * activamos los eventos*/
    labelTitulo= new JLabel();
    labelTitulo.setText("JPanel Principal");
    labelTitulo.setBounds(200, 20, 180, 23);

    /*propiedades Panel1*/
    componentesPanel1();

    /*propiedades Panel2*/
    componentesPanel2();

    botonMostrar = new JButton();
    botonMostrar.setText("Mostrar Panel2");
    botonMostrar.setBounds(50, 230, 150, 23);
    botonMostrar.addActionListener(this);

    botonOcultar = new JButton();
    botonOcultar.setText("Ocultar Panel2");
    botonOcultar.setBounds(50, 260, 150, 23);
    botonOcultar.addActionListener(this);

    botonReiniciar = new JButton();
    botonReiniciar.setText("Reiniciar Datos");
    botonReiniciar.setBounds(300, 230, 150, 23);
    botonReiniciar.addActionListener(this);

    botonAceptar = new JButton();
    botonAceptar.setText("Aceptar");
    botonAceptar.setBounds(300, 260, 150, 23);
    botonAceptar.addActionListener(this);

    /*Agregamos los componentes al Contenedor*/
    miPanelPrincipal.add(labelTitulo);
    miPanelPrincipal.add(panel1);
    miPanelPrincipal.add(panel2);
    miPanelPrincipal.add(botonMostrar);
    miPanelPrincipal.add(botonOcultar);
    miPanelPrincipal.add(botonReiniciar);
    miPanelPrincipal.add(botonAceptar);
    add(miPanelPrincipal);
}

class PanelUno extends JPanel{
    public void paintComponent(Graphics g){
        super.paintComponent(g);
        setLayout(null);
        tituloPanel1= new JLabel();
        tituloPanel1.setText("Titulo Panel1");
        tituloPanel1.setBounds(40, 20, 90, 23);
        nombreLabel=new JLabel("Nombre:");

```

```

        nombreLabel.setBounds(30, 50, 80, 20);
        apellidosLabel=new JLabel("Apellidos:");
        apellidosLabel.setBounds(30, 90, 80, 20);
        nombre=new JTextField();
        nombre.setBounds(100, 50, 90, 20);
        apellidos=new JTextField();
        apellidos.setBounds(100, 90, 90, 20);

        add(tituloPanel1);
        add(nombreLabel);
        add(apellidosLabel);
        add(nombre);
        add(apellidos);
    }

    }

    private void componentesPanel1() {
        panel1=new PanelUno();
        panel1.setBounds(50, 50, 200, 150);
        panel1.setBackground(Color.lightGray);
    }

    private void componentesPanel2() {
        panel2 = new JPanel();
        panel2.setLayout(null);//con esto defino que voy a asignar
posiciones de forma manual, sin layout
        panel2.setBounds(280, 50, 200, 150);
        panel2.setBackground(Color.green);

        tituloPanel2= new JLabel();
        tituloPanel2.setText("Titulo Panel2");
        tituloPanel2.setBounds(60, 20, 90, 23);

        miBotonPanel2 = new JButton();
        miBotonPanel2.setText("Boton");
        miBotonPanel2.setBounds(50, 80, 90, 23);

        panel2.add(tituloPanel2);
        panel2.add(miBotonPanel2);
    }

    @Override
    public void actionPerformed(ActionEvent e) {
        if (e.getSource()==botonMostrar)
        {
            panel2.setVisible(true);
        }
        if (e.getSource()==botonOcultar)
        {
            panel2.setVisible(false);
        }
        if (e.getSource()==botonReiniciar)
        {
            reiniciarPanel(panel1);
            nombre.requestFocus();
        }
        if (e.getSource()==botonAceptar)
        {
            comprobarDatos();
        }
    }
    //reiniciar los valores de un panel

```

```

        private void reiniciarPanel (JPanel panel) {
            Component componentes [] = panel.getComponents(); //obtenemos los
            componentes situados dentro del panel
            for (Component c : componentes){ //por cada uno de ellos
                if (c instanceof JTextField){ //vemos si es un jTextField
                    ((JTextField)c).setText(""); //y lo vaciamos
                }
                else if (c instanceof JScrollPane) { //vemos si es un JScrollPane
                    Component csp = ((JScrollPane)c).getViewport().getView();
                    //obtenemos el componente
                    if (csp instanceof JTextArea){ //vemos si es un jTexArea
                        ((JTextArea)csp).setText(""); // y lo vaciamos
                    }
                    else if (c instanceof JFormattedTextField) { //vemos si es un
                        JFormattedTextField
                        ((JFormattedTextField)c).setValue(null); // lo vaciamos
                        try {
                            ((JFormattedTextField)c).commitEdit(); //y actualizarnos su
                            valor
                        } catch (ParseException ex) {
                        }
                    }
                    else if (c instanceof JSpinner) { //vemos si es un JSpinner
                        SpinnerModel modelo = ((JSpinner)c).getModel();
                        if (modelo instanceof SpinnerDateModel) { //si es de tipo fecha
                            Calendar cal = Calendar.getInstance();
                            Date fechaHoy = cal.getTime();
                            modelo.setValue(fechaHoy); //le ponemos la fecha de hoy
                        } else if (modelo instanceof SpinnerNumberModel) { //si es de
                            tipo numérico
                            modelo.setValue(1); //le ponemos el valor más bajo inicial
                            (1)
                        }
                    }
                    else if (c instanceof JComboBox) { //vemos si es un JComboBox
                        ((JComboBox)c).setSelectedIndex(0); //y seleccionamos el primer
                        elemento
                    }
                    else if (c instanceof JCheckBox) { //vemos si es un JCheckBox
                        ((JCheckBox)c).setSelected(false); //y lo desmarcamos
                    }
                }
            }
        } //fin reiniciarPanel

        private void comprobarDatos() {
            boolean ok = !(nombre.getText().equals(""))
                        && !(apellidos.getText().equals(""));

            if (ok) {
                JOptionPane.showMessageDialog(this, "Registro guardado", "Datos del
                Formulario",
                JOptionPane.INFORMATION_MESSAGE);
                reiniciarPanel(panel1);
            } else {
                JOptionPane.showMessageDialog(this, "Faltan datos por
                complementar", "Datos del Formulario",
                JOptionPane.ERROR_MESSAGE);
            }
            nombre.requestFocus();
        } //fin comprobarFormulario
    }

```



# El Ejemplo Contenedores.

<http://codejavu.blogspot.com.es/2013/10/contenedores-java-swing.html>

Nuevamente presento un ejemplo sencillo, no tiene mayor grado de complejidad que el que le pueda dar cada componente, la aplicación presenta una Ventana Principal con un botón por cada elemento a exponer, como se mencionó anteriormente la Ventana Principal será un **JFrame** y el resto de ventanas del sistema lo trabajaremos como **JDialog**.



La idea es mostrar rápidamente el funcionamiento básico de estos elementos, si se necesitan alguno se recomienda profundizar mediante ejemplos o mas investigación..... es importante que si no se entiende algún procedimiento soliciten ayuda, por ejemplo es muy común omitir en ocasiones el método pack(); debemos saber que este permite trabajar con las dimensiones de los contenedores (si es necesario).

- ▼ Contenedores
  - ▼ src
    - ▼ aplicacion
      - > Principal.java
      - > VentanaPrincipal.java
    - ▼ jdesktoppane
      - > ClaseDesktopPane.java
    - ▼ jdialog
      - > ClaseDialog.java
    - ▼ jframe
      - > ClaseFrame.java
    - ▼ jpanel
      - > ClasePanel.java
    - ▼ jscrollpane
      - > ClaseScrollPane.java
    - ▼ jsplitpane
      - > ClaseSplitPane.java
    - ▼ jtabbedpane
      - > ClaseTabbedPane.java
    - ▼ jtoolbar
      - > ClaseToolBar.java

8.



DemoAreaTexto.java

Debe copiar el texto seleccionado del area de la izquierda al area de la derecha cuando se haga un clic sobre el boton copiar.

