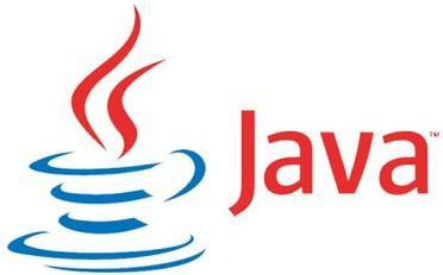
Interfaces gráficas “sencillas”



https://docs.oracle.com/javase/tutorial/uiswing/TOC.html

GUI en JAVA. JFC (Java Foundation Clases)

Históricamente AWT, basada en el sistema operativo

- Elementos pesados
- Filosofía multiplataforma comprometida
- Poco flexible

Posteriormente SWING

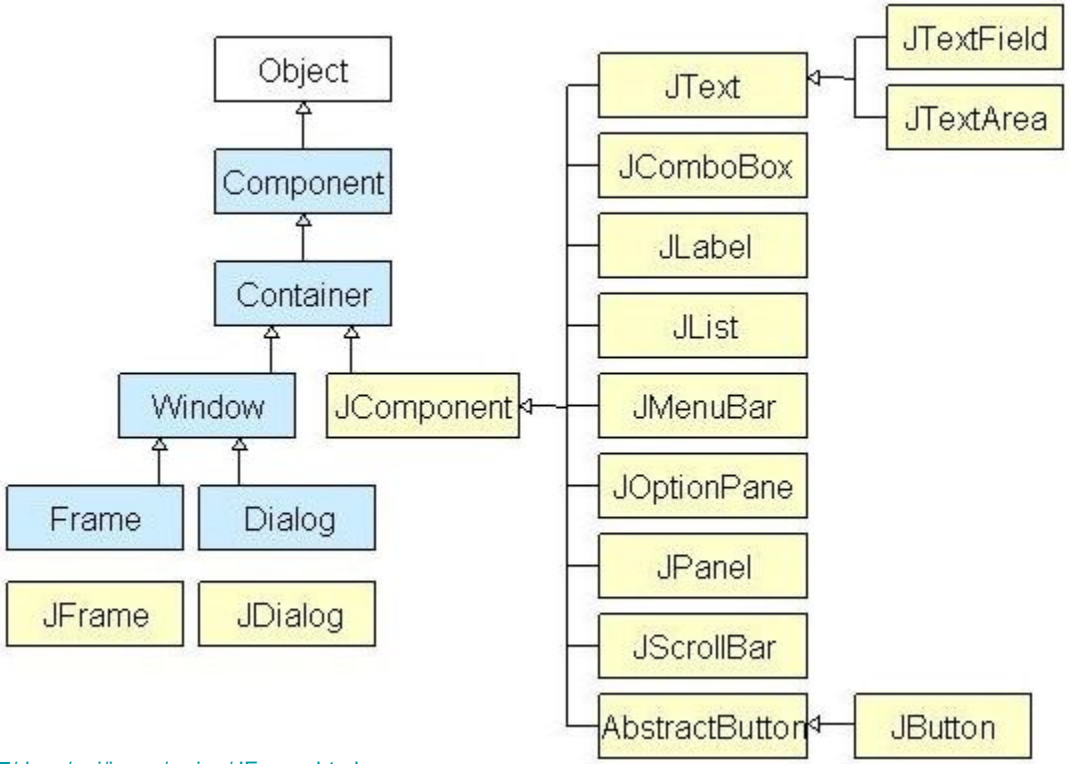
- Elementos ligeros
- Independiente de S.O.
- Se apoya en gran parte en AWT, no lo sustituye

JFC, agrupación de características para generar GUI’s en Java

JFC

Feature	Description
Swing GUI Components	Includes everything from buttons to split panes to tables. Many components are capable of sorting, printing, and drag and drop, to name a few of the supported features.
Pluggable Look-and-Feel Support	The look and feel of Swing applications is pluggable, allowing a choice of look and feel. For example, the same program can use either the Java or the Windows look and feel. Additionally, the Java platform supports the GTK+ look and feel, which makes hundreds of existing look and feels available to Swing programs. Many more look-and-feel packages are available from various sources.
Accessibility API	Enables assistive technologies, such as screen readers and Braille displays, to get information from the user interface.
Java 2D API	Enables developers to easily incorporate high-quality 2D graphics, text, and images in applications and applets. Java 2D includes extensive APIs for generating and sending high-quality output to printing devices.
Internationalization	Allows developers to build applications that can interact with users worldwide in their own languages and cultural conventions. With the input method framework developers can build applications that accept text in languages that use thousands of different characters, such as Japanese, Chinese, or Korean.

Jerarquía de Java Swing



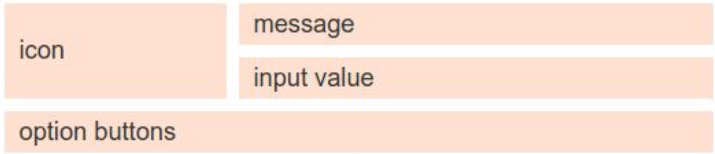
https://docs.oracle.com/javase/7/docs/api/javax/swing/JFrame.html

JOptionPane

Esta clase pertenece a swing y facilita lanzar ventanas de diálogo para pedir valores al usuario o informarle de algo.

Tiene muchos métodos pero se utilizan fundamentalmente los estáticos:

- showConfirmDialog
- showInputDialog
- showMessageDialog
- showOptionDialog



Los diálogos son modales (bloquean la aplicación hasta que terminan la interacción con el usuario).

https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/JOptionPane.html

Componentes y contenedores

Componente: Control visual independiente, como un botón o un campo.

Contenedor: Contiene un grupo de componentes (también es un componente)

Para que un componente pueda mostrarse debe almacenarse en un contenedor.

Todas las GUI tendrán al menos un contenedor.

Componentes

En general derivan de JComponent (menos cuatro excepciones que veremos)

Proporciona la funcionalidad común a todos los componentes. Hereda de Container y de Component de AWT

Están en el paquete

javax.swing

Ejemplos: JButton, JFrame, JLabel, JTextField, JCheckBox, JList, JRadioButton, JSeparator, JMenu...

Contenedores

Dos tipos de contenedores:

- De nivel superior:
  - No heredan de JComponent pero sí de Component y Container de AWT.
  - Son JFrame, JApplet, JWindow y JDialog
  - Son componentes pesados
  - Deben situarse en la cima de la jerarquía de contenedores, no se incluye dentro de ningún otro contenedor.
  - Toda jerarquía debe comenzar en un contenedor de nivel superior
  - El más utilizado es JFrame
- Contenedor ligero:
  - Heredan de JComponent
  - Ejemplos: JPanel, JScrollPane y JRootPane
  - Se pueden incluir en otro contenedor
  - Suelen utilizarse para organizar grupos de componentes relacionados

JRootPane

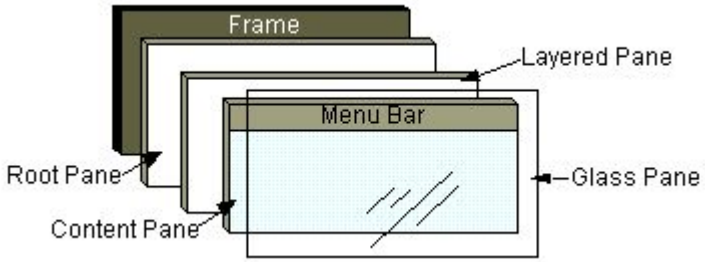
https://docs.oracle.com/en/java/javase/17/docs/api/java.desktop/javax/swing/JRootPane.html

Se encuentra en la parte superior de la jerarquía. Contenedor ligero que gestiona otros paneles.

Gestiona la barra de menús opcional

Lo forman los paneles:

- Panel de cristal (glassPane)
- Panel de contenidos (contentPane)
- Panel de capas (layeredPane)



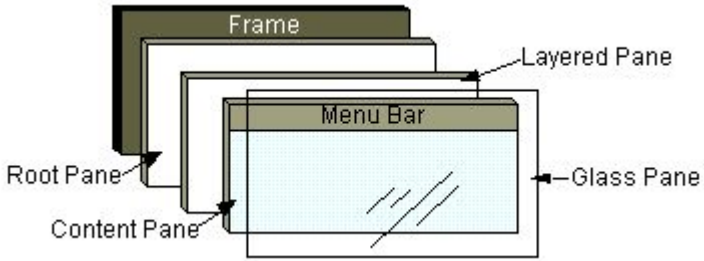
Habitualmente no necesitamos crearlo directamente, se genera al instanciar un JFrame JDialog

El panel de cristal (glassPane)

Es el de nivel superior, situado en la cima de los demás paneles, a los que oculta.

Le permite gestionar eventos del ratón que afectan a todo el contenedor (no a un control individual) o pintar sobre cualquier componente.

En muchos casos no será necesario utilizarlo



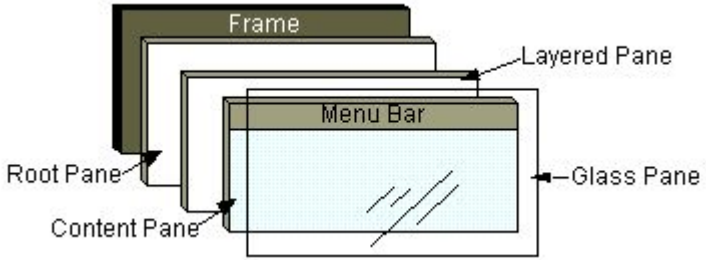
El panel de capas (layeredPane)

Permite asignar a los componentes un valor de profundidad.

Define qué componente solapa a otro.

Permite asignar un orden Z (no es habitual hacerlo)

Contiene el panel de contenidos y la barra de menú.



El panel de contenidos (contentPane)

Es el que recibe los componentes visuales

Es con el que más interactúa una aplicación

Ejemplo:

Cuando estemos usando un JFrame, podemos obtener su panel de contenidos con el método getContentPane(), que nos devuelve un Container (de awt)

Administradores de diseño (layouts)

Controla la posición de los componentes en un contenedor (distribución).

Son todos instancias de una clase que implementa la interface LayoutManager

Ejemplos:

- FlowLayout: Ubica los componentes de izquierda a derecha y de arriba a abajo (“como se lee”) cuidado con los ajustes de ventana.
- BorderLayout: Ubica los componentes en el centro o los bordes del contenedor. Es el diseño predeterminado del panel de contenido. Centro (predeterminado), N, S, E, W.
- GridLayout: Organiza los componentes en una cuadrícula.
- GridBagLayout: Organiza componentes de distinto tamaño en una cuadrícula flexible.
- BoxLayout: Organiza los componentes vertical u horizontalmente en un cuadro.
- SpringLayout: Organiza los componentes con respecto a una serie de restricciones.

Ejemplo

```
import javax.swing.*;

class Gui {
    public static void main(String args[]) {
        JFrame frame = new JFrame("Mi primera GUI");
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setSize(300, 300);
        JButton Boton = new JButton("Presionar");
        frame.getContentPane().add(Boton);
        frame.setVisible(true);
    }
}
```

```
import javax.swing.JFrame;
import javax.swing.JLabel;
public class HelloWorldSwing {
     private static void creaYMuestraGUI() {
          JFrame marco = new JFrame("Hola mundo");
          marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
          JLabel etiqueta = new JLabel("Hola mundo");
          marco.getContentPane().add(etiqueta);
          marco.pack();
          marco.setVisible(true);
     }
     public static void main(String[] args) {
          javax.swing.SwingUtilities.invokeLater(new Runnable() {
               public void run(){
                    creaYMuestraGUI();
               }
          });
     }
}
```

Manejo de eventos en swing

El mecanismo que utiliza swing es el “modelo de eventos delegados”

Una fuente genera un evento y lo envía a uno o varios escuchadores.

El escuchador espera hasta recibir un evento. Cuando llega, lo procesa y lo devuelve.

La lógica que procesa los eventos está separada de la interfaz de usuario que genera los eventos.

Eventos

En java un evento es un objeto que describe un cambio en una fuente de eventos.

La superclase para todos los eventos es java.util.EventObject.

Muchos eventos se declaran en java.awt.event

Los relacionados específicamente con swing se encuentran en javax.swing.event

Fuentes de eventos

Una fuente de eventos es un objeto que genera un evento.

Cuando lo hace envía este evento a todos los escuchadores registrados.

Para que un escuchador reciba un evento, tiene que estar registrado con la fuente del mismo.

Los escuchadores se registran con una fuente invocando un método en el objeto fuente del evento.

```
public void addTipoListener (TipoListener el);
```

Tipo es el nombre del evento y el una referencia al escuchador del evento

Para hacer que un escuchador deje de escuchar un tipo de evento, existe un método;

```
public void removeTipoListener(TipoListener el);
```

Los métodos que añaden o eliminan escuchadores los proporciona la fuente que genera eventos.

Ejemplo:

JButton es una fuente de ActionEvents, por lo que JButton proporciona métodos para añadir o eliminar un escuchador de acción.

Escuchadores de eventos

Un escuchador es un objeto que recibe una notificación cuando se produce un evento.

Debe:

1. Estar registrado con una o varias fuentes para recibir un evento específico.
2. Debe implementar un método para recibir y procesar ese evento.

Los métodos que reciben y procesan eventos están definidos en un conjunto de interfaces como las java.awt.event y javax.swing.event

Ej: La interfaz ActionListener define un método que maneja un ActionEvent. Cualquier objeto puede recibir y procesar este evento si implementa la interfaz ActionListener

Clases de eventos e interfaces de escuchador

En la raíz de la jerarquía de clases está EventObject, que está en java.util

AWTEvent de java.awt, es una subclase de EventObject.

Aunque swing usa eventos de AWT, también incorpora propios, que están en javax.swing.event

Hay muchos, pero los más importantes:

- ActionEvent: Generado cuando se produce una acción dentro de un control. Escuchador correspondiente: ActionListener
- ItemEvent: Generado cuando se selecciona algo. Escuchador: ItemListener.
- ListSelectionEvent: Generado cuando cambia la selección de una lista. Escuchador: ListSelectionListener

JButton

Hereda de AbstractButton, que define la funcionalidad de todos los botones

Puede incluir texto, una imagen o ambos

Sobrecarga de constructores, el más sencillo:

```
JButton(String nombre)
```

Al pulsarlo se genera un ActionEvent, que se define en AWT

JButton proporciona los siguientes métodos para añadir o eliminar un escuchador

```
void addActionListener(ActionListener al);
```

```
void removeActionListener(ActionListener al);
```

`al` especifica un objeto que recibe notificaciones de eventos. Debe ser una instancia de una clase que implemente la interfaz ActionListener

La interfaz ActionListener solo define un método; void actionPerformed(ActionEvent ae);

Del objeto actionEvent pasado a actionPerformed() se puede sacar información relacionada con el evento, por ejemplo el comando (que suele ser la cadena de texto) asociado con el botón.

```
String getActionCommand()
```

Al usar dos o más botones, este método me permite saber cuál de los botones se ha pulsado.

```
import javax.swing.*;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class EjemploBotonFuncional implements ActionListener {

	public EjemploBotonFuncional() {

		JFrame marco = new JFrame("El botón sabio");
		marco.setLayout(new FlowLayout());
		marco.setSize(300, 100);
		marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JButton botonAlfa = new JButton("Alfa");
		botonAlfa.addActionListener(this);
		marco.add(botonAlfa);
		marco.setVisible(true);
	}
	public void actionPerformed(ActionEvent ae) {

		JOptionPane.showMessageDialog(null, "Has pulsado alfa");
	}
	public static void main(String args[]) {

		EjemploBotonFuncional miEjemplo = new EjemploBotonFuncional();
	}
}
```

```
import javax.swing.*;
import java.awt.FlowLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class EjemploBoton {

	public static void main(String[] args) {

		JFrame marco = new JFrame("El marco");
		marco.setBounds(800, 400, 300, 100);
		marco.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		marco.setLayout(new FlowLayout());
		marco.setVisible(true);

		JButton boton1 = new JButton("Pínchame!");
		boton1.addActionListener(new Escuchador());

		marco.add(boton1);

	}
}

class Escuchador implements ActionListener {

	public void actionPerformed(ActionEvent ae) {

		JFrame marco2 = new JFrame("Esta es la respuesta");
		marco2.setBounds(200, 100, 400, 100);
		marco2.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);
		marco2.setVisible(true);

		JLabel etiqueta = new JLabel("Que tengas un buen día!! te desea " + ae.getActionCommand());
		marco2.add(etiqueta);

	}
}
```

Enlaces para el uso de Layouts

La referencia de oracle, visual

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

Un “resumen rápido”

https://www.cis.upenn.edu/~matuszek/cit591-2004/Pages/layout-examples.html

Ejemplos por tipos

https://www.tutorialspoint.com/swingexamples/swingexamples_layouts.htm