Laura Fernández Carballo

Unidad 1

CONFECCIÓN DE INTERFACES

2º Curso de DAM

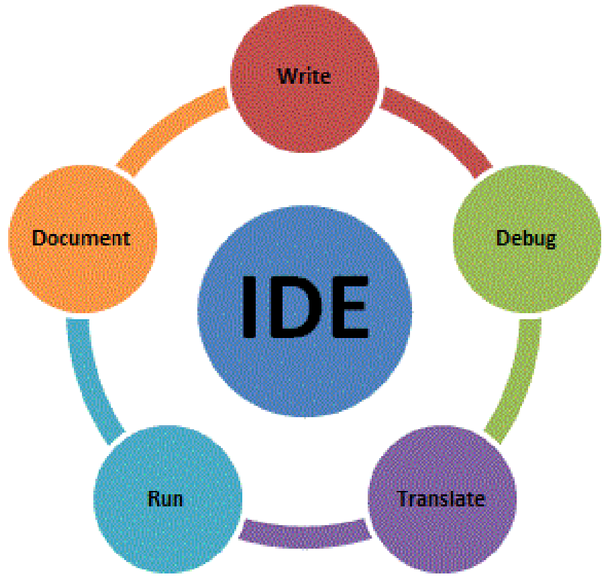
IES DE TEIS

3.1. Introducción(IDEs)

Existen diversos aplicaciones destinadas al desarrollo de interfaces. Estas aplicaciones se conocen con el acrónimo IDE (Integrated Development Environment)

Sus funcionalidades principales suelen integrar la **codificación**, **compilación**, **depuración, documentación** y **testing** de programas software.

Algunas de las soluciones IDE más populares son **NetBeans**, **Eclipse**, **Visual Studio** o **Aptana**.



Elaboración de interfaces

Tipos de interfaces de usuario:

- **Textuales**. La comunicación se produce por medio de la inserción de órdenes escritas en un intérprete de órdenes.
- **Gráficas**. La interfaz consta de un conjunto de elementos visuales, como iconos o menús con los que se interacciona, normalmente, mediante un elemento apuntador (el ratón, por ejemplo). Son las más habituales y tienen a gala haber popularizado el mundo de la informática para usuarios noveles.
- **Táctiles**. La comunicación se produce mediante a través de un dispositivo táctil, generalmente una pantalla que puede reaccionar ante la presión táctil de elementos apuntadores o incluso de los dedos. Se usan habitualmente en dispositivos móviles, terminales de puntos de venta y para el diseño de gráficos por ordenador.

A lo largo de esta unidad nos centraremos en la creación de interfaces gráficas. Veremos que una interfaz gráfica está formada por un conjunto de ventanas, llamadas formularios, y que dentro de ellos podemos colocar diferentes elementos visuales, que se denominan controles o componentes con los que al interactuar damos órdenes o podemos recibir información.

LIBRERÍA DE COMPONENTES

Java Foundation Classes

AWT

Primera biblioteca de Java para la creación de interfaces gráficas. Es común a todas las plataformas, pero cada una tiene sus propios componentes, escritos en código nativo para ellos. Prácticamente en desuso.

SWING

Surgida con posterioridad, sus componentes son totalmente multiplataforma porque no tienen nada de código nativo, tienen su precursor en AWT, de hecho, muchos componentes swing derivan de AWT, basta con añadir una J al principio del nombre AWT para tener el nombre swing, por ejemplo, el elemento Button de AWT tiene su correspondencia swing en Jbutton aunque se han añadido gran cantidad de componentes nuevos. Es el estándar actual para el desarrollo de interfaces gráficas en Java.

MSDN

.NET FRAMEWORK

Hace alusión tanto al componente integral que permite la compilación y ejecución de aplicaciones y webs como a la propia biblioteca de componentes que permite su creación. Para el desarrollo de interfaces gráficas la biblioteca incluye ADO.NET, ASP.NET, formularios Windows Forms y la WPF (Windows Presentation Fundation).

XML

También existen bibliotecas implementadas en lenguajes intermedios basados en tecnologías XML. Normalmente disponen de mecanismos para elaborar las interfaces y traducirlas a diferentes lenguajes de programación, para después ser integradas en la aplicación final.

OUTRAS

DirectX, GTK, QT...

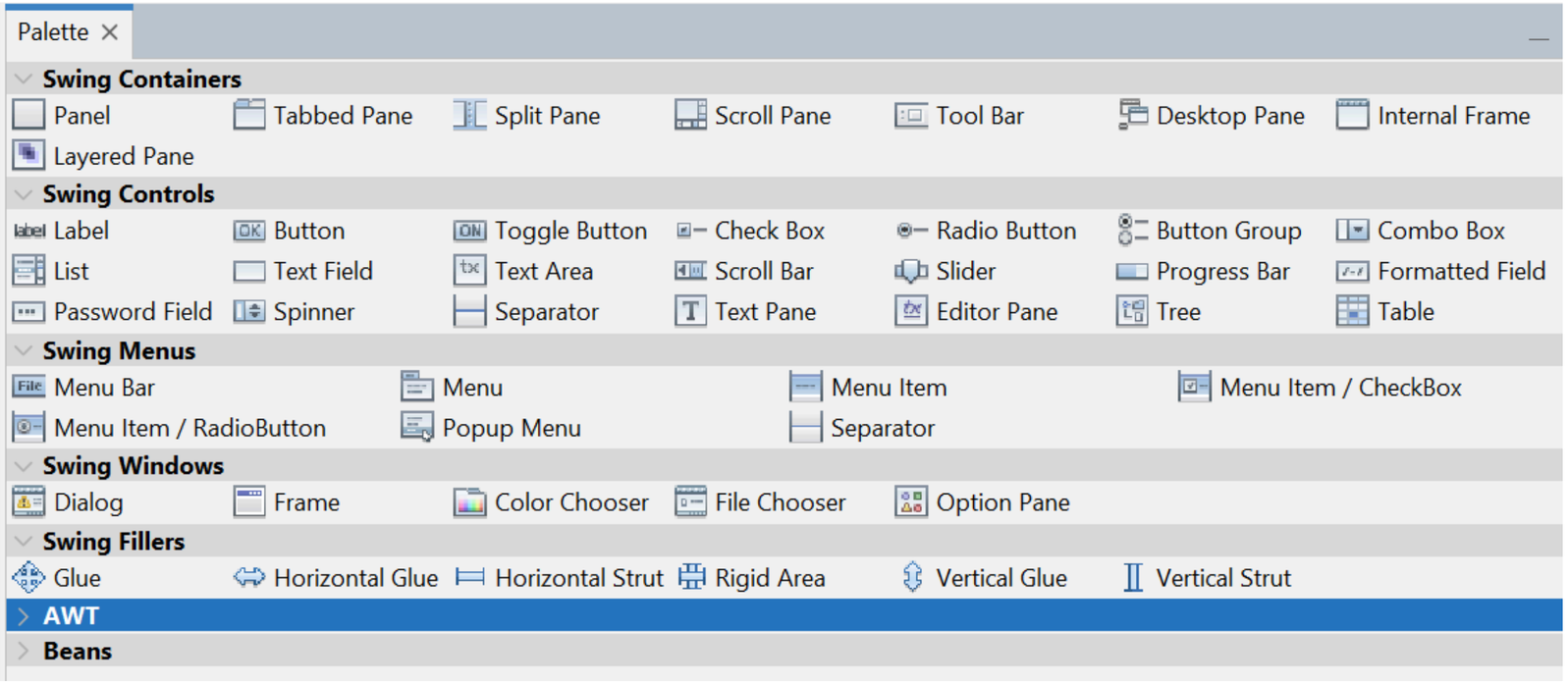
Componentes

Una interfaz gráfica se comporta como un todo para proporcionar un servicio al usuario permitiendo que éste realice **peticiones**, y **mostrando el resultado de las acciones realizadas por la aplicación**. Sin embargo, se compone de una serie de elementos gráficos atómicos que tiene sus propias características y funciones y que se combinan para formar la interfaz. A estos elementos se les llama componentes o controles.
Algunos de los componentes más típicos son:

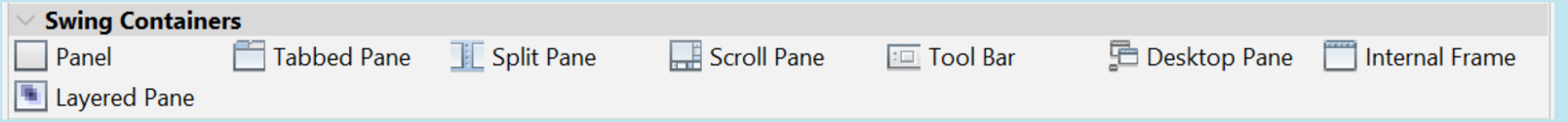
- **Etiquetas**: Permiten situar un texto en la interfaz. No son interactivos y puede utilizarse para escribir texto en varias líneas.
- **Campos de texto**: cuadros de una sola línea en los que podemos escribir algún dato.
- **Áreas de texto**: cuadros de varias líneas en los que podemos escribir párrafos.
- **Botones**: áreas rectangulares que se pueden pulsar para llevar a cabo alguna acción.
- **Botones de radio**: botones circulares que se presentan agrupados para realizar una selección de un único elemento entre ellos. Se usan para preguntar un dato dentro de un conjunto. El botón marcado se representa mediante un círculo.
- **Cuadros de verificación**: botones en forma de rectángulo. Se usan para marcar una opción. Cuando está marcada aparece con un tic dentro de ella. Se pueden seleccionar varios en un conjunto.
- **Imágenes**: se usan para añadir información gráfica a la interfaz.
- **Password**: es un cuadro de texto en el que los caracteres aparecen ocultos. Se usa para escribir contraseñas que no deben ser vistas por otros usuarios.
- **Listas**: conjunto de datos que se presentan en un cuadro entre los que es posible elegir uno o varios.
- **Listas desplegables**: combinación de cuadro de texto y lista, permites escribir un dato o seleccionarlo de la lista que aparece oculta y puede ser desplegada.

Componentes

Exemplo de compoñentes en NetBeans



Contenedores

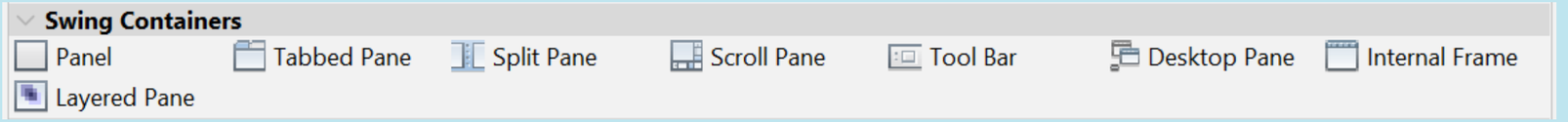


Antes de empezar a añadir elementos a la interfaz necesitarás **un sitio dónde ponerlos** por lo que lo primero que tendrás que hacer es seleccionar un contenedor y luego empezar a diseñar.

Un **formulario** es una ventana que dispone de **tres botones para minimizarse, maximizarse o cerrarse, una barra de título y está delimitado por unos bordes**.

Es la base para crear una aplicación de escritorio. Sobre él se añadirán controles o componentes, sencillos como botones o cajas de texto o más complejos, como barras de menú o rejillas de datos, que le dan funcionalidad.

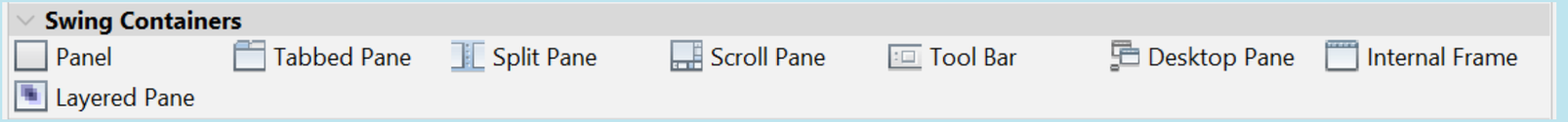
Contenedores



Una aplicación de escritorio de NetBeans **se compone de una serie de formularios**. Para **crear un formulario**, tendrás que **usar un contenedor Java** que es un componente que permite incluir otros componentes incluidos otros contenedores que se usarán para distribuir los controles. Por eso se dice que los contenedores forman una estructura jerárquica.

Un formulario está formado por un contenedor especial que se llama contenedor de nivel superior. Este tipo incluye un panel de contenido (contentpane) que permite añadir otros componentes, incluidos otros contenedores que se utilicen facilitar la distribución de elementos.

Contenedores



Como contenedor de nivel superior de un formulario puedes elegir entre una **ventana (JFrame), un diálogo (JDialog) o un applet (JApplet)**, según la necesidad. Todos estos componentes derivan, en la jerarquía de clases de java, de Window que representa una ventana típica.

- **Ventana (JFrame)**: es un formulario con título, los botones para maximizar, minimizar o cerrar y borde. Aparece un icono por defecto en forma de taza de café que puedes modificar, y puede contener una barra de menú.
- **Dialogo (JDialog)**: formularios que se suelen usar para solicitar información al usuario. Su principal característica es que pueden ser modales o no, una ventana modal recibe todas las entradas de usuario e impide que se active otra ventana.
- **Applet (JApplet)**: ventana que ejecuta una aplicación Java en el contexto de una página web.

Contenedores secundarios

Caso práctico:

Ana está inmersa en la creación de una interfaz, a pesar de que NetBeans le facilita mucho el proceso con las ayudas visuales, no puede terminar su interfaz puesto que el formulario que está construyendo es algo más complejo ya que necesitaría añadir algún tipo de panel con pestañas para mostrar diferentes controles en función de la selección del usuario, y esto no lo puede conseguir sólo con una ventana y los controles más sencillos. Decide investigar si puede usar otro tipo de contenedores que le resuelvan su problema.

Contenedores secundarios

También se interpretan como diálogos los siguientes componentes:

- **Panel de opciones (JOptionPane):** genera ventanas con botones para responder cuestiones con respuestas del tipo si-no, aceptar-cancelar, aceptar, etc.
- **Selector de archivos (JFileChooser):** permite seleccionar un archivo del sistema de archivos del equipo donde se ejecuta la aplicación.
- **Selector de colores (JColorChooser):** permite seleccionar entre un conjunto de colores y devolverlo usando el código adecuado.

Contenedores secundarios

Puedes usar otro tipo de contenedores para distribuir el resto de los controles que se incluyen en la ventana principales, entre los más habituales tienes:

- **Paneles (JPanel)**: representa un contenedor intermedio, cuya función principal es la de colocar controles.
- **Barra de menús (JMenu)**: permite la creación de menús complejos de opciones.
- **Barra de herramientas (JToolBar)**: se utiliza para contener iconos de acceso a las opciones de la aplicación.
- **Pestañas (JTabbedPane)**: tipo particular de panel que permite la distribución de elementos en pestañas o tarjetas.
- **Paneles deslizables (JScrollPane)**: tipo especial de panel que permite desplazar sus contenidos de manera automática.

Contenedores secundarios

- **Ventanas internas (JInternalFrame)**: ventanas hijas que no pueden rebasar los límites de la ventana padre donde se han creado. Se usan en aplicaciones que tienes varios documentos abiertos simultáneamente.
- **Paneles divididos (JSplitPane)**: permite visualizar dos componentes, uno a cada lado, asignando espacio dinámicamente a cada uno.

Añadir y eliminar componentes de la interfaz

Los componentes se pueden añadir desde la paleta, que, si recordamos suele anclarse a la derecha de la interfaz del IDE NetBeans.
A la derecha tienes la lista de controles para añadir a una interfaz en NetBeans. Están organizados en las siguientes categorías:

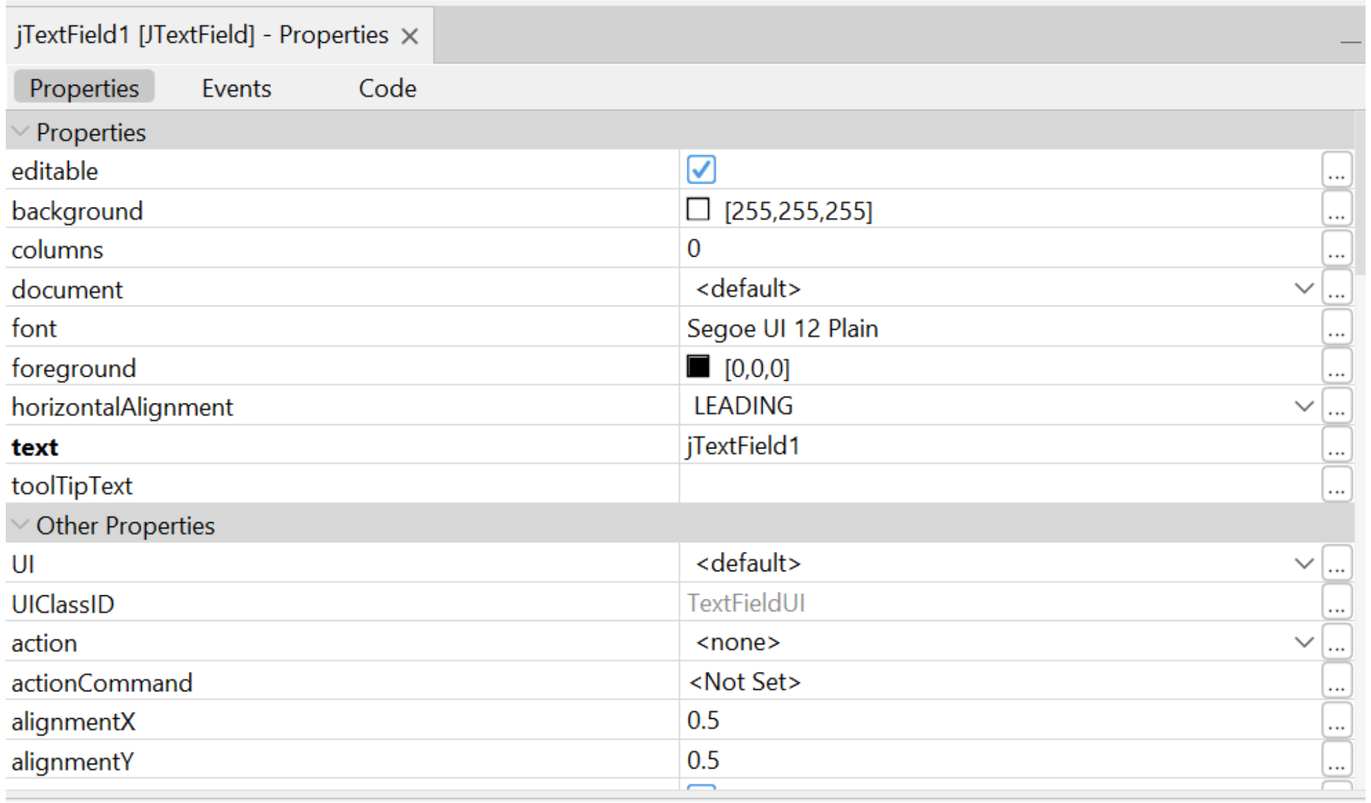
- **Contenedores swing**: son secundarios en la jerarquía de contenedores y se usan para distribuir y organizar el resto de controles.
- **Controles Swing**: básicos para crear una interfaz útil para comunicarse con el usuario y mostrar o solicitar información.
- **Menús Swing**: incluyen los controles necesarios para crear menús de aplicación complejos, con varios bloques, elementos activos e inactivos, etc y menús contextuales (Popup Menu)
- **Ventanas Swing**: permiten añadir a la aplicación ventanas (JFrame), diálogos (JDialog), selectores de ficheros y colores (JFileChooser y JColorChooser) y paneles de opciones (JOptionPane) para crear diálogos que se contestan con Sí/No.

Añadir y eliminar componentes de la interfaz

Para añadir componentes a la interfaz, seleccionaremos el control en la paleta y pinchando sobre la interfaz que se está construyendo. El control aparece en la interfaz con su aspecto por defecto que puedes modificar. Si arrastras el control se moverá sobre la superficie de su contenedor y si haces clic sobre una esquina y desplazas el ratón lo cambiarás de tamaño.
Al colocar un control sobre un formulario aparecen unas guías que te permiten colocarlo con más facilidad, esto será así mientras que tengas activa la opción diseño libre, lo puedes comprobar en el inspector haciendo clic con el botón secundario en el nodo raíz de la interfaz y seleccionando activar gestor de distribución. Si mueves un control estas guías te permitirán relacionarlo con otros componentes para que lo puedas alinear mejor.
Conforme vas colocando controles en el panel del formulario éstos aparecen reflejados, además, en el Inspector, de forma que los seleccionas tanto haciendo clic sobre ellos como sobre su nombre.
Trabajar con el inspector facilita colocar controles dentro de contenedores porque admite operaciones de arrastrar y soltar.
Para eliminar un control basta con seleccionarlo (de cualquiera de las formas descritas) y pulsar la tecla Supr, o bien seleccionar la opción Suprimir del menú contextual (el menú contextual aparece cuando pulsamos el botón derecho del ratón).

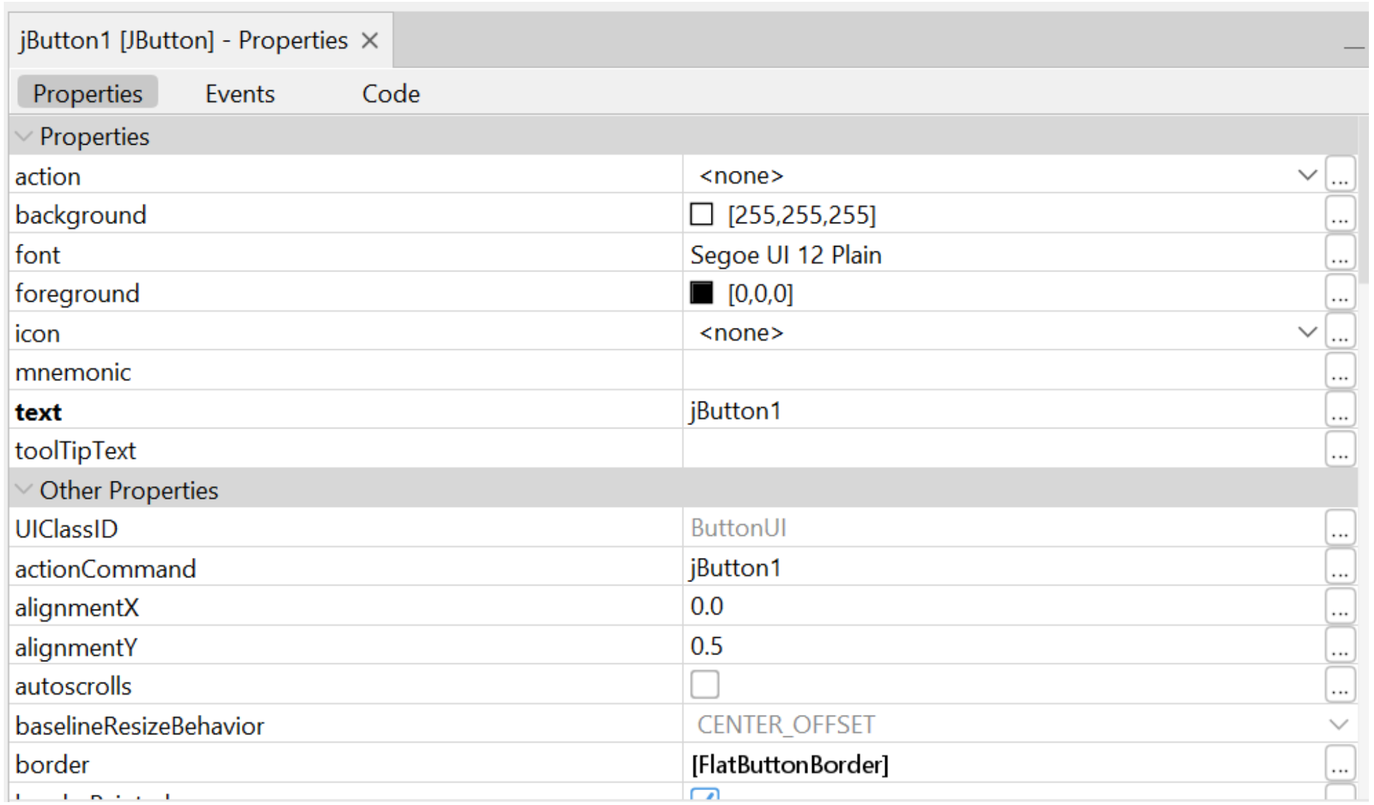
Modificar propiedades

Una vez que has colocado un control en el formulario puedes modificar sus propiedades para adaptarlo a tu interfaz. Con el control seleccionado accedes a sus propiedades en el panel propiedades, que lógicamente, será diferente para cada tipo de control.



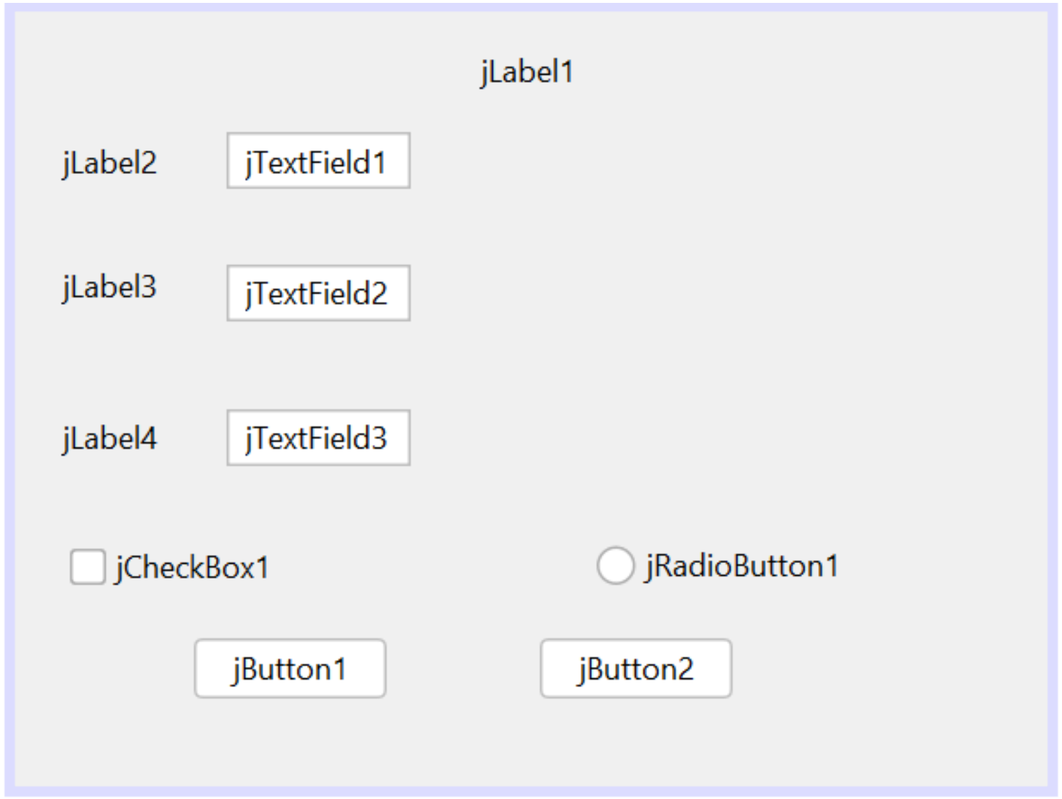
Exemplo de propiedades de un Botón

Realiza una u otra acción al ser pulsado



Tarefa 3

Realizar un pequeno formulario de rexistro.



Creación do seguinte formulario

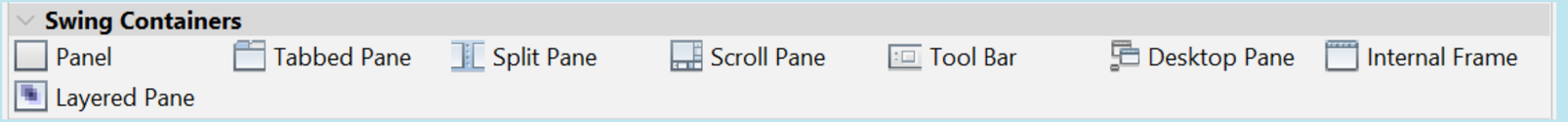
Para continuar necesitamos crear o seguinte formulario

Añadir funcionalidad desde NetBeans

Creamos interfaces para permitir que el usuario pueda interactuar con la aplicación, pero siempre será necesario añadir código para darles la funcionalidad para la que ha sido creadas.

Cuando usamos un entorno integrado como NetBeans parte de este código se genera de forma automática facilitando en gran medida el trabajo del desarrollador o desarrolladora, pero tendrás que abrir este código y modificar algunas cosas para que el formulario realice las tareas para las que ha sido diseñado.

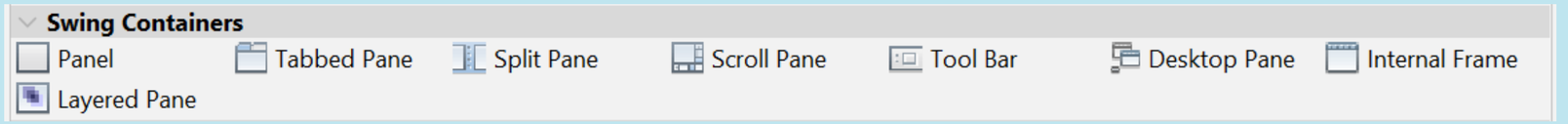
Ubicación y alineamiento de los componentes



Internamente la herramienta emplea el mecanismo de Java para disponer los elementos llamado Layout, distribución o diseño. Swing dispone de ocho tipos de distribuciones:

BorderLayout: aloja los componentes en los límites del formulario, por lo que cuando los colocamos debemos indicar si van al norte, sur este u oeste.
GridLayout: Diseña mediante una rejilla, en la que los componentes se organizan por filas y columnas.
GridBagLayout: semejante a GridLayout, pero permite a un componente que ocupe más de una celda.
CardLayout: diseño por paneles. Permite la colocación de distintos componentes en momentos distintos de la ejecución.
BoxLayout: diseño en caja. Coloca los componentes en una fila o columna ajustándose al espacio que haya.

Ubicación y alineamiento de los componentes



FlowLayout: diseña alojando los componentes de izquierda a derecha mientras quede espacio, si no queda pasa a la fila siguiente.
GroupLayout: se creó para ser utilizado en herramientas de diseño gráfico de interfaces. Trabaja por separado la distribución vertical y horizontal para definir exactamente el posicionamiento de los componentes. Se utiliza en NetBeans.

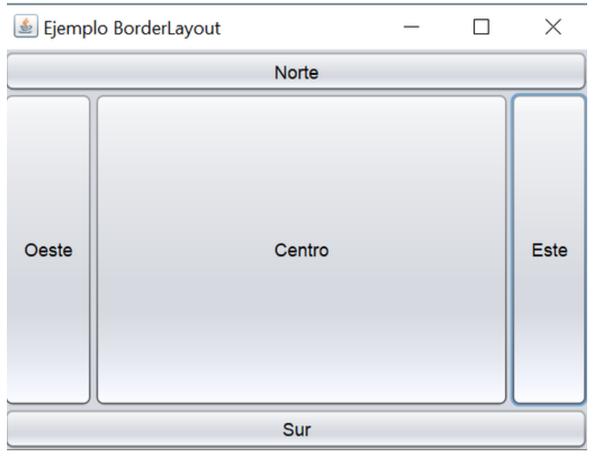
Programar el diseño de un formulario es una de las tareas más arduas en Java, si bien está ampliamente superado gracias al uso de IDEs que facilitan la colocación de componentes a golpe de ratón y sin necesidad de escribir código. Por ejemplo, NetBeans, usa el diseño GroupLayout.

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

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

Ubicación y alineamiento de los componentes

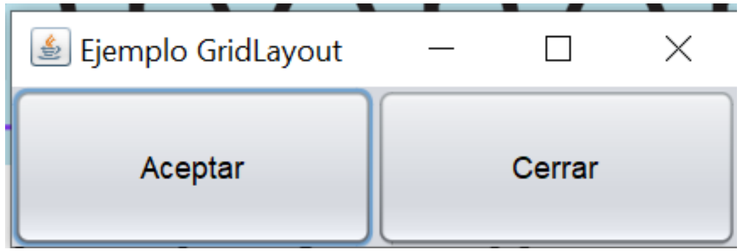
Ejemplo de Border Layout




```java
import javax.swing.*;
import java.awt.*;
/**
 *
 * @author laucarb
 */
public class NewJFrame extends javax.swing.JFrame {

    /**
     * Creates new form NewJFrame
     */

    public NewJFrame() {
        JFrame frame = new JFrame("Ejemplo BorderLayout");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        // Establecer el layout del frame como BorderLayout
        frame.setLayout(new BorderLayout());

        // Agregar componentes a las distintas regiones
        JButton botonNorte = new JButton("Norte");
        frame.add(botonNorte, BorderLayout.NORTH);

        JButton botonSur = new JButton("Sur");
        frame.add(botonSur, BorderLayout.SOUTH);

        JButton botonEste = new JButton("Este");
        frame.add(botonEste, BorderLayout.EAST);

        JButton botonOeste = new JButton("Oeste");
        frame.add(botonOeste, BorderLayout.WEST);

        JButton botonCentro = new JButton("Centro");
        frame.add(botonCentro, BorderLayout.CENTER);

        // Mostrar el frame
        frame.setVisible(true);
       // initComponents();
    }
```

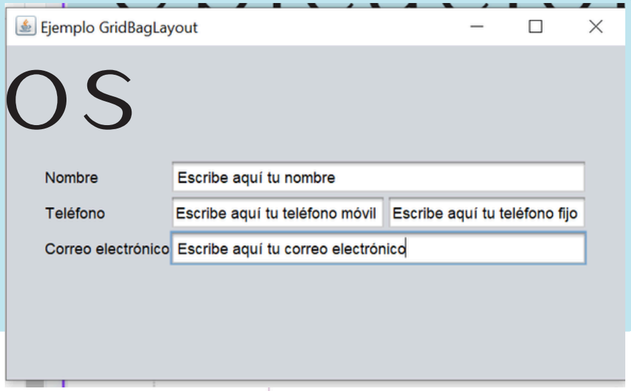
Ubicación y alineamiento de los componentes

Ejemplo de GridLayout




```
public EjGridLayout() {
    JFrame frame = new JFrame("Ejemplo GridLayout");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    //definimos el tamaño del frame
    frame.setSize(300, 100);

    // Layout del frame con 4 filas y 2 columnas
    frame.setLayout(new GridLayout(1, 2));


    frame.add(new JButton("Aceptar"));
    frame.add(new JButton("Cerrar"));
    // Mostrar el frame
    frame.setVisible(true);
}
```

Ubicación y alineamiento de los componentes



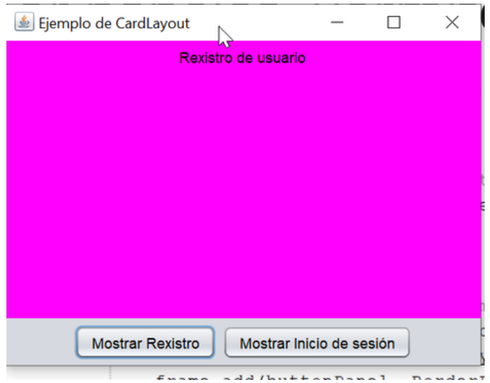
Ejemplo de GridBagLayout

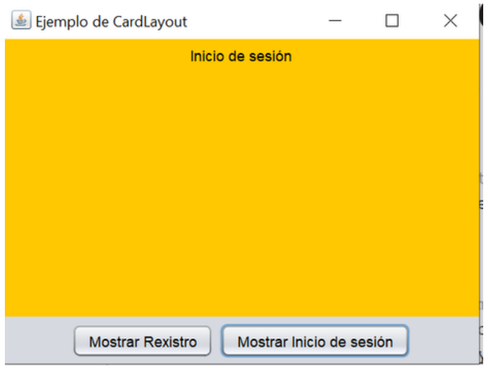
```
public EjGridBagLayout() {
    // Crear el frame
    JFrame frame = new JFrame("Ejemplo GridBagLayout");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(500, 300);

    // Establecer el layout del frame como GridBagLayout
    frame.setLayout(new GridBagLayout());

    // Crear el objeto GridBagConstraints
    GridBagConstraints gbc = new GridBagConstraints();
    gbc.fill = GridBagConstraints.BOTH; // Los componentes llenan el espacio disponible


    // creamos el componente de tipo JLabel en la posición 0,0
    gbc.gridx = 0;  // Columna 0
    gbc.gridy = 0;  // Fila 0
    gbc.gridwidth = 1;  // Número de columnas ocupadas
    gbc.gridheight = 1; // Número de filas ocupadas
    frame.add(new JLabel("Nombre"), gbc);

    // creamos el componente de tipo JTextField en la posición 1,0
    gbc.gridx = 1;  // Columna 1
    gbc.gridy = 0;  // Fila 0
    gbc.gridwidth = 10;  // Ocupa 2 columnas
    gbc.gridheight = 1; // Ocupa 1 fila
    frame.add(new JTextField("Escribe aquí tu nombre"), gbc);

    // creamos el componente de tipo JLabel en la posición 0,1
    gbc.gridx = 0;  // Columna 0
    gbc.gridy = 1;  // Fila 1
    gbc.gridwidth = 1;  // Ocupa 1 columna
    gbc.gridheight = 1; // Ocupa 1 fila
    frame.add(new JLabel("Teléfono"), gbc);

    // creamos el componente de tipo JTextField en la posición 1,1
    gbc.gridx = 1;  // Columna 1
    gbc.gridy = 1;  // Fila 1
    gbc.gridwidth = 1;  // Ocupa 1 columna
    gbc.gridheight = 1; // Ocupa 1 fila
    frame.add(new JTextField("Escribe aquí tu teléfono móvil"), gbc);

    // creamos el componente de tipo JTextField en la posición 2,1
    gbc.gridx = 2;  // Columna 2
    gbc.gridy = 1;  // Fila 1
    gbc.gridwidth = 1;  // Ocupa 1 columna
    gbc.gridheight = 1; // Ocupa 1 fila
    frame.add(new JTextField("Escribe aquí tu teléfono fijo"), gbc);

    // creamos el componente de tipo JLabel en la posición 0,2
    gbc.gridx = 0;  // Columna 0
    gbc.gridy = 2;  // Fila 2
    gbc.gridwidth = 1;  // Ocupa 1 columna
    gbc.gridheight = 2; // Ocupa 2 filas
    frame.add(new JLabel("Correo electrónico"), gbc);

    // creamos el componente de tipo JTextField en la posición 1,2
    gbc.gridx = 1;  // Columna 1
    gbc.gridy = 2;  // Fila 2
    gbc.gridwidth = 20;  // Ocupa 2 columnas
    gbc.gridheight = 1; // Ocupa 1 fila
    frame.add(new JTextField("Escribe aquí tu correo electrónico"), gbc);

    // Mostrar el frame
    frame.setVisible(true);
}
```

Ubicación y alineamiento de los componentes

Ejemplo de CardLayout: diseño por paneles. Permite la colocación de distintos componentes en momentos distintos de la ejecución.

```
public NewJFrame() {
    // Crear el frame principal y definir su tamaño
    JFrame frame = new JFrame("Ejemplo de CardLayout");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 300);//tamaño de la ventana

    // Crear el CardLayout
    CardLayout cardLayout = new CardLayout();
    JPanel cardPanel = new JPanel(cardLayout);  // Panel principal

    // Crear dos paneles para agregar al CardLayout
    JPanel panel1 = new JPanel();
    panel1.setBackground(Color.MAGENTA);
    panel1.add(new JLabel("Rexistro de usuario"));

    JPanel panel2 = new JPanel();
    panel2.setBackground(Color.ORANGE);
    panel2.add(new JLabel("Inicio de sesión"));

    // Agregar los paneles al contenedor del CardLayout
    cardPanel.add(panel1, "Rexistro de usuario");
    cardPanel.add(panel2, "Inicio de sesión");

    // Crear los botones para cambiar entre los paneles
    JButton button1 = new JButton("Mostrar Rexistro");
    JButton button2 = new JButton("Mostrar Inicio de sesión");

    // Agregar ActionListeners para los botones
    button1.addActionListener((ActionEvent e) -> {
        cardLayout.show(cardPanel, "Rexistro de usuario");
    });

    // Crear un panel para los botones
    JPanel buttonPanel = new JPanel();
    buttonPanel.add(button1);
    buttonPanel.add(button2);

    // Agregar los paneles al frame
    frame.setLayout(new BorderLayout());
    frame.add(cardPanel, BorderLayout.CENTER);
    frame.add(buttonPanel, BorderLayout.SOUTH);

    // Mostrar la ventana
    frame.setVisible(true);
    //initComponents();
}
```

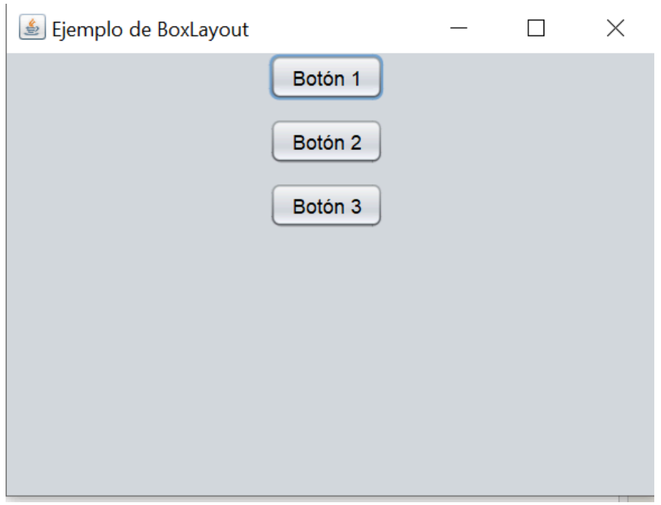




Ubicación y alineamiento de los componentes

Ejemplo de BoxLayout: diseño en caja. Coloca los componentes en una fila o columna ajustándose al espacio que haya.

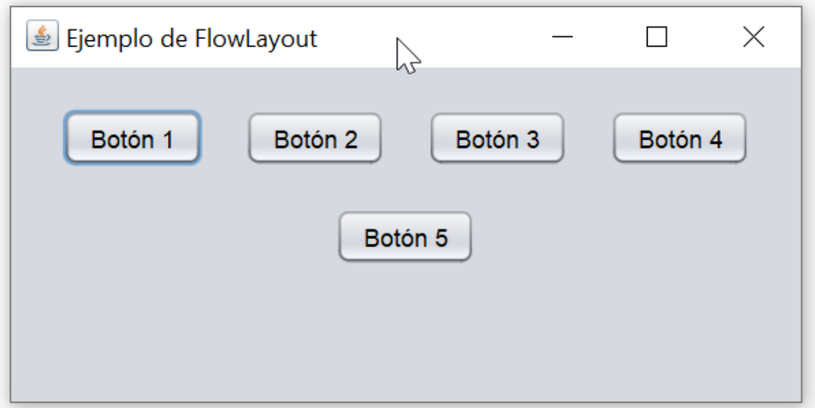
```java
JFrame frame = new JFrame("Ejemplo de BoxLayout");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(400, 300);

// Crear un panel con BoxLayout en el eje Y (vertical)
JPanel panel = new JPanel();
panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

// Crear botones y agregarlos al panel
JButton button1 = new JButton("Botón 1");
JButton button2 = new JButton("Botón 2");
JButton button3 = new JButton("Botón 3");

// Agregar un pequeño margen alrededor de cada botón
button1.setAlignmentX(Component.CENTER_ALIGNMENT);
button2.setAlignmentX(Component.CENTER_ALIGNMENT);
button3.setAlignmentX(Component.CENTER_ALIGNMENT);

// Agregar los botones al panel
panel.add(button1);
panel.add(Box.createRigidArea(new Dimension(0, 10)));  // Espacio vertical entre botones
panel.add(button2);
panel.add(Box.createRigidArea(new Dimension(0, 10)));  // Espacio vertical entre botones
panel.add(button3);

// Agregar el panel al frame
frame.add(panel);

// Mostrar la ventana
frame.setVisible(true);
```



Ubicación y alineamiento de los componentes

Ejemplo de FlowLayout

```
JFrame frame = new JFrame("Ejemplo de FlowLayout");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(400, 200);

// Crear un panel con FlowLayout (por defecto, los componentes estarán centrados)
JPanel panel = new JPanel(new FlowLayout(FlowLayout.CENTER, 20, 20));

// Crear botones y agregarlos al panel
JButton button1 = new JButton("Botón 1");
JButton button2 = new JButton("Botón 2");
JButton button3 = new JButton("Botón 3");
JButton button4 = new JButton("Botón 4");
JButton button5 = new JButton("Botón 5");

// Agregar los botones al panel
panel.add(button1);
panel.add(button2);
panel.add(button3);
panel.add(button4);
panel.add(button5);

// Agregar el panel al frame
frame.add(panel);

// Mostrar la ventana
frame.setVisible(true);
```

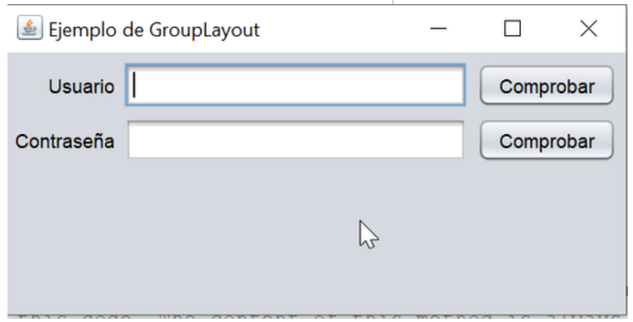


Ubicación y alineamiento de los componentes

Ejemplo de GroupLayout

```
// Crear el frame principal
JFrame frame = new JFrame("Ejemplo de GroupLayout");
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setSize(400, 200);

// Crear un panel para contener los componentes con GroupLayout
JPanel panel = new JPanel();

// Crear los componentes
JLabel label1 = new JLabel("Usuario");
JTextField textField1 = new JTextField(10);

JLabel label2 = new JLabel("Contraseña");
JTextField textField2 = new JTextField(10);

JButton button1 = new JButton("Comprobar");
JButton button2 = new JButton("Comprobar");

// Crear el GroupLayout y asignarlo al panel
GroupLayout groupLayout = new GroupLayout(panel);
panel.setLayout(groupLayout);

// Habilitar el tamaño automático de los gaps (espacios) entre componentes
groupLayout.setAutoCreateGaps(true);
groupLayout.setAutoCreateContainerGaps(true);
```

```
// Definir el diseño horizontal del GroupLayout
groupLayout.setHorizontalGroup(
    groupLayout.createSequentialGroup()
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.TRAILING)
            .addComponent(label1)
            .addComponent(label2))
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addComponent(textField1)
            .addComponent(textField2))
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.LEADING)
            .addComponent(button1)
            .addComponent(button2))
);

// Definir el diseño vertical del GroupLayout
groupLayout.setVerticalGroup(
    groupLayout.createSequentialGroup()
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
            .addComponent(label1)
            .addComponent(textField1)
            .addComponent(button1))
        .addGroup(groupLayout.createParallelGroup(GroupLayout.Alignment.BASELINE)
            .addComponent(label2)
            .addComponent(textField2)
            .addComponent(button2))
);

// Agregar el panel al frame
frame.add(panel);

// Mostrar la ventana
frame.setVisible(true);
```



JTextField

Clase JTextField

https://docs.oracle.com/javase/8/docs/api/javax/swing/JTextField.html

```
private JTextField nombre;
public NewJFrame() {
    JFrame frame = new JFrame("Formulario en Java");
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setSize(400, 300);

    // Crear un panel para contener los elementos del formulario
    JPanel panel = new JPanel();
    // Usamos un GridLayout para organizar los campos
    panel.setLayout(new GridLayout(4, 2));


    nombre=new JTextField();//creamos un objeto de tipo JTexTField
    nombre.setText("Escribe aquí tu nombre");

    panel.add(nombre);
    // Agregar el panel del formulario al frame
    frame.add(panel, BorderLayout.CENTER);

    frame.setVisible(true);
}
```

JTextField

```
public JTextField(String text)
```

Constructs a new `TextField` initialized with the specified text. A default model is created and the number of columns is 0.

```
public JTextField(String text)
```

Constructs a new `TextField` initialized with the specified text. A default model is created and the number of columns is 0.

```
public JTextField(int columns)
```

Constructs a new empty `TextField` with the specified number of columns. A default model is created and the initial string is set to `null`.

```
public JTextField(String text,
                  int columns)
```

Constructs a new `TextField` initialized with the specified text and columns. A default model is created.

JTextField

Métodos

public int getHorizontalAlignment()

setHorizontalAlignment(int alignment)

getColumns()

public void setColumns(int columns)

protected int getColumnWidth()

public void setFont(Font f)

public void addActionListener(ActionListener l)

public void removeActionListener(ActionListener l)

JTextField

Ejemplos de uso:

```
 JFrame frame = new JFrame("Ejemplo de BoxLayout");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(400, 300);

        // Crear un panel con BoxLayout en el eje Y (vertical)
        JPanel panel = new JPanel();
        panel.setLayout(new FlowLayout(FlowLayout.CENTER, 20, 20));

        JTextField campoTextoVacio=new JTextField();
        JTextField campoTextoConTexto=new JTextField("Escribe aquí tu nombre");
        JTextField campoTextoConTamano=new JTextField(10);
          JTextField campoTextoConTextoyTamano=new JTextField("Escribe aquí tu nombre"
```

Eventos y escuchadores

En un entorno GUI, cuando se produce un movimiento de ratón, se pulsa una tecla, se cierra una ventana, etc. se produce un evento.

El evento es recibido por un objeto, como puede ser un botón, una caja de texto, etc.

El objeto que recibe el evento, para poder responder a él, debe implementar la interfaz apropiada (event handler) y registrarla como un escuchador (event listener) en el componente GUI (event source u objeto que va recibir el evento) apropiado.

Los eventos se agrupan en función al componente que lo produce o a la acción que lo provoca.

El evento tiene los siguientes **elementos**:

- **La fuente del evento (event source)**: Es el componente que origina el evento.
- **El escuchador (event listener)**: es el encargado de atrapar o escuchar el evento.
- **El manejador del evento (event handler)**, es el método que permite implementar la interfaz.

Eventos y escuchadores

El manejador del evento recibe un objeto evento (ActionEvent) que contiene información sobre el evento que se ha disparado, cuando esto sucede, el manejador del evento descifra el evento con dicho objeto y procesa lo solicitado por el usuario o usuaria.

Un componente de una interfaz gráfica dispara o activa los manejadores (event handler) dependiendo del tipo de evento que ha ocurrido. El componente puede tener registrado más de un manejador que maneja el mismo tipo de evento. Un evento puede ser escuchado por más de un manejador de eventos.

El manejador del evento requiere que en la declaración de la clase que maneja el evento (event handler), se indique la interfaz correspondiente al evento (XListener, donde X es el tipo de evento a escuchar). La clase puede implementar más de un Xlistener. Si la clase no implementa la interfaz puede ser que extienda a una clase que si la implementa:

```
public class miClase implements ActionListener{...}
```

Eventos y escuchadores

En los componentes que son la fuente del evento (event source) se registra una instancia del manejador del evento (event handler), como un observador o escucha del tipo de eventos que maneja (XListener). Es decir, se le dice al componente que va a escuchar los eventos del tipo del manejador.

```
componente.addActionListener( instancia_de_miClaseListener);
```

En la clase que maneja el evento (event handler) se deben implementar los métodos de la interfaz XListener que descifren el evento (XEvent) y lo procesen.

```
public void actionPerformed(ActionEvent e) {
...//Código que reaccione a la acción
}
```

Escuchadores

Un escuchador (listener) es el objeto de la clase que implementa la interfaz de escucha de un evento y que contiene el método de respuesta al propio evento. Cuando se produce un determinado evento dentro del aplicación GUI, si se desea responder a esos eventos, la aplicación deberá implementar una serie de métodos que se ejecuten de forma automática cuando se produzca esos eventos. Los métodos que se van a implementar para responder a los diferentes eventos dentro de una aplicación de interfaz gráfica, se definen en unas interfaces denominadas interfaces de escucha.

Escuchadores

Cada interfaz de escucha contiene los métodos para gestionar un determinado grupo de eventos. La interfaz de escucha WindowsListener define el formato de los métodos para la gestión de los eventos de ventana, como pueden ser abrir la ventana, cerrarla, maximizarla, minimizarla etc. Algunos de los métodos de **WindowsListener** son:
public void windowActivated(WindowEvent e): Invocado cuando la ventana es la ventana activa.
public void windowDesactivated(WindowEvent e): Invocado cuando la ventana deja de ser la ventana activa.
public void windowClosing(WindowEvent e): Cuando la ventana se ha cerrado.
public void windowClosed(WindowEvent e): Cuando la ventana se minimiza.
public void windowIconified(WindowEvent e): Cuando la ventana se restablece.
public void windowDeiconified(WindowEvent e): La primera vez que la ventana se minimiza.
public void windowOpened(WindowEvent e): La primera vez que la ventana se hace visible.

Escuchadores

Cada interfaz de escucha contiene sus propios métodos para la gestión de eventos. Ejemplos de eventos que son generados por componentes de swing son:

ActionListener: Captura cierto tipo de acción realizada sobre ciertos componentes. Por ejemplo, pulsar un botón, seleccionar un elemento en una lista desplegable o una opción en un menú.

ChangeListener: Registra los cambios sobre ciertos componentes.

ItemListener: Recoge el cambio de estado en un componente tipo listas desplegables.

MouseListener: Eventos producidos por la manipulación del ratón.

MouseMotionListener: Movimiento del ratón sobre un componente.

ComponentListener: Visibilidad de componentes.

FocusListener: Captura eventos relacionados con la recepción o pérdida del foco.

KeyListener: Captura pulsaciones del ratón sobre el componente activo.

ListSelectionListener: Selección de elementos dentro de una lista.

Para responder a un evento dentro de una aplicación, deberás definir una clase que interprete la interfaz de escucha correspondiente al tipo de evento que quieras gestionar. A los objetos de esa clase de escucha se les denomina escuchadores.

Tipos de eventos

ActionEvent: generado por activación de componentes.
AdjustmentEvent: generado por ajuste de ajustables como barras de desplazamiento.
ContainerEvent: generado cuando los componentes se agregan o se quitan de un contenedor.
FocusEvent: generado cuando un componente entra o sale del foco.
ItemEvent: generado cuando un artículo se selecciona de una lista opción o caja de chequeo.
KeyEvent: generado por actividad del teclado.
MouseEvent: generado por actividad del ratón.
PaintEvent: generado cuando un componente se pinta.
TextEvent: generado cuando un componente del texto se modifica.
WIndowEvent: generado por actividad de la ventana.

Vayamos con los ejemplos

Ejemplo de ActionEvent

```
public Eventos() {
    // Crear un marco (ventana)
    JFrame frame = new JFrame("Mostrar mensaje en un diálogo");
    //defino el tamaño de la ventana
    frame.setSize(300, 150);
    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //layout de 1 fila y 2 columnas
    frame.setLayout(new GridLayout(1,2));
    // Crear un botón
    JButton boton = new JButton("Púlsame");
    //añado el botón al frame
    frame.add(boton);
    // Añadir un ActionListener al botón
    boton.addActionListener(new ActionListener() {
        @Override
        public void actionPerformed(ActionEvent e) {
            //Muestra un mensaje cuando hace clic en el botón
            JOptionPane.showMessageDialog(frame, "Hola!");
        }
    });
    frame.setVisible(true);
    //initComponents();
}
```

Vayamos con los ejemplos

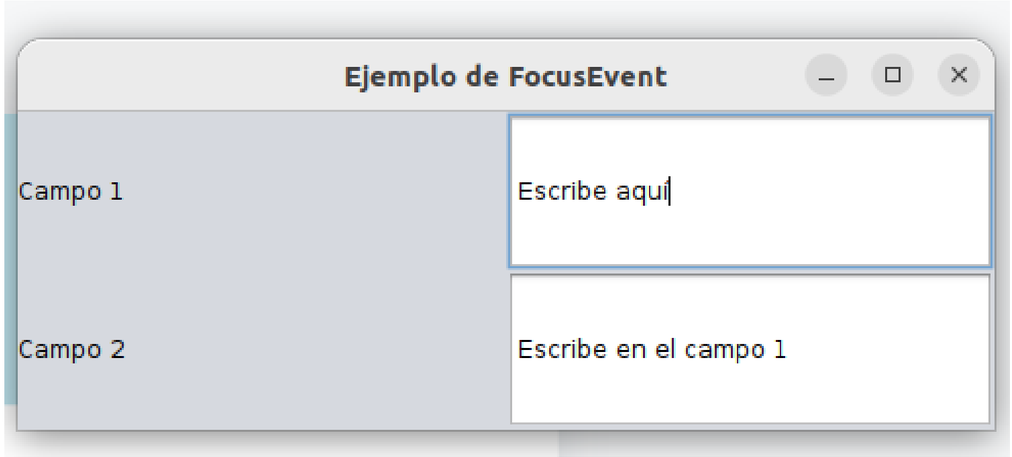
Ejemplo de FocusEvent

```
public EjFocusEvent() {
    // Configurar la ventana
    JFrame frame = new JFrame();

    frame.setSize(500,200);
    frame.setTitle("Ejemplo de FocusEvent");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    frame.setLayout(new GridLayout(2,2));

    //creo las etiquetas del formulario
    JLabel lCampo1=new JLabel("Campo 1");
    JLabel lCampo2=new JLabel("Campo 2");


    // Crear dos campos de texto
    JTextField campo1 = new JTextField("Campo 1", 15);
    JTextField campo2 = new JTextField("Campo 2", 15);

    // Añadir un FocusListener al primer campo de texto
    campo1.addFocusListener(new FocusListener() {
        @Override
        public void focusGained(FocusEvent e) {
            campo1.setText("Escribe aquí");
        }
        @Override
        public void focusLost(FocusEvent e) {
            campo1.setText("Escribe en el campo 2");
        }
    });

    // Añadir un FocusListener al segundo campo de texto
    campo2.addFocusListener(new FocusListener() {
        @Override
        public void focusGained(FocusEvent e) {
            campo2.setText("Escribe aquí");
        }

        @Override
        public void focusLost(FocusEvent e) {
           campo2.setText("Escribe en el campo 1");
        }
    });

    // Agregar las etiquetas y los campos de texto al frame
    frame.add(lCampo1);
    frame.add(campo1);
    frame.add(lCampo2);
    frame.add(campo2);

    //hacer visible la ventana
    frame.setVisible(true);
```

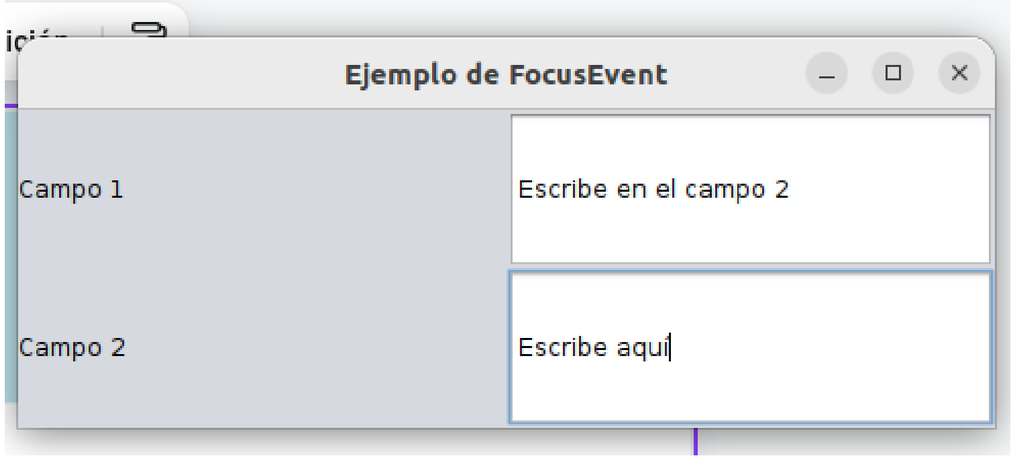
Vayamos con los ejemplos

Ejemplo de FocusEvent

Si pulso en el campo 1:



Si pulso en el campo 2:



Vayamos con los ejemplos

Ejemplo de ItemEvent

```
public EjItemEvent() {

    // Configurar la ventana
    JFrame frame = new JFrame();

    frame.setSize(500,100);
    frame.setTitle("Ejemplo de ItemEvent");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    //1 fila con 2 columnas
    frame.setLayout(new GridLayout(1,2));


    // Crear un panel para organizar los componentes

    // Crear un JCheckBox
    JCheckBox checkBox = new JCheckBox("Recibir suscripciones");

    // Añadir un ItemListener al JCheckBox
    checkBox.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent e) {
            if (e.getStateChange() == ItemEvent.SELECTED) {
                System.out.println("Deseo recibir suscripciones");
            } else {
                System.out.println("No deseo recibir suscripciones");
            }
        }
    });
```

```
    // Crear un JComboBox con opciones
    String[] opciones = {"Faro de Vigo", "La voz de Galicia", "El País"};
    JComboBox<String> comboBox = new JComboBox<>(opciones);

    // Añadir un ItemListener al JComboBox
    comboBox.addItemListener(new ItemListener() {
        @Override
        public void itemStateChanged(ItemEvent evento) {
            if (evento.getStateChange() == ItemEvent.SELECTED) {
                System.out.println(evento.getItem());
            }
        }
    });

    // Agregar los componentes al panel
    frame.add(checkBox);
    frame.add(comboBox);
    frame.setVisible(true);
    //initComponents();
```

Vayamos con los ejemplos

Ejemplo de MouseEvent

```
public EjMouseEvent() {
    // Configurar la ventana
    // Configurar la ventana
    JFrame frame = new JFrame();

    frame.setSize(500,100);
    frame.setTitle("Ejemplo de ItemEvent");

    frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    // Crear un panel para capturar eventos del ratón

    frame.setBackground(Color.LIGHT_GRAY);

    JPanel panel=new JPanel();
    // Etiqueta para mostrar la información del evento del
    JLabel label = new JLabel("Interactúa con el panel usa
    panel.add(label);
```

```
// Añadir un MouseListener para manejar los clics, entrada, salida, et
panel.addMouseListener(new MouseListener() {
    //pulsamos un botón del ratón
    @Override
    public void mouseClicked(MouseEvent e) {
        label.setText("Clic en ratón");
    }
    //se presiona  un botón del ratón
    @Override
    public void mousePressed(MouseEvent e) {
        label.setText("Ratón presionado");
    }

    //el ratón entra en la ventana
    @Override
    public void mouseEntered(MouseEvent e) {
        label.setText("El ratón ha entrado en la ventana");
    }
    //el ratón sale de la ventana
    @Override
    public void mouseExited(MouseEvent e) {
        label.setText("El ratçpm ha salido de la ventana");
    }
    //el botón pulsado en el ratón se libera
    @Override
    public void mouseReleased(MouseEvent me) {
        label.setText("Mouse Released");
    }

});
```

Vayamos con los ejemplos

Ejemplo de MouseEvent

```
    // Añadir un MouseMotionListener para manejar el movimiento del ratón
    panel.addMouseMotionListener(new MouseMotionListener() {
        @Override
        public void mouseMoved(MouseEvent e) {
            label.setText("Se ha movido el ratón");
        }

        @Override
        public void mouseDragged(MouseEvent e) {
            label.setText("Se ha  arrastrado el ratón");
        }
    });
    frame.add(panel);
    frame.setVisible(true);
nitComponents();
 }
```