Interfaces Graficas de Usuario con Java Swing.

Las interfaces graficas son parte fundamental en el desarrollo de aplicaciones, estas permiten al usuario tener una mejor interacción con el sistema y se compone de ventanas que a su vez contienen diferentes controles como etiquetas, cajas de texto, botones entre otros.

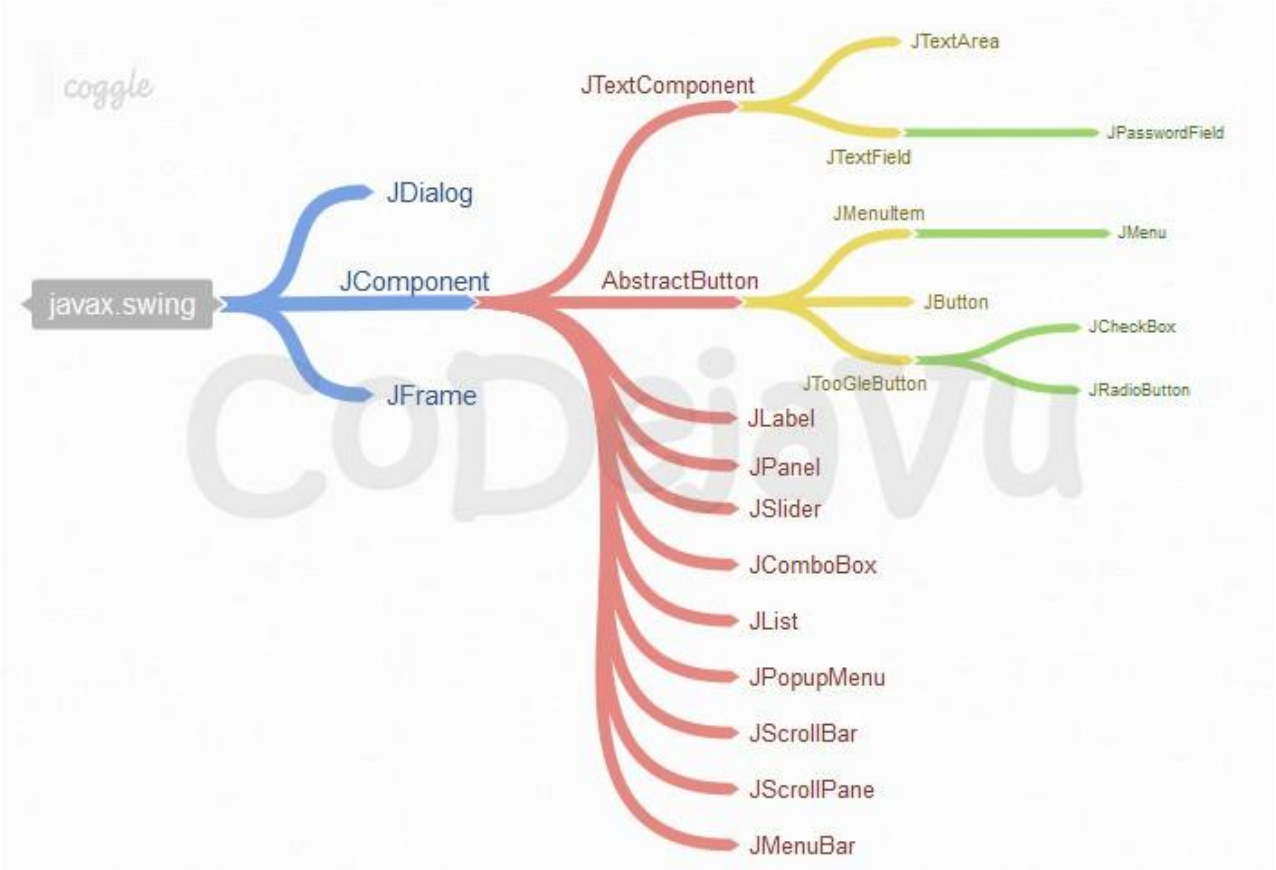
Java proporciona una biblioteca de clases llamada JSF (Java Fundation Classes) de la que hace parte Swing, la cual provee herramientas o facilidades para la construcción de GUI's o interfaces Graficas de Usuario (graphical user interface).

Swing es un conjunto de componentes para diseñar interfaces gráficas de usuario que se ejecutan en cualquier plataforma que soporte la máquina virtual de Java.

Anteriormente se trabajaba con AWT (Abstract Window Toolkit), la cual al igual que Swing es un conjunto de librerías enfocadas a la construcción de interfaces, solo que con esta se presentaron algunos problemas en cuanto a portabilidad principalmente cuando se desarrollaban aplicaciones para diferentes sistemas operativos, pues el comportamiento de los componentes gráficos en ocasiones podían variar

Controles Java Swing.

Existen diferentes controles o componentes gráficos que provee Java Swing, los cuales se pueden evidenciar en el siguiente árbol que contiene los más comunes , dichos componentes se almacenan en el paquete javax.swing



Los componentes Swing se reconocen porque anteponen la letra J antes del nombre, por ejemplo un botón en AWT se llama Button, mientras que en Java Swing es JButton

Cada componente corresponde a una clase en Java, por esta razón cuando desarrollamos y queremos vincular uno de estos elementos simplemente instanciamos la clase que necesitamos, es decir, si queremos un Área de texto debemos crear un objeto de la clase JTextArea.

Categorías de componentes.

Los componentes anteriores se agrupan en diferentes categorías dependiendo de su funcionalidad. (Contenedores, componentes atomicos o simples, componentes de texto, componentes de menú, componentes complejos.)

Contenedores:

Un contenedor es el tapiz donde pintaremos nuestros componentes gráficos, si queremos crear una ventana, por defecto está será un contenedor, pues en ella se almacenan el resto de componentes gráficos.

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

Componentes Atómicos o Simples:

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

- **JLabel** – Permite Vincular Etiquetas, tanto de texto como de imágenes
- **JButton** – Permite vincular Botones simples.
- **JCheckBox** – Son Casilla de verificación, ideal para selección múltiples.
- **JRadioButton** – Permite presentar opciones de selección similares a las checkbox, solo que el enfoque de estas es de única selección.
- **JToggleButton** – Botón que al oprimirlo se quedará presionado hasta que se ejecute otro evento.

- **JComboBox** – Permite mostrar una lista de elementos como un combo de selección.
- **JScrollBar** – Permite mostrar una barra de desplazamiento, regularmente usada en Áreas de texto o paneles donde el contenido es mayor que el tamaño del componente.
- **JSeparator** – Permite separar opciones, es una barra simple.
- **JSlider** - Permite vincular un Deslizador en nuestra ventana.
- **JSpinner** – permite vincular una caja de texto con botones integrados para seleccionar algún valor.
- **JProgressBar** – Establece una barra de progreso.

Componentes de Texto.

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

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

Componentes de Menú.

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

- **JMenuBar** – Permite vincular una barra de menús.
- **JMenu**– Permite vincular botones o enlaces que al ser pulsados despliegan un menú principal.
- **JMenuItem** – Botón u opción que se encuentra en un menú.
- **JCheckBoxMenuItem**– Elemento del menú como opciones de checkbox.
- **JRadioButtonMenuItem**– Elemento del menú como botón de selección.
- **JPopupMenu**– Opciones de menú emergente.

Componentes Complejos.

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

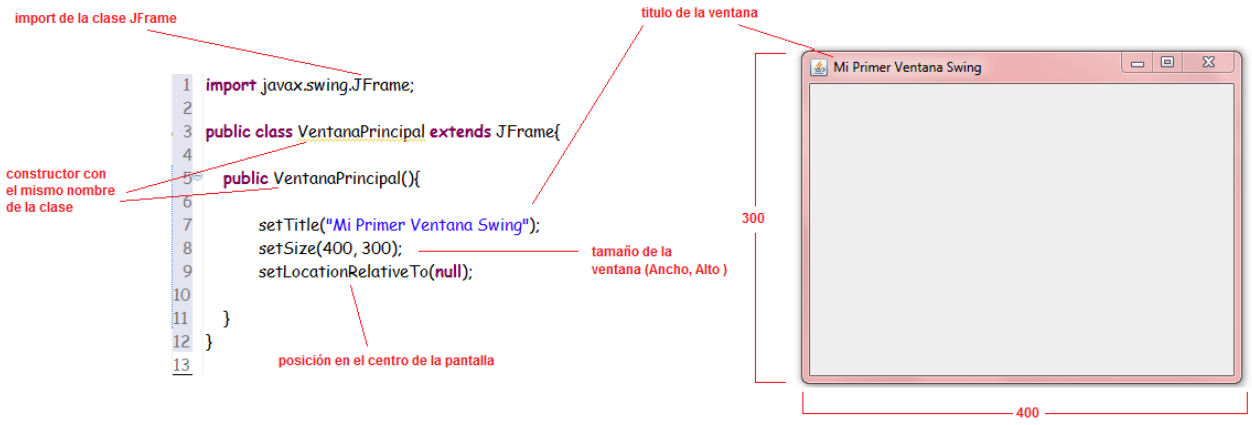
- **JTable** - Permite vincular una tabla de datos con sus respectivas filas y columnas.

- **JTree** - Carga un árbol donde se establece cierta jerarquía visual, tipo directorio.
- **JList** – Permite cargar una lista de elementos, dependiendo de las propiedades puede tenerse una lista de selección múltiple.
- **JFileChooser** – Es un componente que permite la búsqueda y selección de ficheros entre otras.
- **JColorChooser** – Componente que permite cargar un panel selector de color
- **JOptionPane** – No es algo complejo sino más un componente independiente que permite mostrar un cuadro de diálogo personalizable.

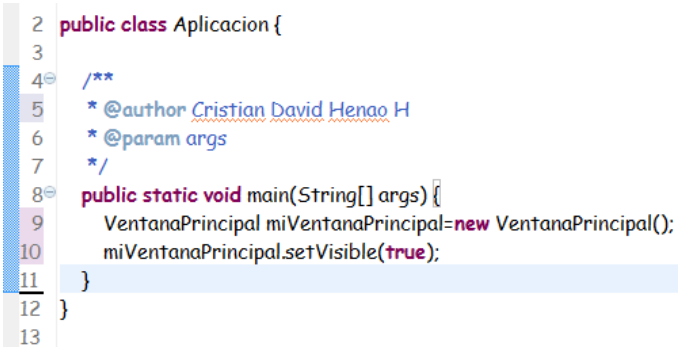
Los anteriores son los principales componentes gráficos (más usados) en java, cuando creamos interfaces gráficas, en este caso usando la librería Java Swing, empezamos a trabajar aplicando los conceptos de programación orientada a objetos, ya que como se mencionó y pudimos ver anteriormente cada componente grafico está representado por medio de una clase, así cuando agregamos un nuevo elemento por ejemplo un botón, lo que estamos haciendo es creando un objeto botón de la clase JButton y lo agregamos a nuestra ventana.

Crear una Ventana Con Java Swing.

Inicialmente la creación de ventanas se puede realizar de forma muy simple, creando la clase para nuestra interface gráfica, esta debe heredar (extends) de la clase JFrame y en el constructor poner las propiedades básicas de título y tamaño.



Por último se debe crear la clase principal con el método main, y en ella un objeto de nuestra clase ventana para abrirla y visualizarla usando el método setVisible(true) propio de JFrame.



```
2  public class Aplicacion {
3
4      /**
5       * @author Cristian David Henao H
6       * @param args
7       */
8      public static void main(String[] args) {
9          VentanaPrincipal miVentanaPrincipal=new VentanaPrincipal();
10         miVentanaPrincipal.setVisible(true);
11     }
12 }
13
```

Es importante que para que se pueda visualizar la ventana por medio del método setVisible, primero se haya heredado de jframe como se puede ver en la línea 3 de la clase VentanaPrincipal, así como crear el método constructor.

Método Constructor.

El método constructor es un método especial que se debe llamar igual a la clase que lo contiene, este método no tiene retorno, a excepción de esto último se comporta igual a cualquier método, pudiendo ser llamado con o sin parámetros, este método se puede ver en la clase VentanaPrincipal en la línea 5.

El método constructor es el encargado de darle vida al objeto, en él se debe poner todo lo necesario para inicializar componentes, cuando decimos **new NombreClase();** en realidad estamos llamado al método constructor para que se cree el objeto de nuestra clase, este proceso se puede ver en la clase Aplicación en la línea 9.

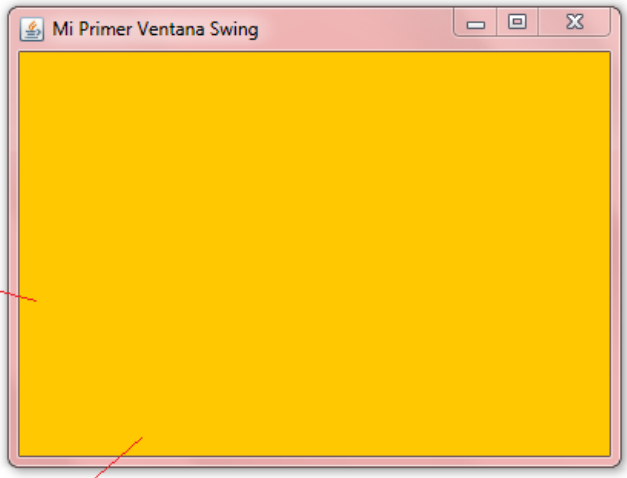
Agregar componentes a la ventana.

Para empezar a crear nuestra GUI, podemos empezar a agregarle componentes de los que provee java swing, tan solo es cuestión de crear los elementos y darle las propiedades de tamaño, ubicación entre otras.

Agregando un JPanel.

Un JPanel es un tipo de contenedor, en el podemos almacenar el resto de componentes, la forma de agregarlo es creando el objeto y posteriormente dándole las propiedades necesarias para su ubicación.

```
1  import java.awt.Color;
2  import javax.swing.JFrame;
3  import javax.swing.JPanel;
4
5  public class VentanaPrincipal extends JFrame{
6
7     JPanel miPanel;
8
9     public VentanaPrincipal(){
10          iniciarComponentes();
11          setTitle("Mi Primer Ventana Swing");
12          setSize(400, 300);
13          setLocationRelativeTo(null);
14    }
15
16    private void iniciarComponentes() {
17       miPanel=new JPanel();
18       miPanel.setLayout(null);
19       miPanel.setBackground(Color.orange);
20
21       add(miPanel);
22    }
23 }
24
```



- Para agregar el panel primero se declara de forma global (ver línea 7).
- Dentro del constructor se hace el llamado a el método iniciarComponentes() que permitirá la creación de los componentes gráficos.
- En la línea 17 dentro del método iniciarComponentes(), se declara el objeto JPanel creado anteriormente en la línea 7.
- En la línea 18 se establece cual va a ser la distribución de los componentes gráficos que vamos a agregar posteriormente al panel, al decirle null estamos indicando que nosotros mismos vamos a darles el tamaño y la ubicación.
- En la línea 19 se define por medio del método setBackground() cuál va a ser el color de nuestro panel
- Por ultimo en la línea 21 por medio del método add() agregamos el componente creado a nuestra ventana.

Agregando un JLabel y un botón.

El JLabel es la clase que permite agregar etiquetas de texto plano a nuestra ventana, el JButton es la clase que permite agregar botones.

Así como el JPanel se debe crear el objeto y posteriormente en el método iniciarComponentes() agregar las propiedades necesarias.

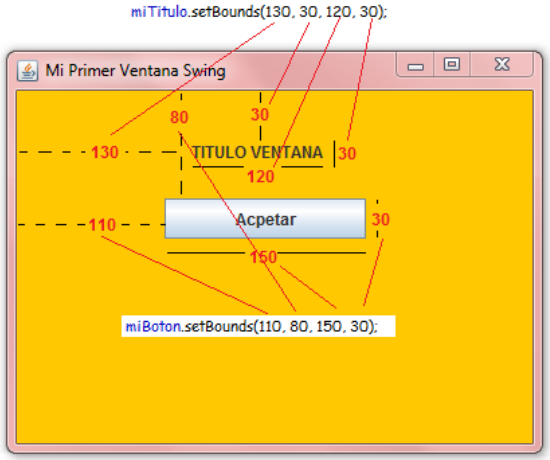
```
3  import javax.swing.JFrame;
4  import javax.swing.JLabel;
5  import javax.swing.JPanel;
6
7  public class VentanaPrincipal extends JFrame{
8
9     JPanel miPanel;
10    JLabel miTitulo;
11    JButton miBoton;
12
13    public VentanaPrincipal(){
14          iniciarComponentes();
15          setTitle("Mi Primer Ventana Swing");
16          setSize(400, 300);
17          setLocationRelativeTo(null);
18    }
19
20    private void iniciarComponentes() {
21       miPanel=new JPanel();
22       miPanel.setLayout(null);
23       miPanel.setBackground(Color.orange);
24
25       miTitulo=new JLabel();
26       miTitulo.setText("TITULO VENTANA");
27       miTitulo.setBounds(130, 30, 120, 30);
28
29       miBoton=new JButton();
30       miBoton.setText("Acpetar");
31       miBoton.setBounds(110, 80, 150, 30);
32
33       miPanel.add(miTitulo);
34       miPanel.add(miBoton);
35
36       add(miPanel);
37    }
38 }
```

Imports necesarios por cada componente

declaración de componentes graficos

creación de los componentes graficos con sus respectivas propiedades

se agregan los componentes al Panel creado



Se debe tener en cuenta que cada componente debe ser importado a la clase, como vemos en la línea 9,10 y 11, ahora teniendo en cuenta que los componentes se encuentran en el mismo paquete javax.swing, nos podemos ahorrar el import de cada uno escribiendo solamente

```
5  import javax.swing.*;
```

Así java permitirá usar cualquier componente de dicho paquete.

Eventos al Botón.

Hasta ahora tenemos una pequeña aplicación con una ventana y 3 componentes gráficos (Panel, etiqueta de texto y botón), pero hasta el momento nuestra aplicación no hace nada, si presionamos el botón veremos que el sistema se mantiene igual, para esto debemos agregar eventos al botón.

Eventos.

Los eventos son básicamente las acciones que suceden al ejecutar o realizar alguna invocación, un evento es una acción que podemos controlar cuando por ejemplo se ejecuta algún proceso, si ingresamos a cierto componente, presionamos un botón, movemos el mouse etc internamente el sistema identifica que es lo que se está haciendo, si hacemos clic, presionamos una tecla, movemos el mouse, todas esas son acciones que pueden ser identificadas y controladas

En Java estos eventos son conocidos como **Listeners** o escuchadores, como se mencionó podemos conocer gran variedad de eventos, para este ejemplo tan solo vamos a centrarnos en el evento ocurrido cuando presionamos el botón.

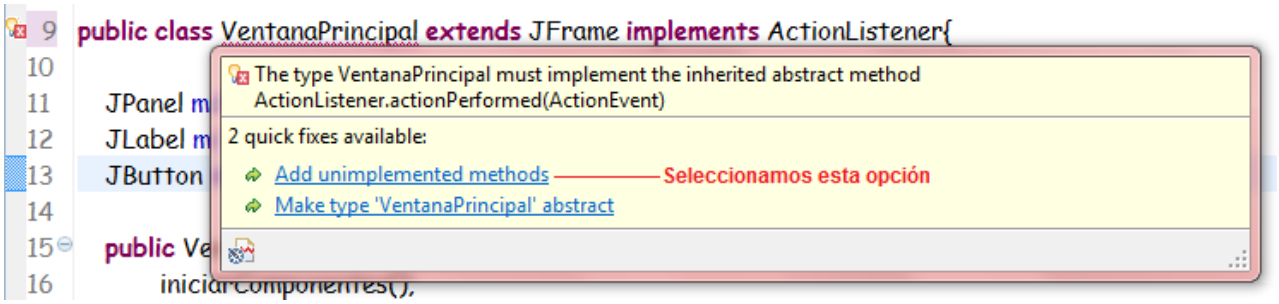
Escuchando al Botón.

Cuando queremos darnos cuenta de que el usuario presionó un botón, tenemos que agregar a la declaración de la clase la palabra implements y despues la interface que define cuales son los eventos que queremos controlar (cuando se habla de interface no se refiere a interface gráfica de usuario, sino a una clase especial que permite administrar propiedades que vamos a utilizar)

La interface que vamos a usar es la ActionListener por lo tanto agregamos esto en nuestra clase.

```
10 public class VentanaPrincipal extends JFrame implements ActionListener{
```

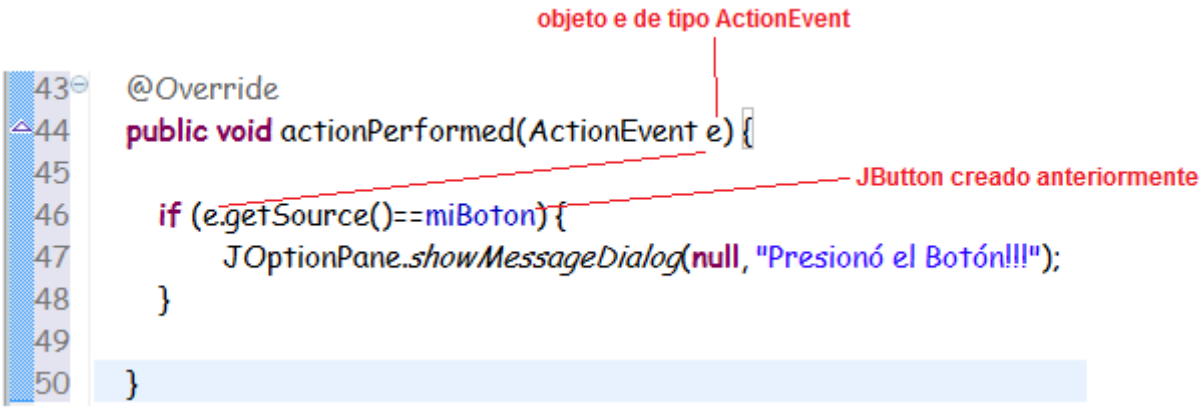
Al hacerlo en el nombre de la clase nos aparecerá un error, este error se presenta ya que cuando implementamos interfaces, java nos obliga a usar todos los métodos que allí se encuentren, para solucionar esto, desde eclipse se pone el mouse sobre el error y se selecciona la opción de implementar los métodos necesarios.



Al hacerlo, al final de la clase se crea el método ActionPerformed() que permite escuchar cuando el botón es presionado.

```
41
42   @Override
43   public void actionPerformed(ActionEvent e) {
44      // TODO Auto-generated method stub
45
46   }
47 }
48
```

Lo siguiente que debemos hacer es poner la lógica para que el botón que tenemos sea escuchado, esto se hace por medio del parámetro "e" de tipo ActionEvent y el método getSource que permite darse cuenta de que botón se presionó.

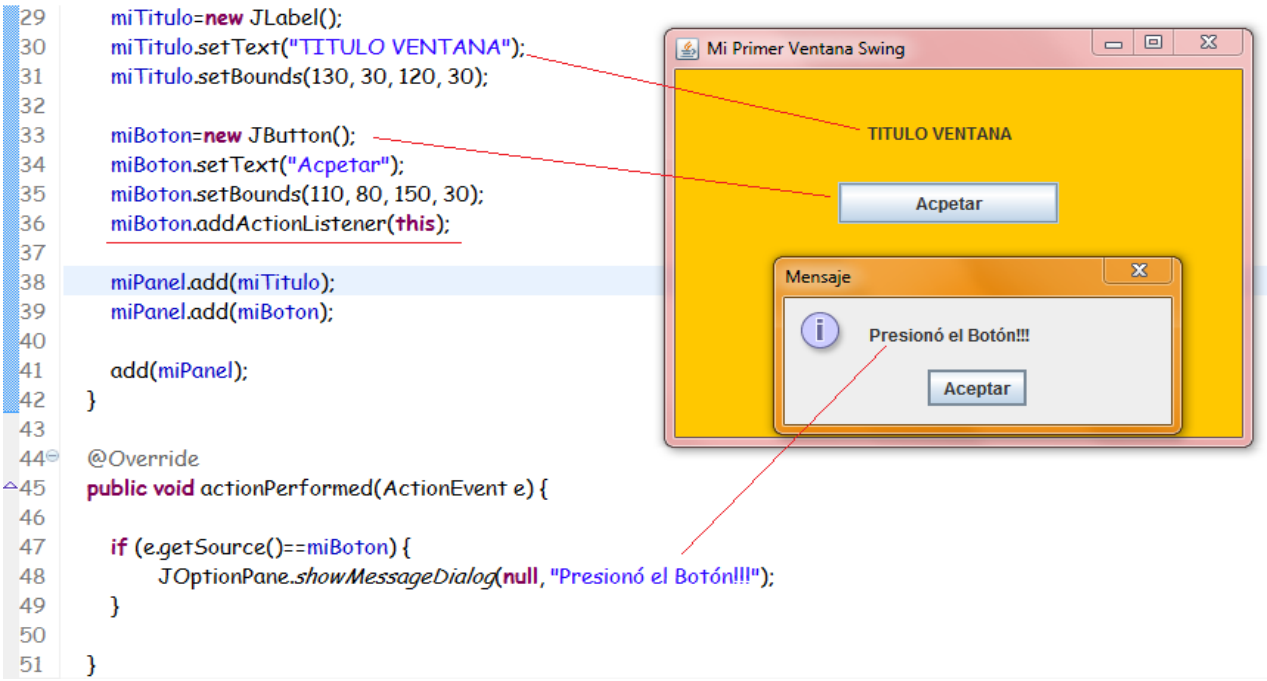



```
43    @Override
44    public void actionPerformed(ActionEvent e) {
45
46        if (e.getSource()==miBoton) {
47            JOptionPane.showMessageDialog(null, "Presionó el Botón!!!");
48        }
49
50    }
```

Por ultimo para que el botón pueda ser escuchado, se debe agregar al botón la propiedad addActionListener(this) que permite escuchar cuando se presiona el botón, sin esto el botón no funcionará.

```
miBoton.addActionListener(this);
```

Al final tendremos esto:




```
29        miTitulo=new JLabel();
30        miTitulo.setText("TITULO VENTANA");
31        miTitulo.setBounds(130, 30, 120, 30);
32
33        miBoton=new JButton();
34        miBoton.setText("Acpetar");
35        miBoton.setBounds(110, 80, 150, 30);
36        miBoton.addActionListener(this);
37
38        miPanel.add(miTitulo);
39        miPanel.add(miBoton);
40
41        add(miPanel);
42    }
43
44    @Override
45    public void actionPerformed(ActionEvent e) {
46
47        if (e.getSource()==miBoton) {
48            JOptionPane.showMessageDialog(null, "Presionó el Botón!!!");
49        }
50
51    }
```

Agregando un campo de Texto.

Uno de los principales usos de las interfaces graficas de usuario es para la creación de formularios de registro, en estos el uso de la clase JTextfield es fundamental, ya que permite vincular un campo de texto para el ingreso de información.

Vamos a modificar nuestra aplicación, agregándole un campo de texto y que al momento de presionar el botón muestre cual fue el dato ingresado en dicho campo.

El proceso de vincular el JTextfield es el mismo que usamos para vincular el JPanel, JLabel y Jbutton, se debe declarar previamente y luego en el método inicializarComponentes() agregar las propiedades necesarias.

```
JPanel miPanel;
JLabel miTitulo;
JButton miBoton;
JTextField miCampoDeTexto;
```

Para agregar el JTextField lo que se va a hacer es agregar el nuevo campo de texto, darle una posición y un tamaño en medio del título y el botón.

Para eso bajamos un poco el botón modificando la propiedad setBounds cambiando el alto que tenía 80 por alto de 130

```
miBoton.setBounds(110, 130, 150, 30);
```

Por ultimo no olvidar agregar el campo de texto al panel, al final se debe tener el siguiente código.

```
21  private void iniciarComponentes() {
22      miPanel=new JPanel();
23      miPanel.setLayout(null);
24      miPanel.setBackground(Color.orange);
25
26      miTitulo=new JLabel();
27      miTitulo.setText("TITULO VENTANA");
28      miTitulo.setBounds(130, 30, 120, 30);
29
30      miCampoDeTexto=new JTextField();
31      miCampoDeTexto.setBounds(110, 80, 150, 30);
32
33      miBoton=new JButton();
34      miBoton.setText("Aceptar");
35      miBoton.setBounds(110, 130, 150, 30);
36      miBoton.addActionListener(this);
37
38      miPanel.add(miTitulo);
39      miPanel.add(miCampoDeTexto);
40      miPanel.add(miBoton);
41
42      add(miPanel);
```

Obteniendo el dato del campo.

Obtener lo que el usuario ingresa en el campo se realiza de forma muy fácil tan solo por medio del método getText(), este método permite retornar la información introducida.

Por defecto el dato que obtiene es de tipo String, por lo tanto si quisiéramos ingresar datos numéricos se deberá hacer la conversión correspondiente de texto a int.

El dato lo vamos a obtener al momento de escuchar el evento del botón cuando imprimimos el mensaje por medio del JOptionPane

```
27      miTitulo.setText("TITULO VENTANA");
28      miTitulo.setBounds(130, 30, 120, 30);
29
30      miCampoDeTexto=new JTextField();
31      miCampoDeTexto.setBounds(110, 80, 150, 30);
32
33      miBoton=new JButton();
34      miBoton.setText("Aceptar");
35      miBoton.setBounds(110, 130, 150, 30);
36      miBoton.addActionListener(this);
37
38      miPanel.add(miTitulo);
39      miPanel.add(miCampoDeTexto);
40      miPanel.add(miBoton);
41
42      add(miPanel);
43  }
44
45  @Override
46  public void actionPerformed(ActionEvent e) {
47
48      if (e.getSource()==miBoton) {
49          JOptionPane.showMessageDialog(null, "El dato ingresado es: "+miCampoDeTexto.getText());
50      }
51  }
```