Introducción a Swing

¿Qué es Swing?

Swing proporciona componentes de interfaz gráfica de usuario (GUI) para desarrollar aplicaciones Java con una amplia gama de elementos gráficos, como ventanas, botones, casillas de verificación, etc.

Antes de definir una GUI, definimos primero una interfaz de usuario (UI).

Un programa realiza tres acciones:

- Acepta entradas del usuario
- Procesa las entradas
- Genera salidas

Interfaz de usuario (UI)

Una interfaz de usuario proporciona un medio para intercambiar información entre un usuario y un programa, tanto en términos de entradas como de salidas.

La interfaz de usuario define la forma en que se produce la interacción entre el usuario y un programa.

Escribir texto con el teclado, seleccionar un elemento de menú con el ratón o hacer clic en un botón pueden proporcionar entrada a un programa.

La salida de un programa puede mostrarse en un monitor de ordenador en forma de texto, un gráfico (como un diagrama de barras), una imagen, etc.

Interfaz de usuario (UI)

Una interfaz de usuario donde tanto la entrada del usuario como la salida del programa son texto se conoce como *interfaz de usuario basada en caracteres*.

Una *interfaz gráfica de usuario* (GUI) permite a los usuarios interactuar con un programa mediante elementos gráficos llamados *controles* o *widgets*, utilizando un teclado, un ratón y otros dispositivos.

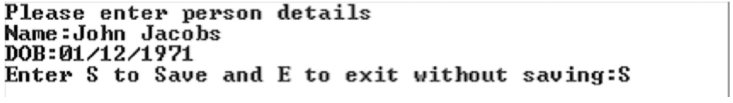


Figura 1-1. Un ejemplo de un programa con una interfaz de usuario basada en caracteres

La figura muestra un programa que permite a los usuarios ingresar el nombre y la fecha de nacimiento de una persona (DOB, *Date Of Birth*), y guardar la información mediante el teclado.

Este es un ejemplo de interfaz de usuario basada en caracteres.

La Figura 1-2 permite al usuario realizar las mismas acciones, pero utilizando una interfaz gráfica de usuario. Muestra seis elementos gráficos en una ventana.

Utiliza dos etiquetas (Nombre: y DOB:), dos campos de texto donde el usuario ingresará los valores de **NAME** y **DOB**, y dos botones (**`Save`**/Guardar y **`Close`**/Cerrar).

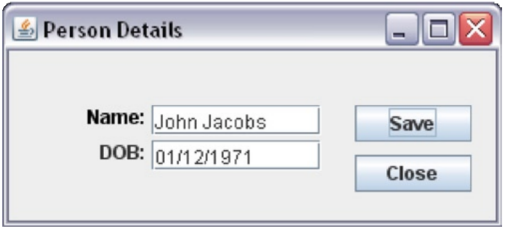


Figura 1-2. Un ejemplo de un programa con una interfaz gráfica de usuario

Una interfaz gráfica de usuario, en comparación con una interfaz de usuario basada en caracteres, facilita la interacción del usuario con un programa.

Un *contenedor* es un componente que puede contener otros componentes en su interior.

Un contenedor en el nivel más alto se llama *contenedor de nivel superior*.

Un **`JFrame`**, un **`JDialog`**, un **`JWindow`** y un **`JApplet`** son ejemplos de contenedores de nivel superior.

Un **`JPanel`** es un ejemplo de contenedor simple.

Un **`JButton`**, un **`JTextField`**, etc. son ejemplos de componentes.

En una aplicación Swing, cada componente debe estar contenido dentro de un contenedor.

El contenedor se conoce como **padre del componente** y el componente se conoce como **hijo del contenedor**.

Esta relación padre-hijo (o relación contenida en contenedor) se conoce como *jerarquía de contención*.

Para mostrar un componente en la pantalla, un contenedor de nivel superior debe estar en la raíz de la jerarquía de contención.

Cada aplicación **`Swing`** debe tener al menos un contenedor de nivel superior.

La Figura 1-3 muestra la jerarquía de contención de una aplicación **`Swing`**.

Un contenedor de nivel superior contiene un contenedor llamado *Contenedor 1*, que a su vez contiene un componente llamado *Componente 1* y un contenedor llamado *Contenedor 2*, que a su vez contiene dos componentes llamados *Componente 2* y *Componente 3*.

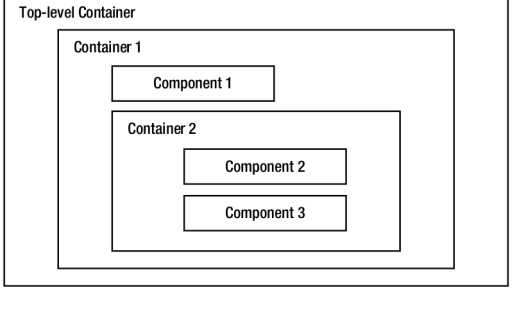


Figura 1-3. Jerarquía de contención en una aplicación **`Swing`**

El programa de Swing más simple

Comencemos con el programa **Swing** más simple.

Mostrará un **JFrame**, que es un contenedor de nivel superior sin componentes.

Para crear y mostrar un **JFrame**, debe hacer lo siguiente:

- Crea un objeto **JFrame**.
- Hazlo visible.

Para crear un objeto **JFrame**, puede utilizar uno de los constructores de la clase **JFrame**.

Uno de los constructores toma una cadena, que se mostrará como título del **JFrame**.

Las clases que representan los componentes **Swing** están en el paquete **javax.swing**, al igual que la clase **JFrame**.

El programa de `Swing` más simple

El siguiente fragmento de código crea un objeto **`JFrame`** con su título establecido en *Simplest Swing*:

```
// Create a JFrame object
JFrame frame = new JFrame("Simplest Swing");
```

Cuando crea un objeto **`JFrame`**, de forma predeterminada, no es visible.

Debe llamar a su método **`setVisible(boolean visible)`** para hacerlo visible.

Si pasa **`true`** a este método, el **`JFrame`** se hace visible y si pasa **`false`**, se vuelve invisible.

```
// Make the JFrame visible on the screen
frame.setVisible(true);
```

El programa de Swing más simple

¡Eso es todo lo que tienes que hacer para desarrollar tu primera aplicación **Swing!**

De hecho, puede envolver las dos declaraciones, para crear y mostrar un **JFrame**, en una sola declaración, así:

```
new JFrame("Simplest Swing").setVisible(true);
```

El programa de Swing más simple

El Listado con el código completo para crear y mostrar un **JFrame**

```
// SimplestSwing.java
import javax.swing.JFrame;

public class SimplestSwing {
        public static void main(String[] args) {
                // Create a frame
                JFrame frame = new JFrame("Simplest Swing");

                // Display the frame
                frame.setVisible(true);
        }
}
```

Listado 1-1. Programa de **Swing** más simple

Cuando se ejecuta este programa, se muestra un **JFrame** en la esquina superior izquierda de la pantalla.

Figura 1-4. El marco más sencillo

¿cómo se sale de una aplicación Swing?

Puede definir uno de los cuatro comportamientos de un **`JFrame`** para determinar qué sucede cuando se cierra el **`JFrame`**.

Se definen en la interfaz **`javax.swing.WindowsConstants`** como cuatro constantes.

La clase **`JFrame`** implementa la interfaz **`WindowsConstants`**.

Puede hacer referencia a todas estas constantes usando la sintaxis:

```
JFrame.CONSTANT_NAME
```

o puede usar la sintaxis

```
WindowsConstants.CONSTANT_NAME
```

Las cuatro constantes

- **`DO_NOTHING_ON_CLOSE`**: esta opción no hace nada cuando el usuario cierra un **`JFrame`**. Si configura esta opción para un **`JFrame`**, *debe proporcionar alguna otra forma de salir de la aplicación*, como un botón *Salir* o una opción de menú *Salir* en el **`JFrame`**.
- **`HIDE_ON_CLOSE`**: esta opción simplemente oculta un **`JFrame`** cuando el usuario lo cierra. **Este es el comportamiento predeterminado**.
  El **`JFrame`** simplemente se hizo invisible y el programa aún se estaba ejecutando.

Las cuatro constantes

- **`DISPOSE_ON_CLOSE`**: esta opción oculta y elimina el **`JFrame`** cuando el usuario lo cierra.
  La eliminación de un **`JFrame`** libera cualquier recurso a nivel del sistema operativo que utilice.
  Tenga en cuenta la diferencia entre **`HIDE_ON_CLOSE`** y **`DISPOSE_ON_CLOSE`**.

Las cuatro constantes

- EXIT_ON_CLOSE: Esta opción **sale de la aplicación**.
  Configurar esta opción funciona cuando un **`JFrame`** está cerrado, como si se hubiera llamado a **`System.exit()`**.
  Esta opción debe utilizarse con cierto cuidado. **Esta opción saldrá de la aplicación**. Si tiene más de un **`JFrame`** o cualquier otro tipo de ventana mostrada en la pantalla, usar esta opción para un **`JFrame`** cerrará todas las demás ventanas.
  Utilice esta opción con precaución, ya que puede perder los datos no guardados cuando cierre la aplicación.

Las cuatro constantes

Puede establecer el comportamiento de cierre predeterminado de un **`JFrame`** pasando una de las cuatro constantes a su método **`setDefaultCloseOperation()`** como se muestra:

```
// Exit the application when the JFrame is closed
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

Posición del Frame

`JFrame` se muestra sin área visible. Muestra solo la barra de título.

Debe establecer tamaño y posición del **`JFrame`** antes o después de que sea visible.

El tamaño se define por su ancho y alto en píxeles que se configura con **`setSize(int width, int height)`**

La posición está definida por las coordenadas (x, y) en **píxeles** (esquina superior izquierda del **`JFrame`**) con respecto a la esquina superior izquierda de la pantalla.

De forma predeterminada, su posición está establecida en (0, 0) y esta es la razón por la que **`JFrame`** se mostró en la esquina superior izquierda de la pantalla.

Posición del Frame

Puede establecer las coordenadas (x, y) del **`JFrame`** utilizando su método **`setLocation(int x, int y)`**

Para establecer el tamaño y posición en un solo paso: **`setBounds(int x, int y, int width, int height)`**

Puede colocar un **`JFrame`** en el centro llamando a su método **`setLocationRelativeTo()`** con un argumento nulo.

Componentes de un JFrame

Mostró un **`JFrame`** en el ejemplo anterior. Parecía vacío; sin embargo, **no estaba realmente vacío**.

Cuando crea un **`JFrame`**, las siguientes cosas se hacen automáticamente por usted:

- **Se agrega un contenedor**, que se denomina **panel raíz**/*root pane*, como único hijo del **`JFrame`**.
  El *root pane* es un contenedor.
  Es un objeto de la clase **`JRootPane`**.
  Puede obtener la referencia del panel raíz utilizando el método **`getRootPane()`** de la clase **`JFrame`**.

Componentes de un JFrame

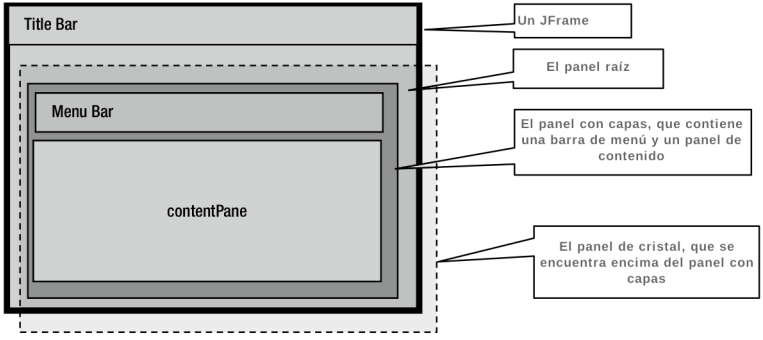
- Se agregan dos contenedores llamados **panel de vidrio**/*glass pane* y **panel en capas**/*layered pane* al panel raíz.
  De forma predeterminada, **el panel de vidrio está oculto y se coloca encima del panel en capas**.
  Como sugiere el nombre, el *panel de vidrio* es *transparente* e incluso si lo haces visible, puedes ver a través de él.
  El *panel en capas* recibe su nombre porque puede contener otros contenedores o componentes en sus diferentes capas.
  Opcionalmente, un *panel en capas* puede contener una barra de menú.
  Sin embargo, no se agrega una barra de menú de forma predeterminada cuando se crea un `JFrame`.
  Puede obtener la referencia del *panel de vidrio* y del *panel en capas* utilizando los métodos `getGlassPane()` y `getLayeredPane()` de la clase `JFrame`, respectivamente.

Componentes de un JFrame

- Se agrega un contenedor llamado ***panel de contenido***/*content pane* al *panel en capas*.
  De forma predeterminada, **el panel de contenido está vacío**.
  Este es el contenedor en el que se supone que debes agregar todos tus componentes **Swing**, como botones, campos de texto, etiquetas, etc.
  La mayor parte del tiempo, trabajarás con el panel de contenido del **`JFrame`**.
  Puede obtener la referencia del panel de contenido utilizando el método **`getContentPane()`** de la clase **`JFrame`**.

Componentes de un JFrame

La Figura 1-6 muestra el ensamblaje de un **`JFrame`**



El *panel raíz*, el *panel en capas* y el *panel de vidrio* cubren toda el área visible de un **`JFrame`**.

Componentes de un JFrame

El área visible de un **`JFrame`** es su tamaño menos las inserciones en los cuatro lados.

Los recuadros de un contenedor consisten en el espacio utilizado por el borde alrededor del contenedor en cuatro lados: superior, izquierdo, inferior y derecho.

Para un **`JFrame`**, el recuadro superior representa la altura de la barra de título.

Jerarquía de contención de un JFrame

La jerarquía de contención de un **`JFrame`** se enumera a continuación.

```
JFrame
        root pane
                glass pane
                layered pane
                        menu bar
                        content pane
```

Un **`JFrame`** está en la parte superior de la jerarquía, y la barra de menú (no se agrega de forma predeterminada; se muestra aquí para que esté completo) y el panel de contenido están en la parte inferior de la jerarquía de contención.

Puede obtener la referencia del panel de contenido de la siguiente manera:

```
// Create a JFrame
JFrame frame = new JFrame("Test");

// Get the reference of the content pane
Container contentPane = frame.getContentPane();
```

Agregar componentes a un JFrame

Utilice el método **`add()`** de un contenedor (tenga en cuenta que un panel de contenido también es un contenedor) para agregar un componente al contenedor

```
// Add aComponent to aContainer
aContainer.add(aComponent);
```

El método **`add()`** está sobrecargado.

Los argumentos del método, además del componente que se agrega, dependen de otros factores, como cómo desea que se coloque el componente en el contenedor.

Vamos a agregar un botón, que es un componente **Swing**, a un **`JFrame`**. Un objeto de la clase **`JButton`** representa un botón.

Un **`JButton`** contiene texto que también se denomina **etiqueta**.

```
// Create a JButton with Close text
JButton closeButton = new JButton("Close");
```

Agregar componentes a un JFrame

Para agregar **`closeButton`** al panel de contenido de un **`JFrame`**, debe hacer dos cosas:

- Obtenga la referencia del panel de contenido del **`JFrame`**
  `Container contentPane = frame.getContentPane();`
- Llame al método **`add()`** del panel de contenido.
  `contentPane.add(closeButton);`

Agregar componentes a un JFrame

Para agregar un **JButton** usando una línea de código, puede hacerlo combinando las tres declaraciones en una, así:

```
frame.getContentPane().add(new JButton(``Cerrar''));
```

Agregar componentes a un JFrame

```
// AddingComponentToJFrame.java
import javax.swing.JFrame;
import javax.swing.JButton;
import java.awt.Container;

public class AddingComponentToJFrame {
        public static void main(String[] args) {
                JFrame frame = new JFrame("Adding Component to JFrame");
                frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
                Container contentPane = frame.getContentPane();

                // Add a close button
                JButton closeButton = new JButton("Close");
                contentPane.add(closeButton);

                // set the size of the frame 300 x 200
                frame.setBounds(50, 50, 300, 200);
                frame.setVisible(true);
        }
}
```