Componente JTextField o campo de texto

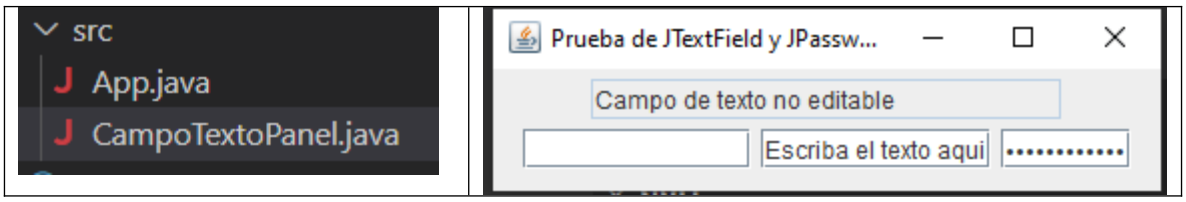
JTextField es un componente liviano que permite la edición de una sola línea de texto.

javax.swing

Class JTextField

java.lang.Object
  java.awt.Component
    java.awt.Container
      javax.swing.JComponent
        javax.swing.text.JTextComponent
          javax.swing.JTextField

Ejemplo 1 JTextField y JPasswordField.




```
// Los componentes JTextField y JPasswordField.
import java.awt.FlowLayout;//administrador de layout
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JPanel;

 public class CampoTextoPanel extends JPanel
 {
   private final JTextField campoTexto1; // campo de texto con tamaño fijo
   private final JTextField campoTexto2; // campo de texto con texto
   private final JTextField campoTexto3; // campo de texto con texto y tamaño
   private final JPasswordField campoContrasenia; // campo de contraseña con texto

   // El constructor de CampoTextoMarco agrega objetos JTextField a JFrame
   public CampoTextoPanel()
   {
      super();
      setLayout(new FlowLayout());

      // construye campo de texto con 10 columnas
      campoTexto1 = new JTextField(10);
      // construye campo de texto con texto predeterminado
```

```
      campoTexto2 = new JTextField("Escriba el texto aqui");
      // construye campo de texto con texto predeterminado y 21 columnas
      campoTexto3 = new JTextField("Campo de texto no editable", 21);
      campoTexto3.setEditable(false); // deshabilita la edición
      // construye campo de contraseña con texto predeterminado
      campoContrasenia = new JPasswordField("Texto oculto");

      //agrego los componentes al panel
      add(campoTexto1); // agrega campoTexto1 a JPanel
      add(campoTexto2); // agrega campoTexto2 a JPanel
      add(campoTexto3); // agrega campoTexto3 a JPanel
      add(campoContrasenia); // agrega campoContrasenia a JPanel

   }
} // fin de la clase CampoTextoMarco
```

```
// Prueba de CampoTextoMarco.
//import javax.swing.JFrame;

import javax.swing.JFrame;

public class App {
   public static void main(String[] args) {
      // crea un panel que contiene nuestro dibujo
      CampoTextoPanel panel = new CampoTextoPanel();
      // crea un nuevo marco para contener el panel
      JFrame aplicacion = new JFrame("Prueba de JTextField y JPasswordField");

      aplicacion.add(panel); // agrega el panel al marc
      aplicacion.setSize(350, 100); // establece el tamaño del marco
      aplicacion.setVisible(true);//hace el marco visible
      // establece el marco para salir cuando se cierre
      aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   }
} // fin de la clase App
```

Ejemplo 2 JTextField y JPasswordField con evento de dar ENTER en JTextField

Un evento en Java (plataforma de programación) es cualquier objeto que se crea cuando algo cambia dentro de la interfaz gráfica de un usuario. Si un usuario hace clic en un botón o en un cuadro combinado, o escribe caracteres en un campo de texto, entre otras cosas, activa un evento y crea el objeto. Este comportamiento es parte del mecanismo de manejo de eventos de la plataforma y se incluye en la biblioteca GUI de Swing (la biblioteca de interfaces gráficas con la que trabaja Java).

Por ejemplo, se tiene un JButton (un botón en Java con el que el usuario puede desencadenar una acción). Si un usuario hace clic en JButton, se activa un evento de clic de botón, el cual se creará y se enviará al oyente de eventos correspondiente (en este caso, ActionListener). El oyente relevante habrá implementado el código que determina la acción a realizar cuando ocurre el evento. Ten en cuenta que la fuente de un evento tiene que estar emparejada con un detector de eventos actualizado, o su activación no resultará en ninguna acción.

Cómo funcionan los eventos

Para manejar los eventos en Java correctamente se deben conocer dos elementos fundamentales: el origen y el oyente del evento. El objeto que se crea cuando ocurre un evento se llama origen del evento. El oyente, por otro lado, es el objeto encargado de recibir los eventos y procesarlos en el momento que ocurren. Es importante destacar que Java facilita varios tipos de orígenes.

Existen también varios tipos de eventos y oyentes en Java. Cada tipo de evento está directamente configurado o vinculado con un oyente específico. Por ejemplo, un tipo común de evento son los eventos de acción, representados por la clase de Java ActionEvent, que son activados cuando el usuario hace clic en un botón o en algún elemento de una lista.

En las acciones del usuario se crea entonces un objeto correspondiente a la clase de ActionEvent, que a su vez corresponde a la acción relevante. En ese momento, este objeto comprende toda la información de origen del evento y la acción específica que ha realizado el usuario. Este objeto de evento transita después al método del objeto del ActionListener correspondiente, es decir, el oyente correspondiente.

Acción vacía

Cuando este procedimiento se ejecuta, se devuelve la respuesta de la GUI adecuada. Puede ser para abrir o cerrar un cuadro de diálogo, hacer una firma digital, descargar archivos o cualquier otra de las muchas acciones que están disponibles para un usuario en una interfaz.

Tipos de eventos

A continuación, se enumeran y explican algunos de los tipos de eventos más comunes en Java:

- ActionEvent: representa a la acción de cuando se hace clic en un elemento gráfico, como un botón o elemento de una lista. Oyente relacionado: ActionListener.
- ContainerEvent: representa a un evento que se produce en el propio contenedor de la GUI, por ejemplo, si un usuario agrega o elimina un objeto de la interfaz. Oyente relacionado: ContainerListener.
- KeyEvent: representa a un evento en el que el usuario presiona, escribe o suelta una tecla. Oyente relacionado: KeyListener.
- WindowEvent: representa a cualquier evento relacionado con una ventana, por ejemplo, cuando una ventana está cerrada y es activada o desactivada. Oyente relacionado: WindowListener.
- MouseEvent: representa cualquier evento relacionado con un mouse, como cuando se hace clic, doble clic, etc. Oyente relacionado: MouseListener.

Es importante tener en cuenta que varios oyentes y fuentes de eventos son capaces de interactuar entre sí. Por ejemplo, un solo oyente puede llegar a registrar varios eventos, si son del mismo tipo. Esto significa que, para un conjunto similar de componentes que realizan el mismo tipo de acción, un detector de eventos podría manejarlos todos. De manera similar, un solo evento se puede vincular a múltiples oyentes, si eso se adapta al diseño del programa, aunque esto es mucho menos común.

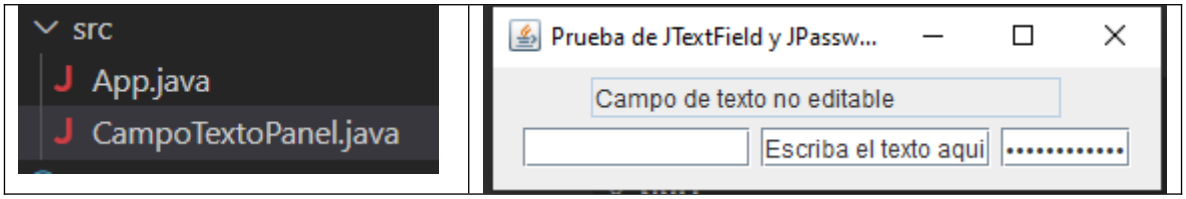
Se usa la class ActionEvent, donde un evento semántico que indica que ocurrió una acción definida por el componente. El evento se pasa a cada objeto ActionListener que se registró para recibir dichos eventos mediante el método addActionListener del componente.

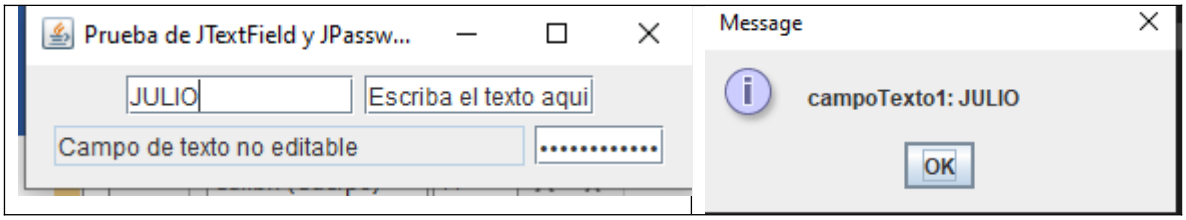
objeto público getSource() El objeto en el que ocurrió inicialmente el Evento.

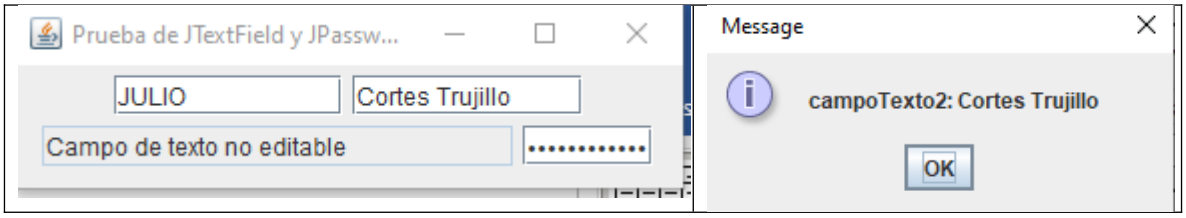
Devoluciones: El objeto en el que ocurrió inicialmente el Evento.

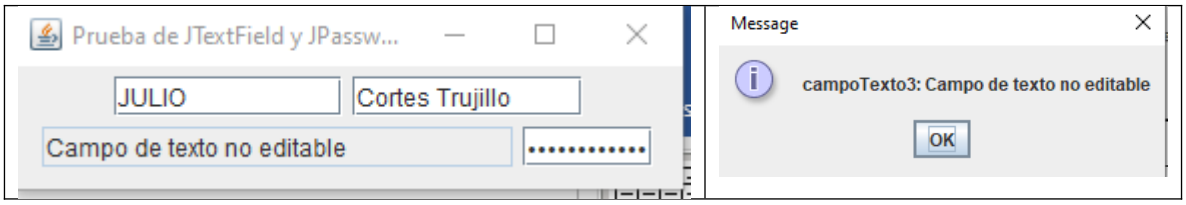
El objeto que implementa la interfaz ActionListener obtiene este ActionEvent cuando ocurre el evento.

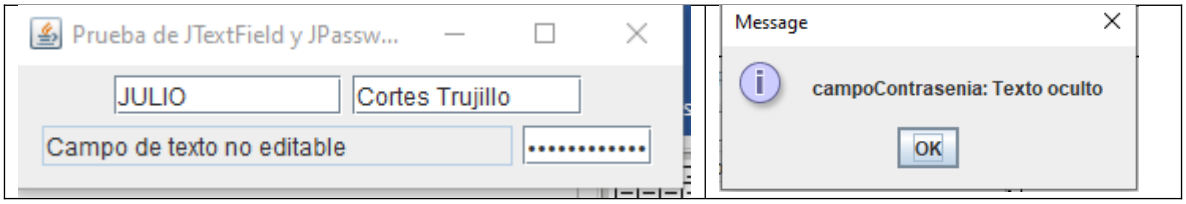
getActionCommand() le da una cadena que representa el comando de acción. El valor es específico del componente; para un JButton, tiene la opción de establecer el valor con setActionCommand (comando de cadena), pero para un JTextField, si no establece esto, automáticamente le dará el valor del campo de texto. Según el javadoc, esto es por compatibilidad con java.awt.TextField.


src
App.java
CampoTextoPanel.java
Prueba de JTextField y JPassw...
Campo de texto no editable
Escriba el texto aqui


Prueba de JTextField y JPassw...
JULIO
Escriba el texto aqui
Campo de texto no editable
Message
campoTexto1: JULIO
OK


Prueba de JTextField y JPassw...
JULIO
Cortes Trujillo
Campo de texto no editable
Message
campoTexto2: Cortes Trujillo
OK


Prueba de JTextField y JPassw...
JULIO
Cortes Trujillo
Campo de texto no editable
Message
campoTexto3: Campo de texto no editable
OK


Prueba de JTextField y JPassw...
JULIO
Cortes Trujillo
Campo de texto no editable
Message
campoContrasenia: Texto oculto
OK

Ejemplo 2 JTextField y JPasswordField con evento de dar ENTER en JTextField

```
// Los componentes JTextField y JPasswordField.
import java.awt.FlowLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JTextField;
import javax.swing.JPasswordField;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

 public class CampoTextoPanel extends JPanel
 {
   private final JTextField campoTexto1; // campo de texto con tamaño fijo
   private final JTextField campoTexto2; // campo de texto con texto
   private final JTextField campoTexto3; // campo de texto con texto y tamaño
   private final JPasswordField campoContrasenia; // campo de contraseña con texto

   // El constructor de CampoTextoMarco agrega objetos JTextField a JFrame
   public CampoTextoPanel()
   {
      super();
      setLayout(new FlowLayout());

      // construye campo de texto con 10 columnas
      campoTexto1 = new JTextField(10);
      // construye campo de texto con texto predeterminado
      campoTexto2 = new JTextField("Escriba el texto aqui");
      // construye campo de texto con texto predeterminado y 21 columnas
      campoTexto3 = new JTextField("Campo de texto no editable", 21);
      campoTexto3.setEditable(false); // deshabilita la edición
      // construye campo de contraseña con texto predeterminado
      campoContrasenia = new JPasswordField("Texto oculto");

      //agregan lo componente al panel
      add(campoTexto1); // agrega campoTexto1 a CampoTextoPanel
      add(campoTexto2); // agrega campoTexto2 a CampoTextoPanel
      add(campoTexto3); // agrega campoTexto3 a CampoTextoPanel
      add(campoContrasenia); // agrega campoContrasenia a CampoTextoPanel

      // registra los manejadores de eventos en este caso dar ENTER
      ManejadorCampoTexto manejador = new ManejadorCampoTexto();
      campoTexto1.addActionListener(manejador);
      campoTexto2.addActionListener(manejador);
      campoTexto3.addActionListener(manejador);
      campoContrasenia.addActionListener(manejador);
```

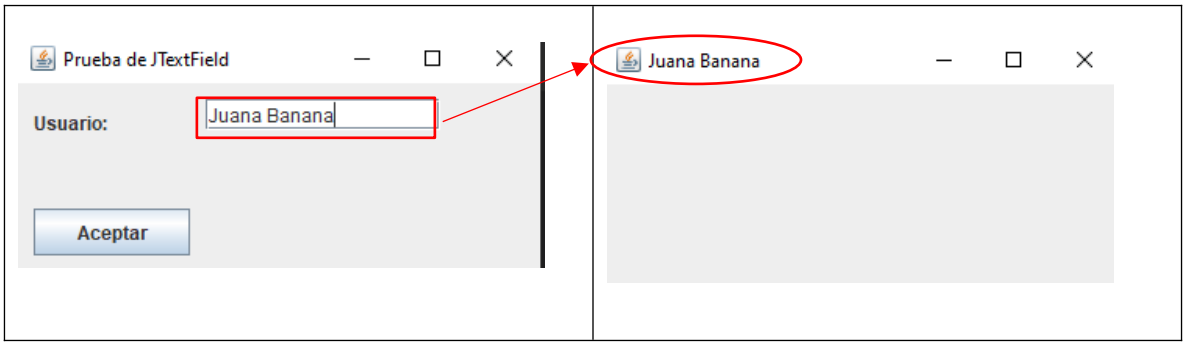
```
   }

   // clase interna privada para el manejo de eventos
   private class ManejadorCampoTexto implements ActionListener
   {
      // procesa los eventos de campo de texto
      @Override
      public void actionPerformed(ActionEvent evento)
      {
         String cadena = "";
         // el usuario oprimió Intro o Enter en el objeto JTextField campoTexto1
         if (evento.getSource() == campoTexto1)
            cadena = String.format("campoTexto1: %s", evento.getActionCommand());

         // el usuario oprimió Intro o Enter en el objeto JTextField campoTexto2
         else if (evento.getSource() == campoTexto2)
            cadena = String.format("campoTexto2: %s",evento.getActionCommand() );

         // el usuario oprimió Intro o Enter en el objeto JTextField campoTexto3
         else if (evento.getSource() == campoTexto3)
            cadena = String.format("campoTexto3: %s",evento.getActionCommand());

         // el usuario oprimió Intro o Enter en el objeto JTextField campoContrasenia
         else if (evento.getSource() == campoContrasenia)
            cadena = String.format("campoContrasenia: %s",evento.getActionCommand());

         // muestra el contenido del objeto JTextField
         JOptionPane.showMessageDialog(null, cadena);
         }
      } // fin de la clase interna privada ManejadorCampoTexto

 } // fin de la clase CampoTextoMarco
```

```
// Prueba de CampoTextoPanel
//import javax.swing.JFrame;

import javax.swing.JFrame;

public class App {
   public static void main(String[] args) {
      // crea un panel que contiene nuestro dibujo
      CampoTextoPanel panel = new CampoTextoPanel();
      // crea un nuevo marco para contener el panel
      JFrame aplicacion = new JFrame("Prueba de JTextField y JPasswordField");

      aplicacion.add(panel); // agrega el panel al marc
      aplicacion.setSize(350, 100); // establece el tamaño del marco
```

```
        aplicacion.setVisible(true);//hace el marco visible
        // establece el marco para salir cuando se cierre
        aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
} // fin de la clase App
```

Ejemplo 3 JLabel, TextField y Button

// Confeccionar un programa que permita ingresar el nombre de usuario y cuando se presione
// un botón mostrar el valor ingresado en la barra de títulos de otro JFrame.



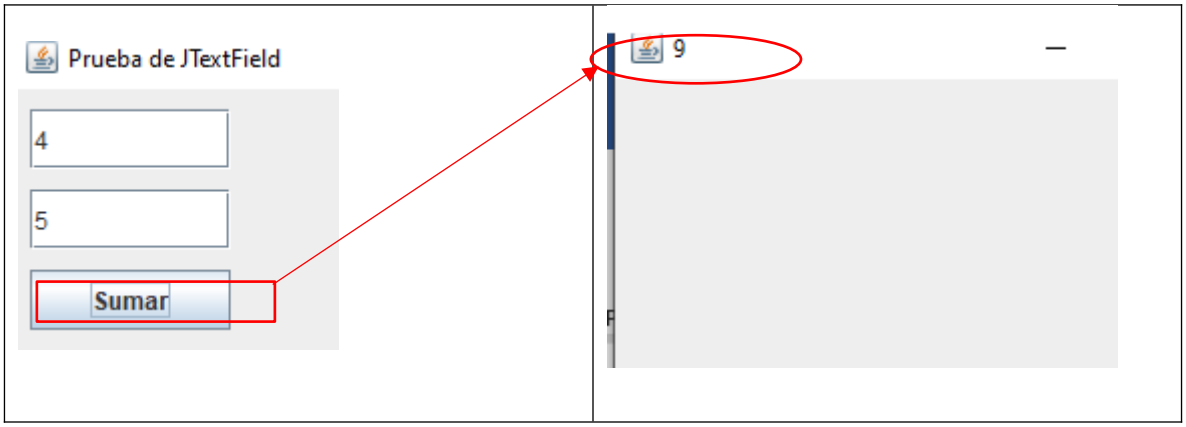

```
import javax.swing.*;
import java.awt.event.*;

public class CampoTextoPanel extends JPanel implements ActionListener{
    private JTextField textfield1;
    private JLabel label1;
    private JButton boton1;

    public CampoTextoPanel() {
        setLayout(null);
        label1=new JLabel("Usuario:");
        label1.setBounds(10,10,100,30);

        textfield1=new JTextField();
        textfield1.setBounds(120,10,150,20);

        boton1=new JButton("Aceptar");
```

```
        boton1.setBounds(10,80,100,30);
        //añadir componentes
        add(label1);
        add(textfield1);
        add(boton1);
        boton1.addActionListener(this);
    }
    @Override
    public void actionPerformed(ActionEvent e) {
        JFrame aplicacion= new JFrame();
        if (e.getSource()==boton1) {
            String cad=textfield1.getText();
            aplicacion.setTitle(cad);
            aplicacion.setSize(350,250);
            aplicacion.setVisible(true);

        }
    }
}
```

```
import javax.swing.JFrame;
public class App
{
    public static void main(String[] args)
    {
        // crea un panel que contiene nuestro dibujo
        CampoTextoPanel panel = new CampoTextoPanel();
        // crea un nuevo marco para contener el panel
        JFrame aplicacion = new JFrame("Prueba de JTextField");

        aplicacion.add(panel); // agrega el panel al marc
        aplicacion.setSize(350, 250); // establece el tamaño del marco
        aplicacion.setVisible(true);//hace el marco visible
        // establece el marco para salir cuando se cierre
        aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
} // fin de la clase App
```

Ejemplo 4 JLabel, TextField y Button

Confeccionar un programa que permita ingresar dos números en controles de tipo JTextField, luego sumar los dos valores ingresados y mostrar la suma en la barra del título del control JFrame.



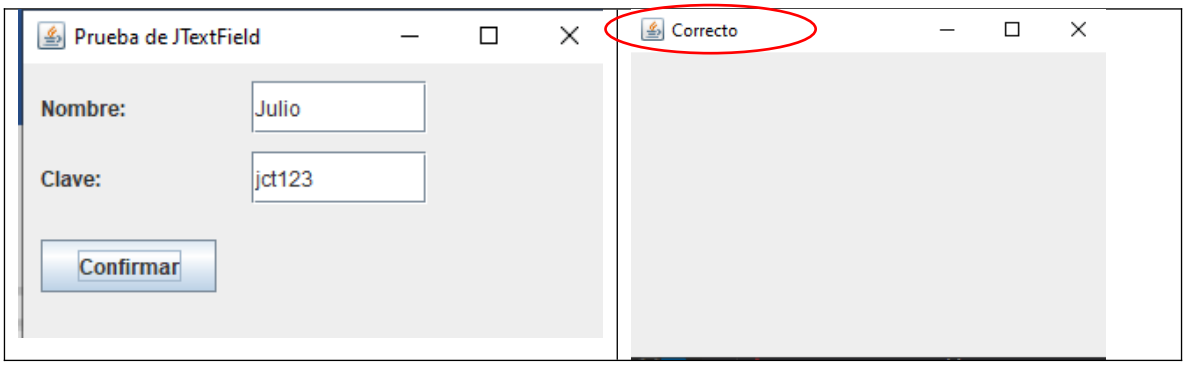

```
import javax.swing.*;
import java.awt.event.*;
public class CampoTextoPanel extends JPanel implements ActionListener{
   private JTextField textfield1;
   private JTextField textfield2;
   private JButton boton1;
   public CampoTextoPanel() {
      setLayout(null);
      textfield1=new JTextField();
      textfield1.setBounds(10,10,100,30);

      textfield2=new JTextField();
      textfield2.setBounds(10,50,100,30);

      boton1=new JButton("Sumar");
      boton1.setBounds(10,90,100,30);
      add(textfield1);
      add(textfield2);
      add(boton1);
      boton1.addActionListener(this);
   }

   public void actionPerformed(ActionEvent e) {
      JFrame aplicacion = new JFrame();
      if (e.getSource()==boton1) {
         String cad1=textfield1.getText();
         String cad2=textfield2.getText();
```

```
            int x1=Integer.parseInt(cad1);
            int x2=Integer.parseInt(cad2);
            int suma=x1+x2;
            String total=String.valueOf(suma);
            aplicacion.setTitle(total);
            aplicacion.setSize(350,250);
            aplicacion.setVisible(true);

        }
    }
}
```

```
import javax.swing.JFrame;
public class App
{
    public static void main(String[] args)
    {
        // crea un panel que contiene nuestro dibujo
        CampoTextoPanel panel = new CampoTextoPanel();
        // crea un nuevo marco para contener el panel
        JFrame aplicacion = new JFrame("Prueba de JTextField");

        aplicacion.add(panel); // agrega el panel al marc
        aplicacion.setSize(350, 250); // establece el tamaño del marco
        aplicacion.setVisible(true);//hace el marco visible
        // establece el marco para salir cuando se cierre
        aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    }
} // fin de la clase App
```

Ejemplo 5 JLabel, TextField y Button

Ingresar el nombre de usuario y clave en controles de tipo JTextField.
Si se ingresa las cadena (usuario: "Julio", clave: "jct123")
luego mostrar en el título del JFrame el mensaje "Correcto" en caso contrario "Incorrecto".



```
import javax.swing.*;
import java.awt.event.*;
public class CampoTextoPanel extends JPanel implements ActionListener {
   private JLabel label1;
   private JLabel label2;
   private JTextField textfield1;
   private JTextField textfield2;
   private JButton boton1;
   public CampoTextoPanel() {
      setLayout(null);
      label1=new JLabel("Nombre:");
      label1.setBounds(10,10,100,30);
      label2=new JLabel("Clave:");
      label2.setBounds(10,50,100,30);
      textfield1=new JTextField();
      textfield1.setBounds(130,10,100,30);

      textfield2=new JTextField();
      textfield2.setBounds(130,50,100,30);

      boton1=new JButton("Confirmar");
      boton1.setBounds(10,100,100,30);

      add(label1);
      add(label2);
      add(textfield1);
      add(textfield2);

      add(boton1);
      boton1.addActionListener(this);
   }


   public void actionPerformed(ActionEvent e) {
      JFrame aplicacion= new JFrame();
      if (e.getSource()==boton1) {
         String cad1=textfield1.getText();
         String cad2=textfield2.getText();
         if (cad1.equals("Julio")==true && cad2.equals("jct123")==true){
            aplicacion.setTitle("Correcto");
            aplicacion.setSize(350,250);
            aplicacion.setVisible(true);
         }else{
            aplicacion.setTitle("Incorrecto");
            aplicacion.setSize(350,250);
```

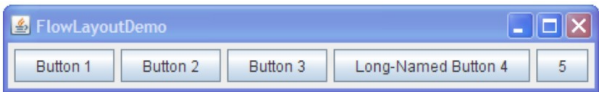
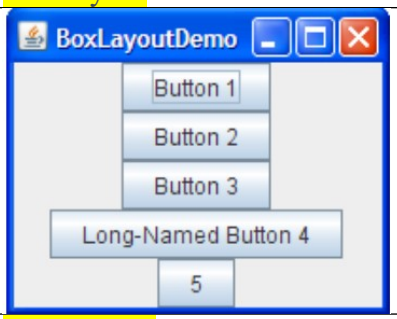
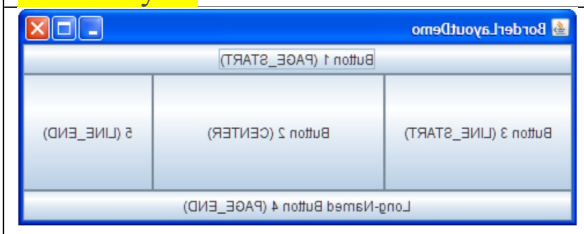
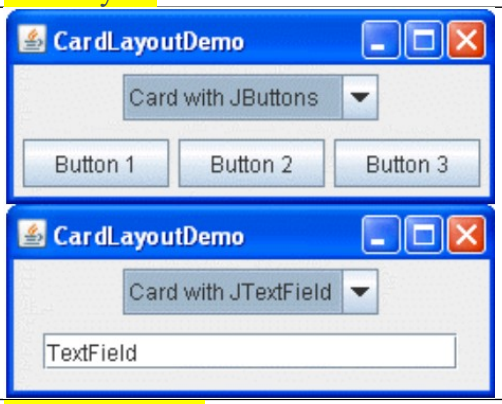
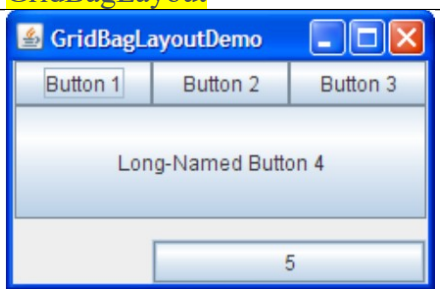
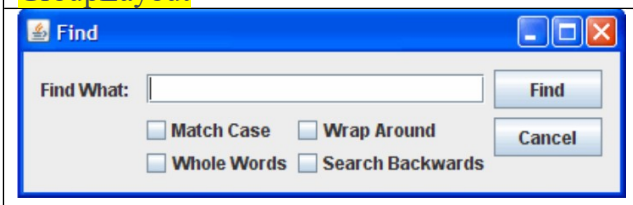
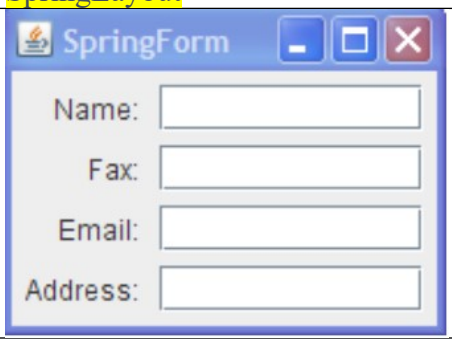
```
            aplicacion.setVisible(true);

        }
    }
  }
}
```

```
import javax.swing.JFrame;
public class App
{
   public static void main(String[] args)
   {
      // crea un panel que contiene nuestro dibujo
      CampoTextoPanel panel = new CampoTextoPanel();
      // crea un nuevo marco para contener el panel
      JFrame aplicacion = new JFrame("Prueba de JTextField");

      aplicacion.add(panel); // agrega el panel al marc
      aplicacion.setSize(350, 250); // establece el tamaño del marco
      aplicacion.setVisible(true);//hace el marco visible
      // establece el marco para salir cuando se cierre
      aplicacion.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

   }
} // fin de la clase App
```

Layout managers o gestores de composición

https://docs.oracle.com/javase/tutorial/uiswing/layout/visual.html

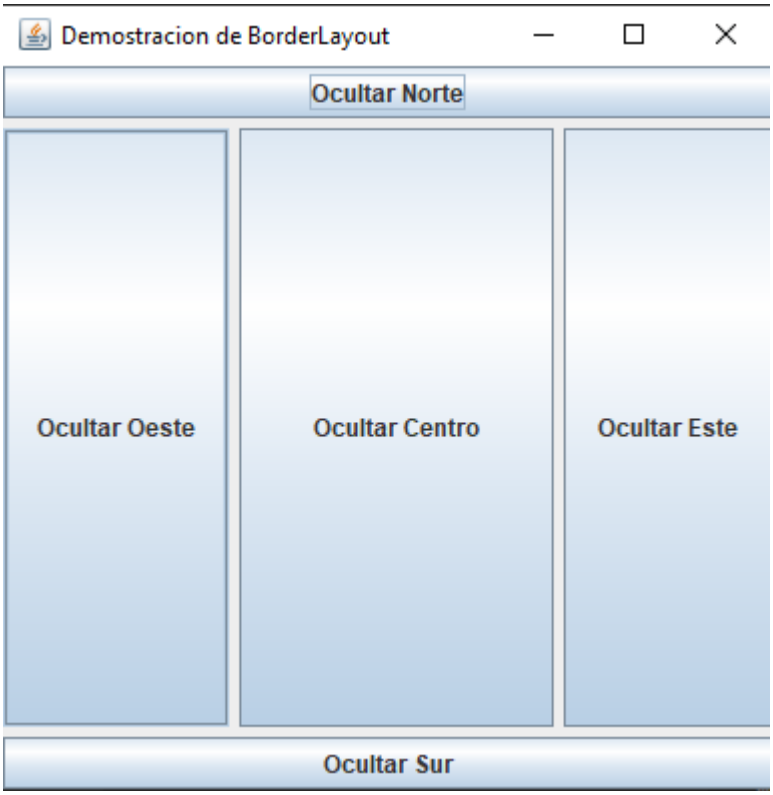
En java, cuando se hace ventanas, la clase que decide cómo se reparten los componentes dentro de la ventana se llama **Layout**. Esta clase es la que decide en qué posición van los botones y demás componentes, si van alineados, en forma de matriz, cuáles se hacen grandes al agrandar la ventana, etc. Otra cosa importante que decide el **Layout** es qué tamaño es el ideal para la ventana en función de los componentes que lleva dentro. Con un **layout** adecuado, el método **pack()** de la ventana hará que tome el tamaño necesario para que se vea todo lo que tiene dentro. Java dispone de varios tipos de *layout manager*, a saber:

FlowLayout	BoxLayout
	
BorderLayout	CardLayout
	
GridLayout	GridBagLayout
	
GroupLayout	SpringLayout
	

Ejemplo 6 Uso FlowLayout con buttons

Ejemplo 7 Uso BoxLayout con buttons

Ejemplo 8 Uso BorderLayout con buttons




```
// MarcoBorderLayout.java
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;

public class MarcoBorderLayout extends JFrame implements ActionListener
{
   private JButton botones[]; // arreglo de botones para ocultar porciones
   private final String nombres[] = { "Ocultar Norte", "Ocultar Sur", "Ocultar Este", "Ocultar
Oeste", "Ocultar Centro" };
   private BorderLayout esquema; // objeto BorderLayout
   // establece la GUI y el manejo de eventos
   public MarcoBorderLayout()    {
      super( "Demostracion de BorderLayout" );
      esquema = new BorderLayout( 5, 5 ); // espacios de 5 píxeles
```

```
      setLayout( esquema ); // establece el esquema del marco
      botones = new JButton[ nombres.length ]; // establece el tamaño del arreglo
      // crea objetos JButton y registra componentes de escucha para ellos
      for ( int cuenta = 0; cuenta < nombres.length; cuenta++ ) {
         botones[ cuenta ] = new JButton( nombres[ cuenta ] );
         botones[ cuenta ].addActionListener( this );
      } // fin de for
      add( botones[ 0 ], BorderLayout.NORTH ); // agrega botón al norte
      add( botones[ 1 ], BorderLayout.SOUTH ); // agrega botón al sur
      add( botones[ 2 ], BorderLayout.EAST ); // agrega botón al este
      add( botones[ 3 ], BorderLayout.WEST ); // agrega botón al oeste
      add( botones[ 4 ], BorderLayout.CENTER ); // agrega botón al centro
   } // fin del constructor de MarcoBorderLayout
   // maneja los eventos de botón
   public void actionPerformed( ActionEvent evento ) {
      //  comprueba el origen del evento y distribuye el panel de contenido de manera acorde
      for ( JButton boton : botones )  {
         if ( evento.getSource() == boton )
            boton.setVisible( false );
      // oculta el botón oprimido
         else
            boton.setVisible( true ); // muestra los demás botones
      } // fin de for
      esquema.layoutContainer( getContentPane() ); // distribuye el panel de contenido
   } // fin del método actionPerformed
} // fin de la clase MarcoBorderLayout
```

```
// DemoBorderLayout.java
import javax.swing.JFrame;
public class DemoBorderLayout {
  public static void main( String[] args ) {
     MarcoBorderLayout marcoBorderLayout = new MarcoBorderLayout();
     marcoBorderLayout.setSize( 400, 400 ); // establece el tamaño del marco
     marcoBorderLayout.setVisible( true ); // muestra el marco
     marcoBorderLayout.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
  } // fin de main
} // fin de la clase DemoBorderLayout
```

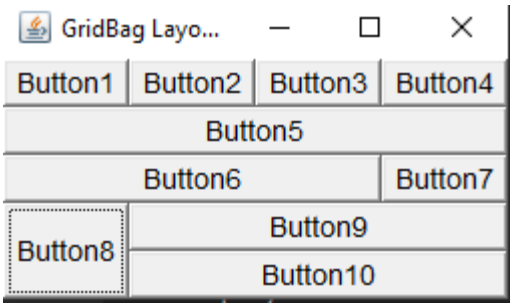
Ejemplo 9 Uso GridLayout con buttons


```
//MarcoGridLayout.java
// Demostración de GridLayout.
import java.awt.GridLayout;
import java.awt.Container;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.JFrame;
import javax.swing.JButton;
public class MarcoGridLayout extends JFrame implements ActionListener  {
   private JButton[] botones;  // arreglo de botones
   private final String[] nombres = { "uno", "dos", "tres", "cuatro", "cinco", "seis" };
   private boolean alternar = true;// alterna entre dos esquemas
   private Container contenedor; // contenedor del marco
   private GridLayout cuadricula1; // primer objeto GridLayout
   private GridLayout cuadricula2; // segundo objeto GridLayout
   // constructor sin argumentos
   public MarcoGridLayout()    {
      super( "Demostracion de GridLayout" );

      cuadricula1 = new GridLayout( 2, 3, 5, 5 ); // 2 por 3; espacios de 5
      cuadricula2 = new GridLayout( 3, 2 ); // 3 por 2; sin espacios
      contenedor = getContentPane(); // obtiene el panel de contenido
      setLayout( cuadricula1 ); // establece esquema de objeto JFrame
      botones = new JButton[ nombres.length ]; // crea arreglo de objetos JButton
      for ( int cuenta = 0; cuenta < nombres.length; cuenta++ )  {
         botones[ cuenta ] = new JButton( nombres[ cuenta ] );
         botones[ cuenta ].addActionListener( this ); // registra componente de escucha
         add( botones[ cuenta ] ); // agrega boton a objeto JFrame
      } // fin de for
   } // fin del constructor de MarcoGridLayout
// maneja eventos de boton, alternando entre los esquemas
   public void actionPerformed( ActionEvent evento ){
      if ( alternar )
      contenedor.setLayout( cuadricula2 );// establece esquema al primero
      else  contenedor.setLayout( cuadricula1 ); // establece esquema al segundo
      alternar = !alternar; // establece alternar a su valor opuesto
```

```
        contenedor.validate(); // redistribuye el contenedor
    } // fin del método actionPerformed
} // fin de la clase MarcoGridLayout
```

```
// DemoGridLayout.java
// Prueba de MarcoGridLayout.
import javax.swing.JFrame;
public class DemoGridLayout {
   public static void main( String[] args ){
      MarcoGridLayout marcoGridLayout = new MarcoGridLayout();
      marcoGridLayout.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
      marcoGridLayout.setSize( 400, 200 ); // establece el tamaño del marco 1
      marcoGridLayout.setVisible( true ); // muestra el marco
   } // fin de main
} // fin de la clase DemoGridLayout
```

Ejemplo 9 Uso GridBagLayout con buttons



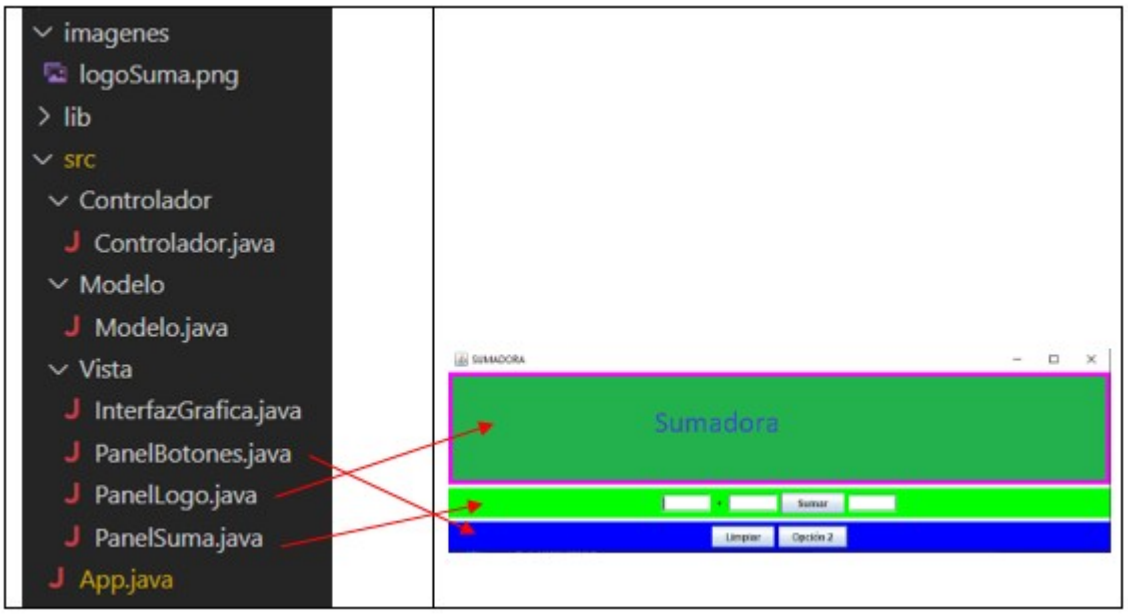

```
import java.awt.*;

import javax.swing.JPanel;

 public class GridBagLayoutDemo extends JPanel {

    protected void makebutton(String name, GridBagLayout gridbag,  GridBagConstraints c) {
       Button button = new Button(name);
       gridbag.setConstraints(button, c);
       add(button);
    }

    public void init() {
       GridBagLayout gridbag = new GridBagLayout();
       GridBagConstraints c = new GridBagConstraints();

       setFont(new Font("SansSerif", Font.PLAIN, 14));
```

```
        setLayout(gridbag);

        c.fill = GridBagConstraints.BOTH;
        c.weightx = 1.0;
        makebutton("Button1", gridbag, c);
        makebutton("Button2", gridbag, c);
        makebutton("Button3", gridbag, c);

        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button4", gridbag, c);

        c.weightx = 0.0;              //reset to the default
        makebutton("Button5", gridbag, c); //another row

        c.gridwidth = GridBagConstraints.RELATIVE; //next-to-last in row
        makebutton("Button6", gridbag, c);

        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        makebutton("Button7", gridbag, c);

        c.gridwidth = 1;              //reset to the default
        c.gridheight = 2;
        c.weighty = 1.0;
        makebutton("Button8", gridbag, c);

        c.weighty = 0.0;              //reset to the default
        c.gridwidth = GridBagConstraints.REMAINDER; //end row
        c.gridheight = 1;             //reset to the default
        makebutton("Button9", gridbag, c);
        makebutton("Button10", gridbag, c);

        setSize(300, 100);
    }

    public static void main(String args[]) {
      //  setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        Frame f = new Frame("GridBag Layout Example");
       GridBagLayoutDemo ex1 = new GridBagLayoutDemo();

        ex1.init();

        f.add("Center", ex1);
        f.pack();
        f.setSize(f.getPreferredSize());
        f.setVisible(true);
        }
}
```

```
import java.awt.*;

import javax.swing.JPanel;

 public class GridBagLayoutDemo extends JPanel {

    protected void makebutton(String name, GridBagLayout gridbag,  GridBagConstraints c) {
       Button button = new Button(name);
       gridbag.setConstraints(button, c);
       add(button);
    }

    public void init() {
       GridBagLayout gridbag = new GridBagLayout();
       GridBagConstraints c = new GridBagConstraints();

       setFont(new Font("SansSerif", Font.PLAIN, 14));
       setLayout(gridbag);

       c.fill = GridBagConstraints.BOTH;
       c.weightx = 1.0;
       makebutton("Button1", gridbag, c);
       makebutton("Button2", gridbag, c);
       makebutton("Button3", gridbag, c);

       c.gridwidth = GridBagConstraints.REMAINDER; //end row
       makebutton("Button4", gridbag, c);

       c.weightx = 0.0;              //reset to the default
       makebutton("Button5", gridbag, c); //another row

       c.gridwidth = GridBagConstraints.RELATIVE; //next-to-last in row
       makebutton("Button6", gridbag, c);

       c.gridwidth = GridBagConstraints.REMAINDER; //end row
       makebutton("Button7", gridbag, c);

       c.gridwidth = 1;              //reset to the default
       c.gridheight = 2;
       c.weighty = 1.0;
       makebutton("Button8", gridbag, c);

       c.weighty = 0.0;              //reset to the default
       c.gridwidth = GridBagConstraints.REMAINDER; //end row
       c.gridheight = 1;             //reset to the default
       makebutton("Button9", gridbag, c);
       makebutton("Button10", gridbag, c);
```

```
    setSize(300, 100);
  }

  public static void main(String args[]) {
    //  setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
     Frame f = new Frame("GridBag Layout Example");
    GridBagLayoutDemo ex1 = new GridBagLayoutDemo();

     ex1.init();

     f.add("Center", ex1);
     f.pack();
     f.setSize(f.getPreferredSize());
     f.setVisible(true);
     }
}
```

Ejemplo 10 MVC Sumadora




```
import Controlador.Controlador;
import Vista.InterfazGrafica;
import Modelo.Modelo;

public class App {
    public static void main(String[] args) {
        InterfazGrafica vista;
        Modelo modelo;
        Controlador controlador;

        vista = new InterfazGrafica();
        modelo = new Modelo();
        controlador = new Controlador(vista, modelo);
    }
}
```

```
public class Modelo {
    private int numeroUno;
    private int numeroDos;
    private int resultado;

    public int getNumeroUno() {
        return numeroUno;
    }
```

```
    public void setNumeroUno(int numeroUno) {
        this.numeroUno = numeroUno;
    }

    public int getNumeroDos() {
        return numeroDos;
    }

    public void setNumeroDos(int numeroDos) {
        this.numeroDos = numeroDos;
    }

    public int getResultado() {
        return resultado;
    }

    public void setResultado(int resultado) {
        this.resultado = resultado;
    }

    public int sumar(){
        this.resultado = this.numeroUno + this.numeroDos;
        return this.resultado;
    }

    public int sumar(int n1, int n2){
        this.resultado = this.numeroUno + this.numeroDos;
        return this.resultado;
    }
}
```

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

public class InterfazGrafica {

    //define los atributos
    private JFrame    marcoPrincipal;
    private Container contenedorMarcoPrincipal;
    private PanelLogo  logoPanel;
    private PanelSuma  sumaPanel;
    private PanelBotones  botonesPanel;

    public InterfazGrafica() {


        // Crear marco principal
```

```
    marcoPrincipal          = new JFrame("SUMADORA");
    contenedorMarcoPrincipal = marcoPrincipal.getContentPane();
    contenedorMarcoPrincipal.setLayout(new BorderLayout(5,5));

    logoPanel = new PanelLogo();
    sumaPanel = new PanelSuma();
     botonesPanel = new PanelBotones();

    contenedorMarcoPrincipal.add(logoPanel, BorderLayout.NORTH);
    contenedorMarcoPrincipal.add(sumaPanel, BorderLayout.CENTER);
    contenedorMarcoPrincipal.add(botonesPanel, BorderLayout.SOUTH);

    // Ajustar el contenedor a los componentes

    marcoPrincipal.pack();// establece el tamaño por defecto
   // marcoPrincipal.setResizable(true);
    marcoPrincipal.setVisible(true); //hace el marco visible
    // establece el marco para salir cuando se cierre
    marcoPrincipal.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

}

public JFrame getMarcoPrincipal() {
    return marcoPrincipal;
}

public void setMarcoPrincipal(JFrame marcoPrincipal) {
    this.marcoPrincipal = marcoPrincipal;
}

public Container getContenedorMarcoPrincipal() {
    return contenedorMarcoPrincipal;
}

public void setContenedorMarcoPrincipal(Container contenedorMarcoPrincipal) {
    this.contenedorMarcoPrincipal = contenedorMarcoPrincipal;
}

public PanelLogo getLogoPanel() {
    return logoPanel;
}

public void setLogoPanel(PanelLogo logoPanel) {
    this.logoPanel = logoPanel;
}

public PanelSuma getSumaPanel() {
    return sumaPanel;
```

```
    }

    public void setSumaPanel(PanelSuma sumaPanel) {
        this.sumaPanel = sumaPanel;
    }

    public PanelBotones getBotonesPanel() {
        return botonesPanel;
    }

    public void setBotonesPanel(PanelBotones botonesPanel) {
        this.botonesPanel = botonesPanel;
    }
}
```

```
import javax.swing.ImageIcon;
import javax.swing.JLabel;
import javax.swing.JPanel;

public class PanelLogo extends JPanel
{
    private static final String RUTA_IMAGEN = "./imagenes/logoSuma.png";//Constante Ruta de la
imagen del banner.

    private JLabel etiqueta;//Etiqueta en la que se implementa la imagen.

    // ---------------------------------------------------------------
    // Constructores
    // ---------------------------------------------------------------
    public PanelLogo( )
    {
        setBackground(Color.MAGENTA);
        ImageIcon ruta = new ImageIcon(RUTA_IMAGEN);
        etiqueta = new JLabel( );
        etiqueta.setIcon(ruta);
        //etiqueta.setIcon( new ImageIcon( RUTA_IMAGEN ) );
        add( etiqueta );
    }
}
```

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

public class PanelSuma extends JPanel{
```

```
// declaration  Campos-
public JButton btnSumar;
private JLabel jLabel1;
public JTextField txtNumeroDos;
public JTextField txtNumeroUno;
public JTextField txtSuma;
// final declaration de campos

public  PanelSuma (){
   setLayout(new FlowLayout());
   setBackground(Color.GREEN);

   btnSumar = new JButton(" Sumar ");
   jLabel1 = new JLabel(" + ");
   txtNumeroDos = new JTextField(5);
   txtNumeroUno = new JTextField(5);
   txtSuma = new JTextField(5);

    add(txtNumeroUno );
    add(jLabel1 );
    add( txtNumeroDos );
    add( btnSumar );
    add(txtSuma );

}

public JButton getBtnSumar() {
   return btnSumar;
}

public void setBtnSumar(JButton btnSumar) {
   this.btnSumar = btnSumar;
}

public JTextField getTxtNumeroDos() {
   return txtNumeroDos;
}

public void setTxtNumeroDos(JTextField txtNumeroDos) {
   this.txtNumeroDos = txtNumeroDos;
}

public JTextField getTxtNumeroUno() {
   return txtNumeroUno;
}

public void setTxtNumeroUno(JTextField txtNumeroUno) {
   this.txtNumeroUno = txtNumeroUno;
```

```
    }

    public JTextField getTxtSuma() {
        return txtSuma;
    }

    public void setTxtSuma(JTextField txtSuma) {
        this.txtSuma = txtSuma;
    }
}
```

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

public class PanelBotones extends JPanel {

     /**
     * Constante para el extensión 1.
     */
    private final String OPCION_1 = "opcion1";

    /**
     * Cosntante para la extensión 2.
     */
    private final String OPCION_2 = "opcion2";
    /**
     * Botón para la extensión 1.
     */
    private JButton btnOpcion1;

    /**
     * Botón para la extensión 2.
     */
    private JButton btnOpcion2;

    public PanelBotones(){
        setLayout(new FlowLayout());
        setBackground(Color.BLUE);

        btnOpcion1 = new JButton( "Limpiar" );
        btnOpcion1.setActionCommand( OPCION_1 );


        btnOpcion2 = new JButton( "Opción 2" );
        btnOpcion2.setActionCommand( OPCION_2 );

        add( btnOpcion1 );
```

```
        add( btnOpcion2 );
    }

    public JButton getBtnOpcion1() {
        return btnOpcion1;
    }

    public void setBtnOpcion1(JButton btnOpcion1) {
        this.btnOpcion1 = btnOpcion1;
    }

    public JButton getBtnOpcion2() {
        return btnOpcion2;
    }

    public void setBtnOpcion2(JButton btnOpcion2) {
        this.btnOpcion2 = btnOpcion2;
    }

}
```

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JOptionPane;

import Modelo.Modelo;
import Vista.InterfazGrafica;

public class Controlador implements ActionListener{

    private InterfazGrafica vista= new InterfazGrafica();
    private Modelo modelo  = new Modelo();

    public Controlador(InterfazGrafica view, Modelo modelo) {
        this.vista = view;
        this.modelo= modelo;
        this.vista.getSumaPanel().getBtnSumar().addActionListener(this);
        this.vista.getBotonesPanel().getBtnOpcion1().addActionListener(this);
        this.vista.getBotonesPanel().getBtnOpcion2().addActionListener(this);
    }

    public void actionPerformed(ActionEvent ae){
        if (ae.getSource() == vista.getSumaPanel().getBtnSumar()){

            modelo.setNumeroUno(Integer.parseInt(vista.getSumaPanel().txtNumeroUno.getText()));
            modelo.setNumeroDos(Integer.parseInt(vista.getSumaPanel().txtNumeroDos.getText()));
            vista.getSumaPanel().getTxtSuma().setText(String.valueOf(modelo.sumar()));
```

```
      }else  if (ae.getSource() == this.vista.getBotonesPanel().getBtnOpcion1()){
          //String comando = this.vista.getBotonesPanel().getBtnOpcion1().getActionCommand( );
          vista.getSumaPanel().txtNumeroUno.setText(null);
          vista.getSumaPanel().txtNumeroDos.setText(null);
          vista.getSumaPanel().txtSuma.setText(null);

         }
         else if (ae.getSource() == this.vista.getBotonesPanel().getBtnOpcion2()) {
            {

               JOptionPane.showMessageDialog( null,"Otra posibilidad de hacer algo ");
            }

         }
   }
}
```