Centro Asociado Palma de Mallorca

Introducción Práctica de Programación Java

Antonio Rivero Cuesta

Sesión

V

Interfaces Gráficas de Usuario

Java tiene dos bibliotecas para la construcción de interfaces gráficas de usuario:

- AWT
- Swing

Pasos básicos

1. El diseño y composición de la apariencia de la interfaz gráfica de la aplicación:

   - La elección de una ventana principal, se le llama ***contenedor***.
   - En esta ventana se van a incluir el resto de los elementos gráficos de interacción, son los ***componentes***.

- El diseño se completa con la elección del resto de los componentes gráficos, ***botones***, ***etiquetas***, ***menús***,… que se añaden a esa ventana principal, su ***disposición*** y la forma en que van a actuar.

2. Escribir el código que crea los componentes y a la apariencia de la interfaz.

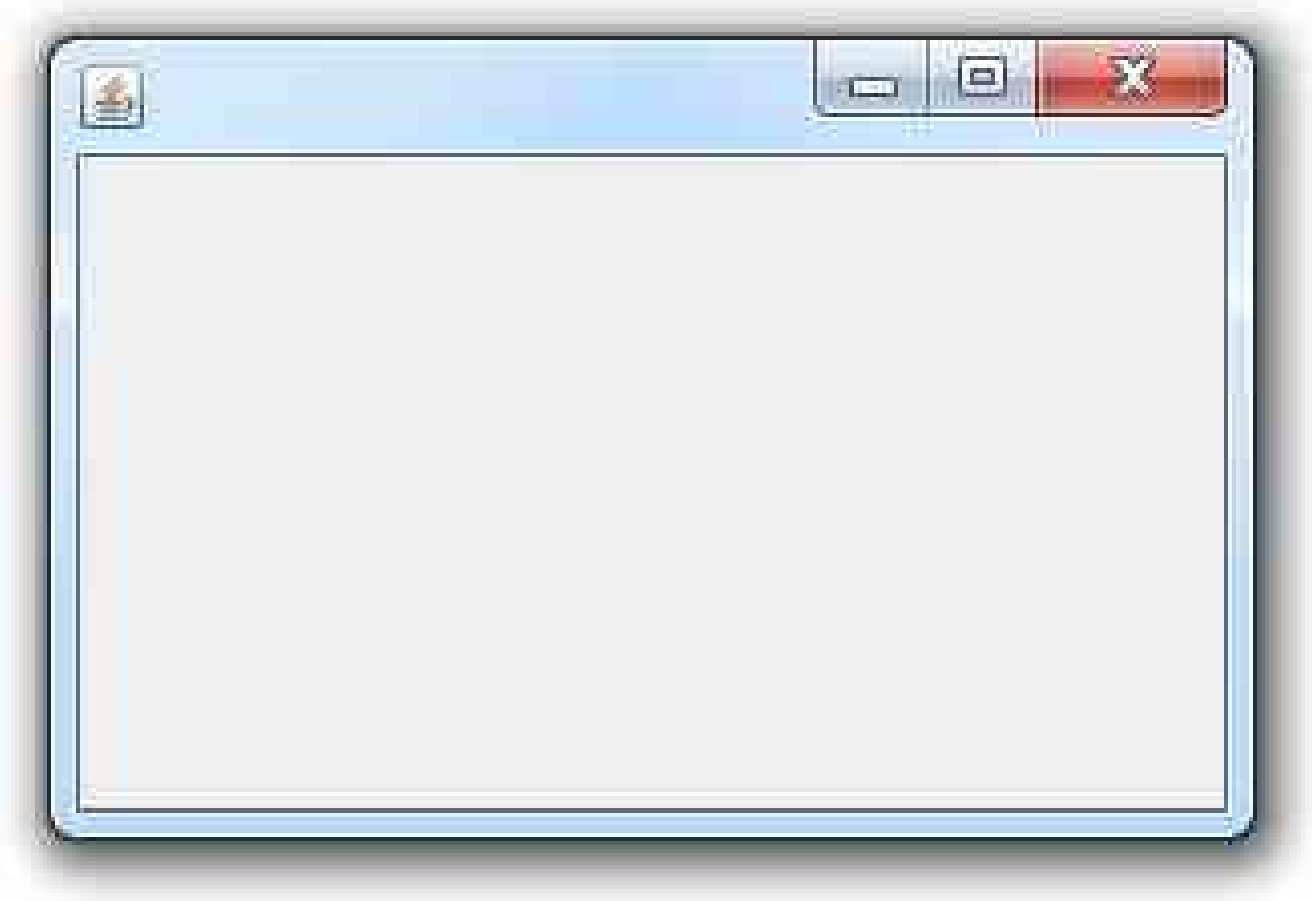
3. Escribir el código que proporciona el comportamiento de dicha interfaz como respuesta a las interacciones de los usuarios. Cada una de las acciones del usuario sobre la interfaz se traduce en la creación de un ***evento*** en *Java*, por ejemplo hacer clic en un botón.

4. La visualización de la interfaz gráfica.

Se acostumbra, pero no es obligatorio, declarar una subclase de JFrame y en el constructor llenar el Frame de componentes:

Vemos un ejemplo:

Ejemplo de interfaz gráfica



Código fuente

```
import javax.swing.JFrame;

public class JFrameBasico{
    public static void main(String[] args){
        JFrame f = new JFrame();
        f.setBounds(10,10,300,200);
        f.setVisible(true);
    }
}
```

Importamos la clase JFrame del paquete javax.swing:

```
import javax.swing.JFrame;
```

y luego en el método main definimos y creamos un objeto de la clase JFrame, llamando luego a los métodos **`setBounds`** y **`setVisible.`**

Lo que necesitamos para desarrollar una aplicación Swing se pueden dividir en tres áreas:

- ¿Qué clase de elementos podemos mostrar en una pantalla?
- ¿Cómo podemos acomodar estos elementos?
- ¿Cómo podemos reaccionar ante una entrada del usuario?

Discutiremos estas cuestiones mediante los términos:

- *Componentes.*
- *Gestores de disposición.*
- *Manejo de eventos.*

Componentes

Un componente es un objeto que tiene una representación gráfica que se puede visualizar en la pantalla y que se puede interactuar con el usuario.

Son las partes individuales a partir de las cuales se construye una GUI.

Son cosas tales como botones, menús, elementos de menú, cajas de verificación, deslizadores, campos de texto, etc.

La clase ***Component*** es la clase base de todos los componentes de una interfaz de usuario que no sea parte del menú.

Dicha clase hereda de la clase ***Object***.

Para colocar un componente es necesario que exista un contenedor.

Todos los componentes tienen al menos las funcionalidades heredadas de la clase ***Object*** y la clase ***Component***.

Gestores de disposición

Los ***gestores de disposición*** participan de cuestiones relacionadas con la ubicación de los componentes en la pantalla.

También se conocen como ***Layout Managers***.

Manejo de eventos

El ***manejo de eventos*** se refiere a la técnica que usaremos para trabajar con las entradas del usuario.

Una vez que hemos creado nuestros componentes y que los posicionamos en la pantalla, también tenemos que estar seguros de que ocurra algo cuando el usuario presione un botón.

Crear una ventana

Casi todo lo que se puede ver en una GUI está contenido en un tipo de ventana del más alto nivel.

Una ventana del nivel más alto es una ventana que está bajo el control del administrador de ventanas del sistema operativo y que típicamente puede moverse, cambiar de tamaño, minimizarse y maximizarse de manera independiente.

En Java, estas ventanas del más alto nivel se denominan ***frames*** y en ***Swing***, se representan mediante la clase de nombre JFrame.

Las primeras tres líneas que necesitamos escribir cuando queremos crear interfaces gráficas son:

```
java.awt
java.awt.event
javax.swing
```

Necesitamos varias de las clases de estos paquetes para todas las aplicaciones Swing que creemos.

La clase que declaramos debe de tener una ***variable de instancia*** de tipo JFrame que se usa para contener a la ventana que necesita el visor para mostrar las imágenes en la pantalla.

```
private JFrame ventana;
```

Necesitamos un método para crear la ventana Swing y su contenido.

La primera línea de este método es:

```
ventana = new JFrame ("Visor de Imágenes");
```

Esta sentencia crea una nueva ventana y la almacena en nuestra variable de instancia, para poder usarla más adelante.

Una ventana consta de tres partes:

- La ***barra del título***.
- Una ***barra de menú*** opcional.
- Un ***panel contenedor***.

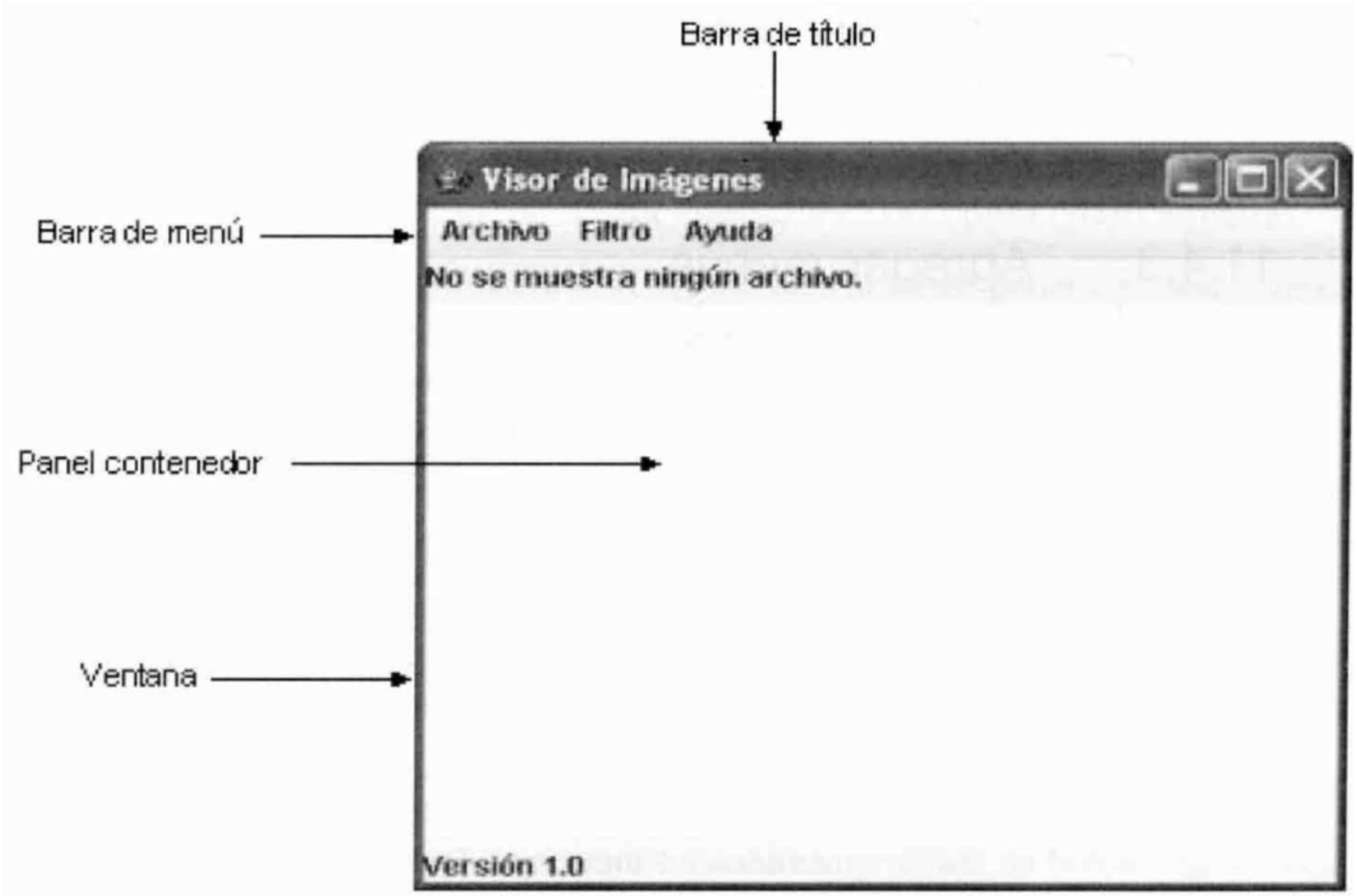
La apariencia exacta de la barra del título depende del sistema operativo que se esté usando.

Generalmente, contiene el título de la ventana y unos pocos controles para la ventana.

La barra de menú y el panel contenedor están bajo el control de la aplicación.

Podemos agregar algunos componentes en ambos para crear una GUI.

Nos concentraremos primero en el panel contenedor.


Barra de título
Visor de Imágenes
Barra de menú
Archivo Filtro Ayuda
No se muestra ningún archivo.
Panel contenedor
Ventana
Versión 1.0

A continuación agregamos una etiqueta al panel contenedor:

```
Container panelContenedor = ventana.getContentPane();

JLabel etiqueta = new JLabel("Soy una etiqueta.");

panelContenedor.add(etiqueta);
```

La primera línea obtiene el panel contenedor de la ventana.

Siempre debemos hacer esto.

Los componentes de la GUI se añaden a la ventana agregándolo al panel contenedor de la misma.

```
Container panelContenedor = ventana.getContentPane();
```

El panel contenedor es en sí mismo de tipo **`Container`**.

Un contenedor es un componente ***Swing*** que puede contener grupos arbitrarios de otro componente, prácticamente de la misma manera en que un **`ArrayList`** puede contener una colección arbitraria de objetos.

Luego, creamos un componente etiqueta de tipo **`JLabel`** y lo agregamos al panel contenedor.

Una etiqueta es un componente que puede mostrar texto o alguna imagen, o ambas cosas a la vez.

```
JLabel etiqueta = new JLabel("Soy una etiqueta.");

panelContenedor.add(etiqueta);
```

Finalmente, tenemos las dos líneas

```
ventana.pack();
```

```
ventana.setVisible(true);
```

La primera línea hace que la ventana distribuya adecuadamente los componentes dentro de ella y le asigne el tamaño apropiado.

Siempre tenemos que invocar el método pack sobre la ventana después de haber agregado o modificado el tamaño de u componentes.

La última línea finalmente hace que la ventana se vuelva visible en la pantalla.

```
ventana.setVisible(true);
```

Siempre comenzamos con una ventana que inicialmente es invisible, por lo que podemos acomodar todos los componentes dentro de ella sin que este proceso sea visible en la pantalla.

Luego, cuando la ventana esté construida, podemos mostrarla en un estado completo.

Agregar menús

Nuestro próximo paso en la construcción de una GUI es agregar menús y elementos de menú.

Esto es conceptualmente fácil pero contiene un detalle delicado:

¿Cómo nos arreglaremos para reaccionar a las acciones del usuario como por ejemplo, a la selección de un elemento de un menú?

Lo veremos en el ***manejo de eventos***.

Primero, creamos los menús.

Las tres clases involucradas en esta tarea son:

- **`JMenuBar`**
- **`JMenu`**
- **`JMenuItem`**

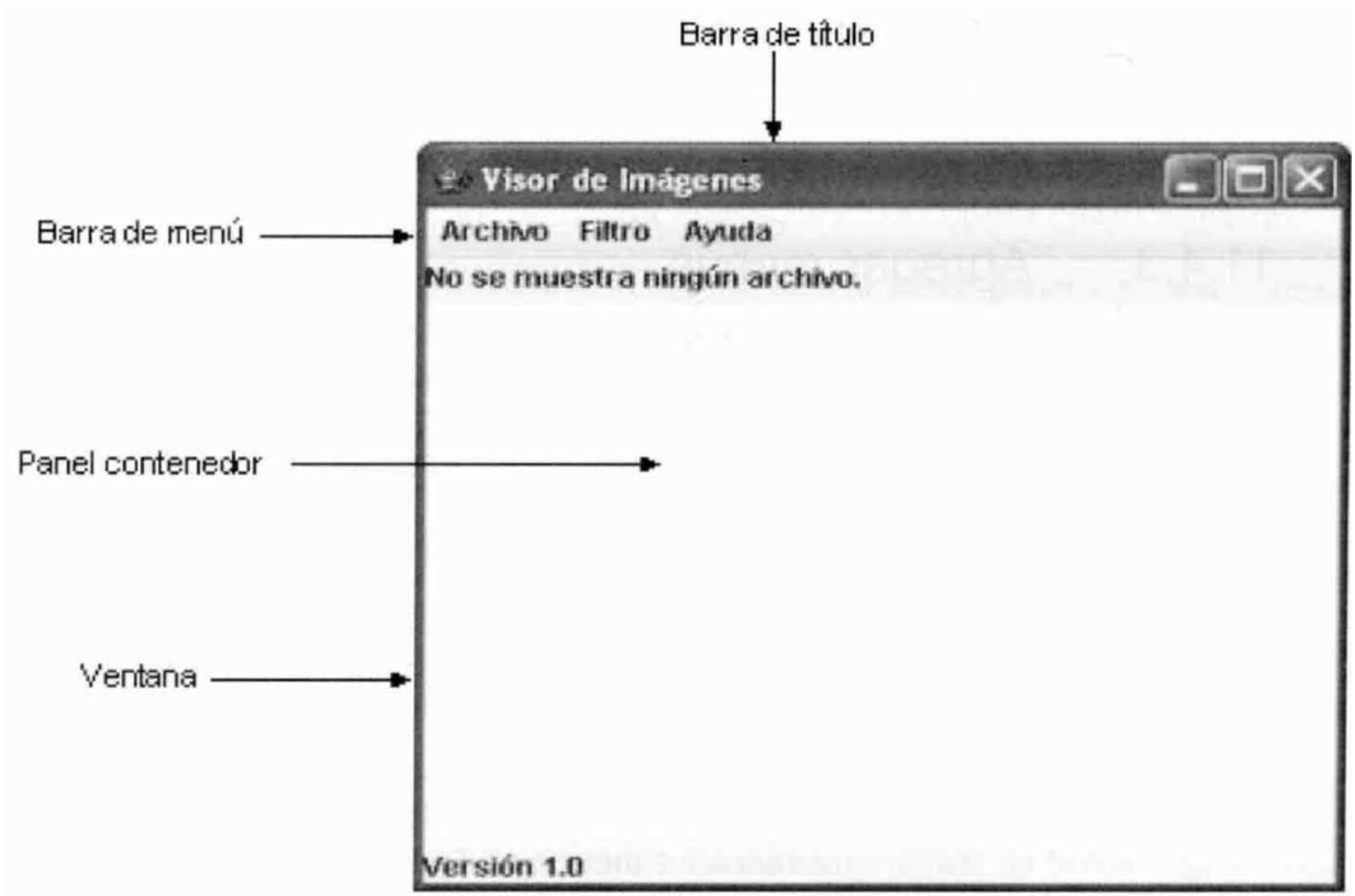
`JMenuBar` - Un objeto de esta clase representa una barra de menú que se puede mostrar debajo de la barra de título, en la parte superior de una ventana (véase la Figura 11.3).

Cada ventana tiene un **`JMenuBar`** como máximo.

`JMenu` - Los objetos de esta clase representan un solo menú (como por ejemplo, los menús comunes «Archivo», «Edición» o «Ayuda»).

Los menús frecuentemente están contenidos en una barra de menú; también pueden aparecer en menús emergentes, pero ahora no haremos esto.

`JMenuItem` - Los objetos de esta clase representan un solo elemento de menú dentro de un menú, como por ejemplo, «Abrir» o «Grabar>.


Barra de título
Visor de Imágenes
Barra de menú
Archivo Filtro Ayuda
No se muestra ningún archivo.
Panel contenedor
Ventana
Versión 1.0

Para nuestro visor de imágenes, crearemos una barra de menú y varios menús y elementos de menú.

La clase **`JFrame`** tiene un método de nombre **`setJMenuBar`**.

Podemos crear una barra de menú y usar este método para adjuntar nuestra barra de menú a la ventana:

```
JMenuBar barraDeMenu = new JMenuBar();
ventana.setJMenuBar(barraDeMenu);
```

Ahora estamos listos para crear un menú y agregarlo a la barra de menú:

```
JMenu menuArchivo = new JMenu(“Archivo”);
        barraDeMenu.add(menuArchivo);
```

Estas dos líneas crean un menú con la etiqueta «Archivo» y lo insertan en la barra de menú.

Finalmente, podemos agregar elementos al menú.

Las siguientes líneas agregan dos elementos con las etiquetas «Abrir> y «Salir> al menú «Archivo».

```
JMenuItem elementoAbrir = new JMenuItem(Abrir);
menuArchivo.add(elementoAbrir);
JMenuItem elementoSalir = new JMenuItem(Salir);
menuArchivo.add(elementoSalir);
```

Hasta ahora, hemos llevado a cabo la mitad de nuestra tarea.

Podemos crear y mostrar menús pero falta la segunda mitad.

Todavía no ocurre nada cuando un usuario selecciona un menú.

Ahora tenemos que agregar código para reaccionar a las selecciones del menú.

Lo vemos en el manejo de eventos.

JFrame y Contenedores

Las ventanas de la librería “javax.swing” se engloban en la clase JFrame.

Lo único que tenemos que hacer es extender la clase principal de nuestro programa con JFrame quedando así:

```
public class Hola extends JFrame
```

¿Que son los contenedores?

Son componentes que permiten almacenar, alojar o contener otros elementos gráficos.

Es el Tapiz donde vamos a pintar.

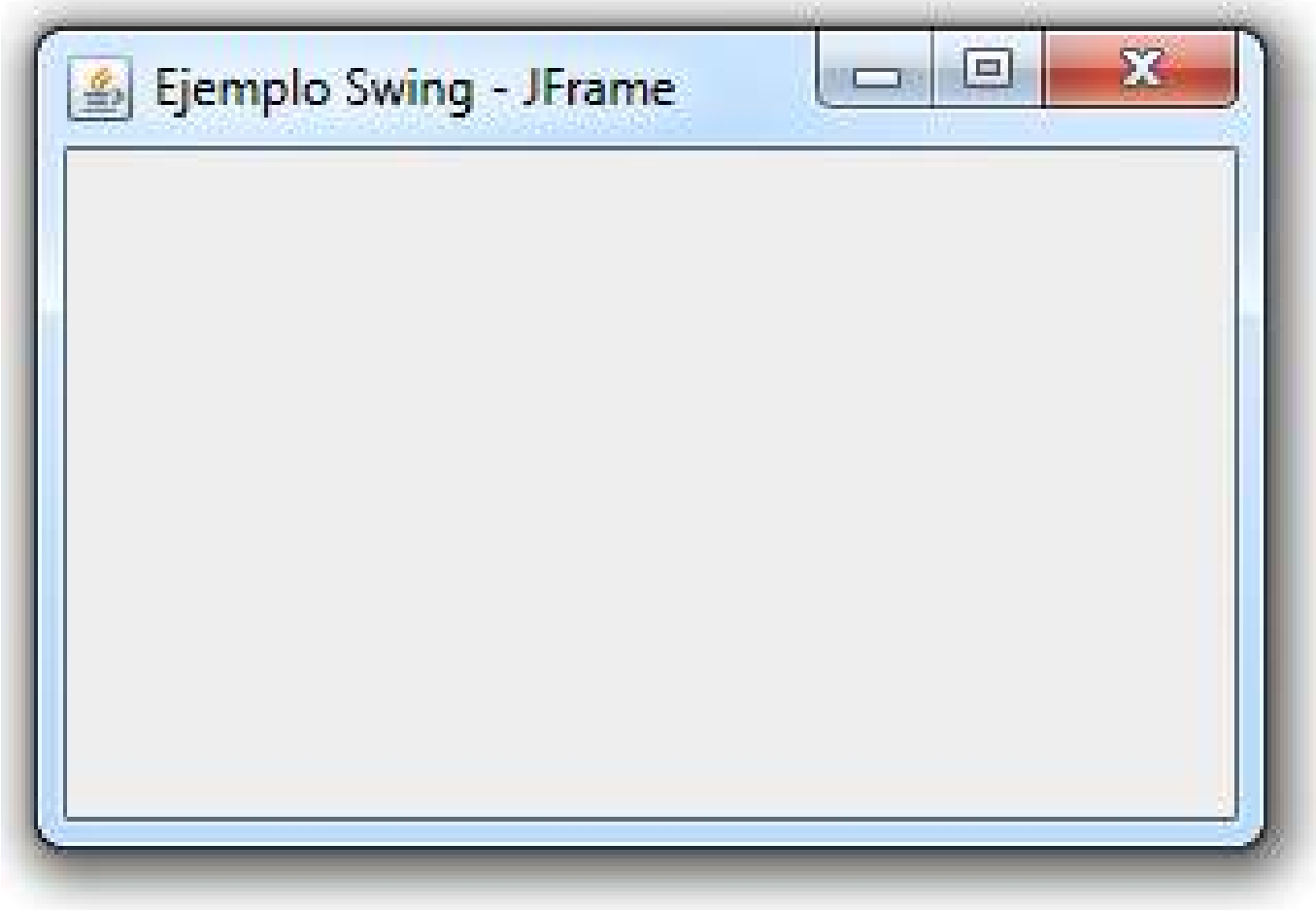
¿Cuáles Son?

Java Swing provee algunos contenedores útiles para diferentes casos.

Así cuando desarrollamos una Ventana podemos decidir:

- De qué manera presentar nuestros elementos.
- Como serán alojados.
- De qué forma serán presentados al usuario.

JFrame



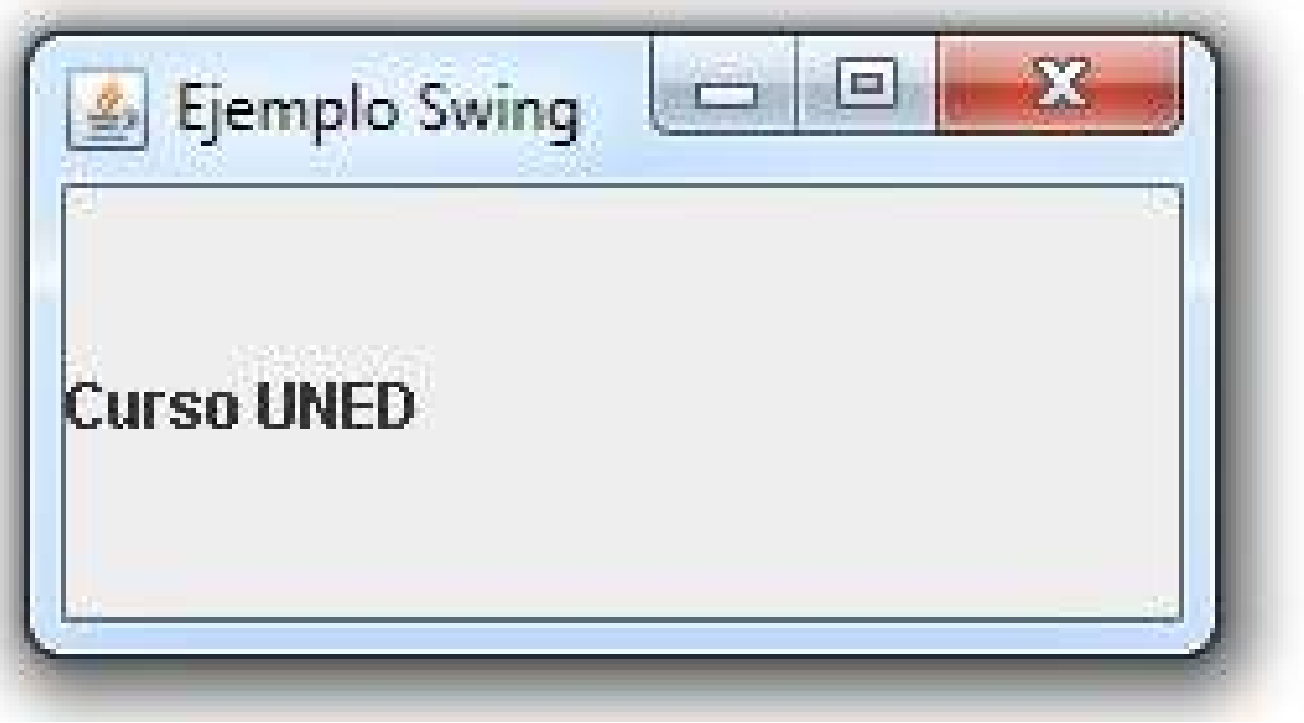
JFrame

Este contenedor es uno de los principales y más usados.

Representa la ventana Principal de nuestra aplicación.

En el podemos alojar otros contenedores.

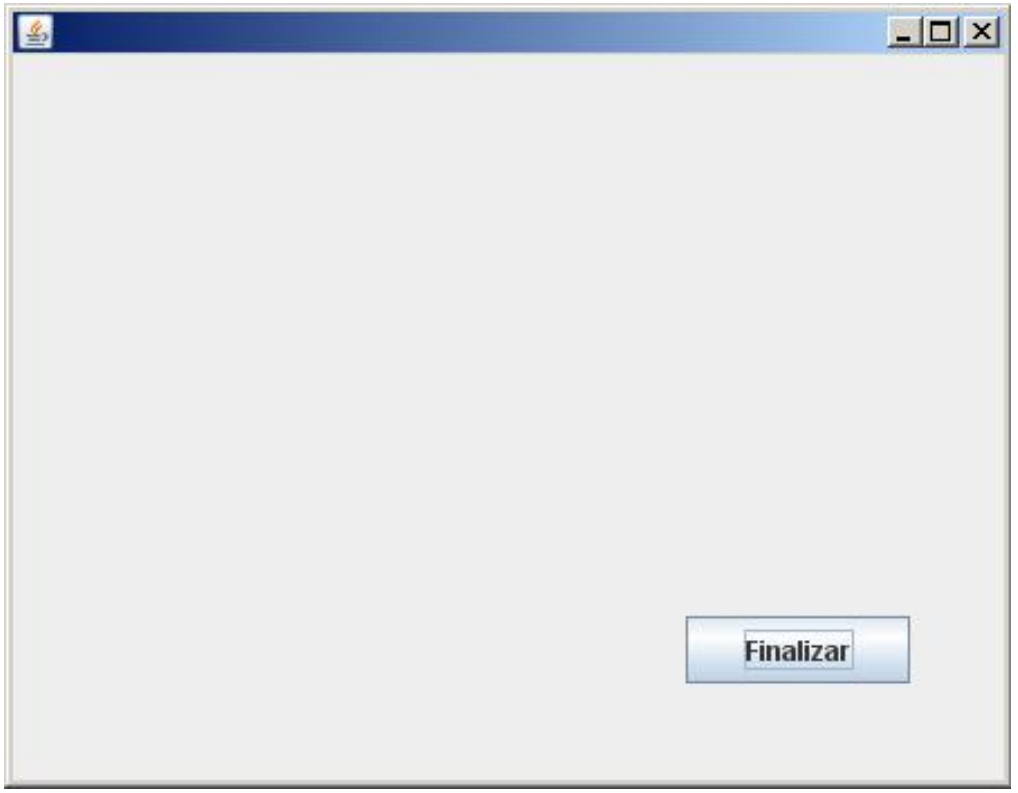
JLabel



JLabel

La clase JLabel nos permite mostrar básicamente un texto.

JButton



JButton

Este control visual muestra un botón.

El proceso para añadir botones a un control JFrame es similar a añadir controles de tipo JLabel.

En este ejemplo veremos la captura de eventos con los controles visuales.

Uno de los eventos más comunes es cuando hacemos clic sobre un botón.

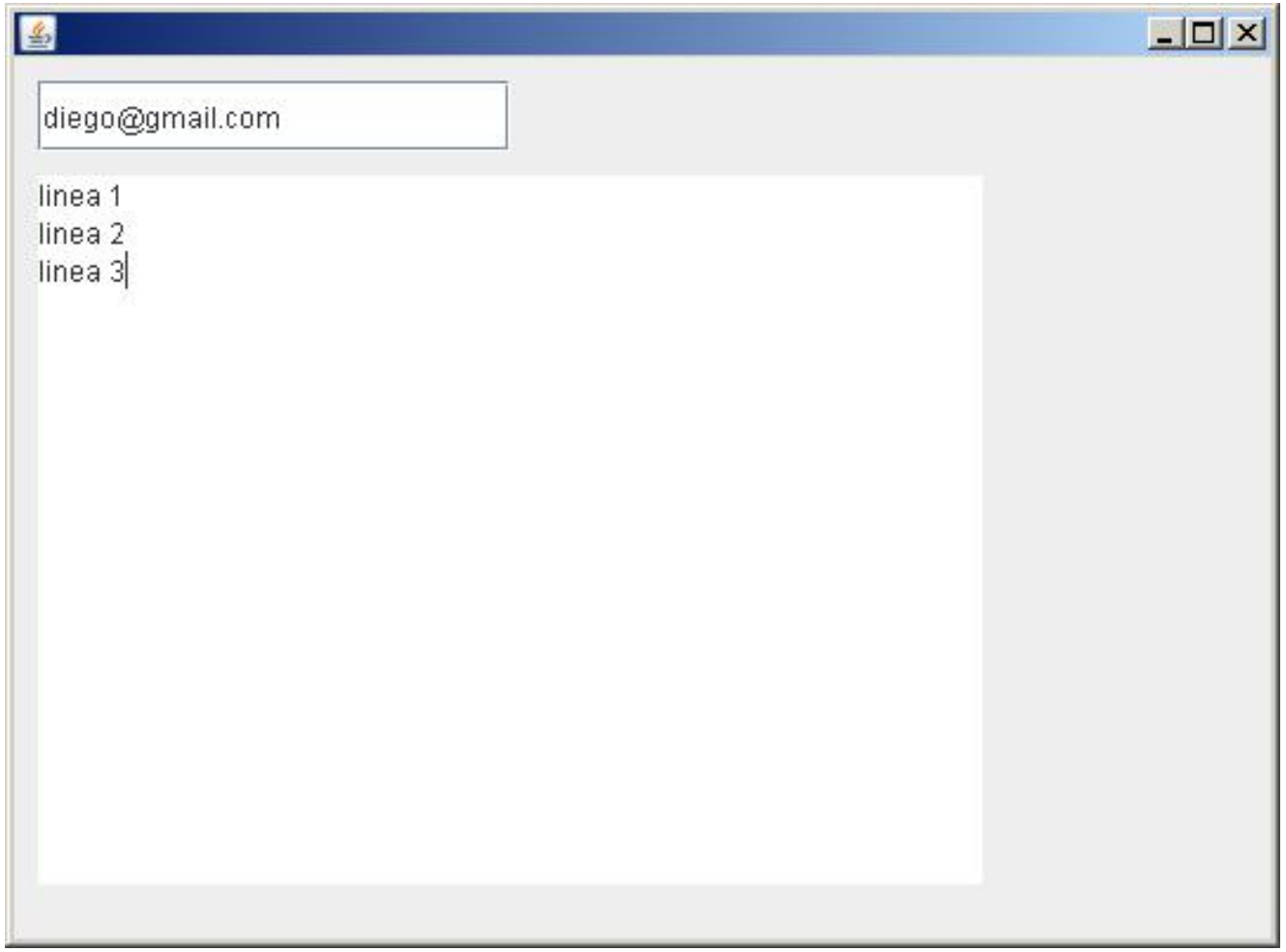
JTextField

JTextField

Así como podríamos decir que el control JLabel remplaza a la salida estándar System.out.print, el control JTextField cumple la función de la clase Scanner para la entrada de datos.

El control JTextField permite al operador del programa ingresar una cadena de caracteres por teclado.

JTextArea



JTextArea

El control de tipo JTextArea permite ingresar múltiples líneas, a diferencia del control de tipo JTextField.

JComboBox

JComboBox

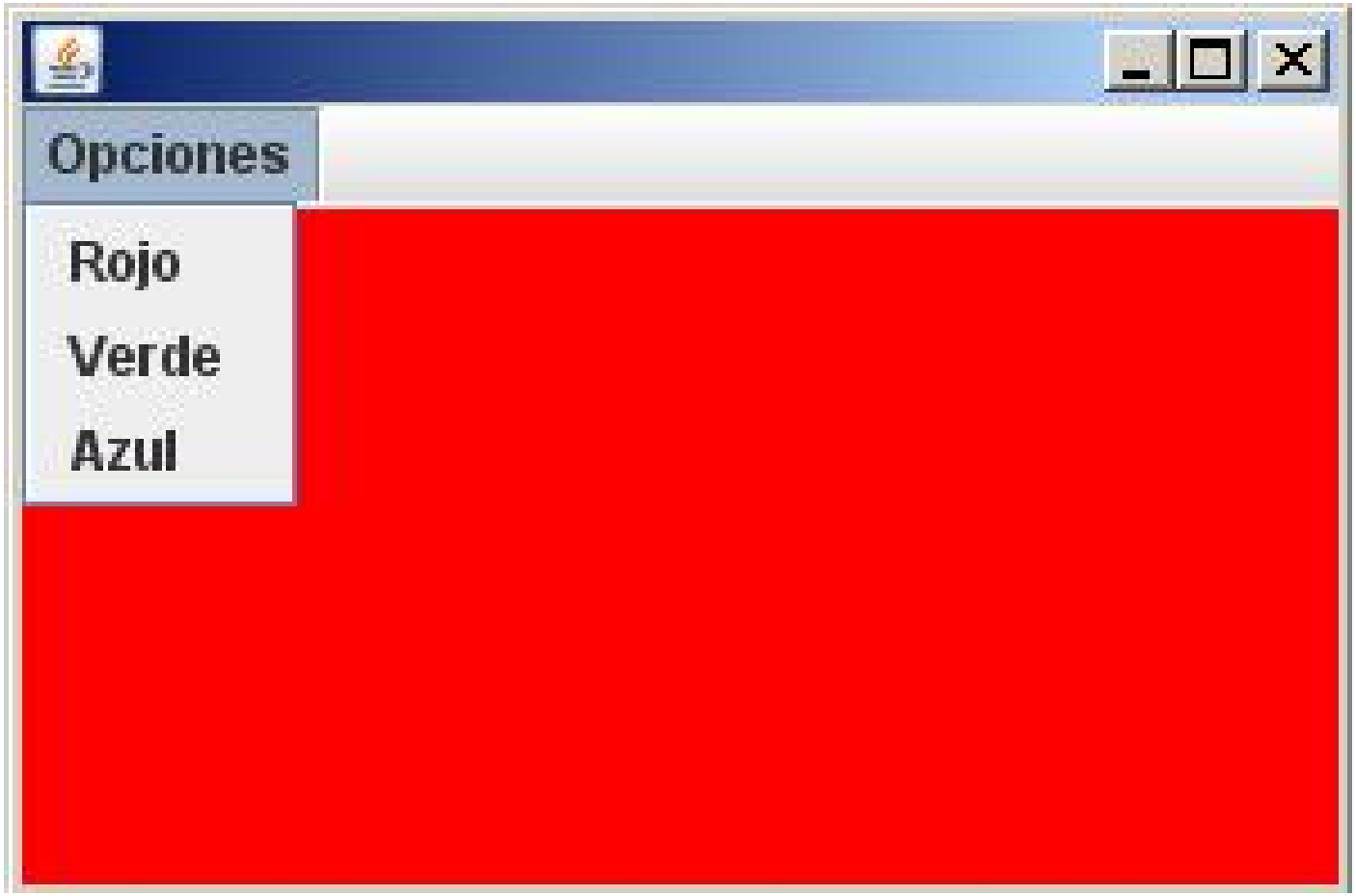
El control JComboBox permite seleccionar un String de una lista.

Para inicializar los String que contendrá el JComboBox debemos llamar al método addItem tantas veces como elementos queremos cargar.

Un evento muy útil con este control es cuando el operador selecciona un Item de la lista.

Para capturar la selección de un item debemos implementar la interface ItemListener que contiene un método llamada itemStateChanged.

JMenuBar, JMenu, JMenuItem



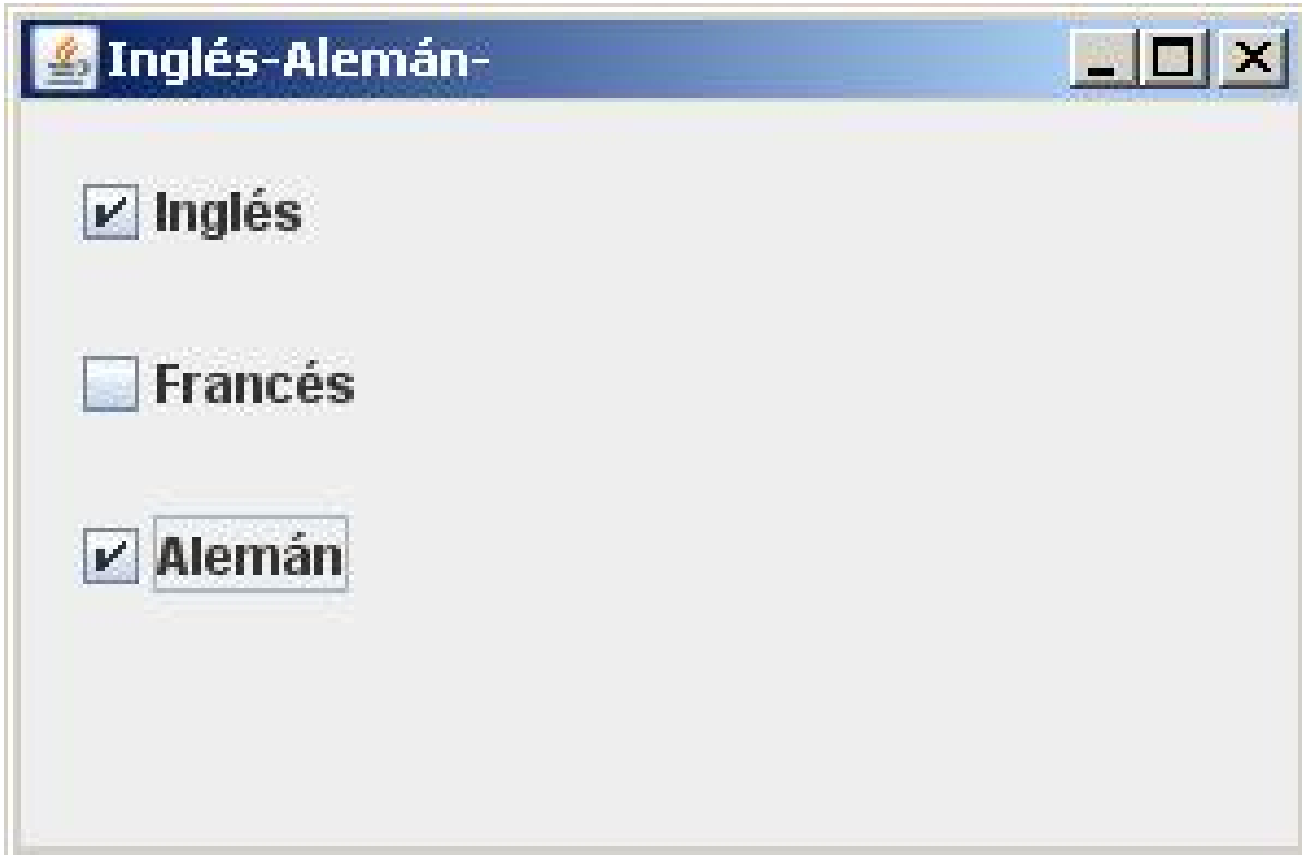
JMenuBar, JMenu, JMenuItem

Lo usamos para crear un menú de opciones y la captura de eventos de los mismos.

Cuando necesitamos implementar un menú horizontal en la parte superior de un JFrame requerimos de un objeto de la clase JMenuBar, uno o más objetos de la clase JMenu y por último objetos de la clase JMenuItem.

Para la captura de eventos debemos implementar la interface ActionListener y asociarlo a los controles de tipo JMenuItem, el mismo se dispara al presionar con el mouse el JMenuItem.

JCheckBox



JCheckBox

El control JCheckBox permite implementar un cuadro de selección.

Básicamente un botón de dos estados.

JRadioButton

JRadioButton

Otro control visual muy común es el JRadioButton que normalmente se muestran un conjunto de JRadioButton y permiten la selección de solo uno de ellos.

Se los debe agrupar para que actúen en conjunto, es decir cuando se selecciona uno automáticamente se deben deseleccionar los otros.

Los gestores de disposición Layout Managers

La forma en que los componentes se distribuyen por la ventana viene dada por un ***distribuidor*** (*Layout*).

Los gestores se adaptan a las dimensiones de la ventana, de manera que si ésta se redimensiona, el distribuidor redistribuye los componentes de acuerdo a unas reglas.

Cualquier ***Container*** tiene operaciones para indicar cuál es su distribuidor, y para añadir componentes: **setLayout()** y **add()** respectivamente.

Los tipos de *Layout* que hay son los siguientes:

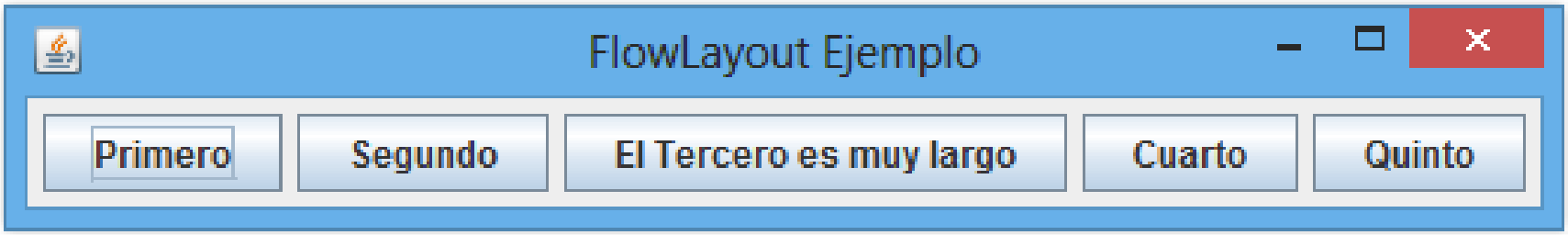
- FlowLayout.
- BorderLayout.
- GridLayout.
- BoxLayout.

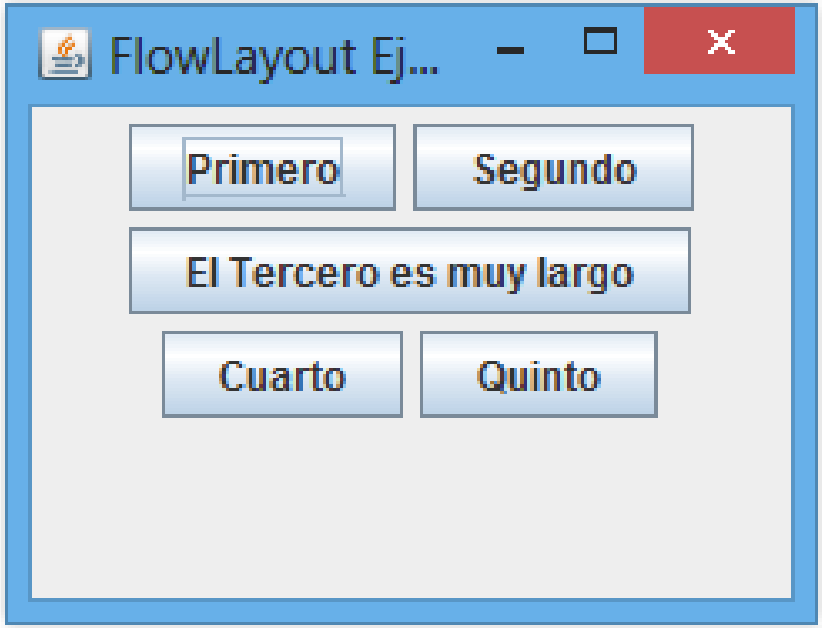
FlowLayout

Es el administrador de disposición más simple que proporciona Java y es el que se proporciona por defecto en los paneles **`JPanel`**.

Este administrador va colocando los componentes de izquierda a derecha, y de arriba abajo

FlowLayout


FlowLayout Ejemplo
Primero
Segundo
El Tercero es muy largo
Cuarto
Quinto

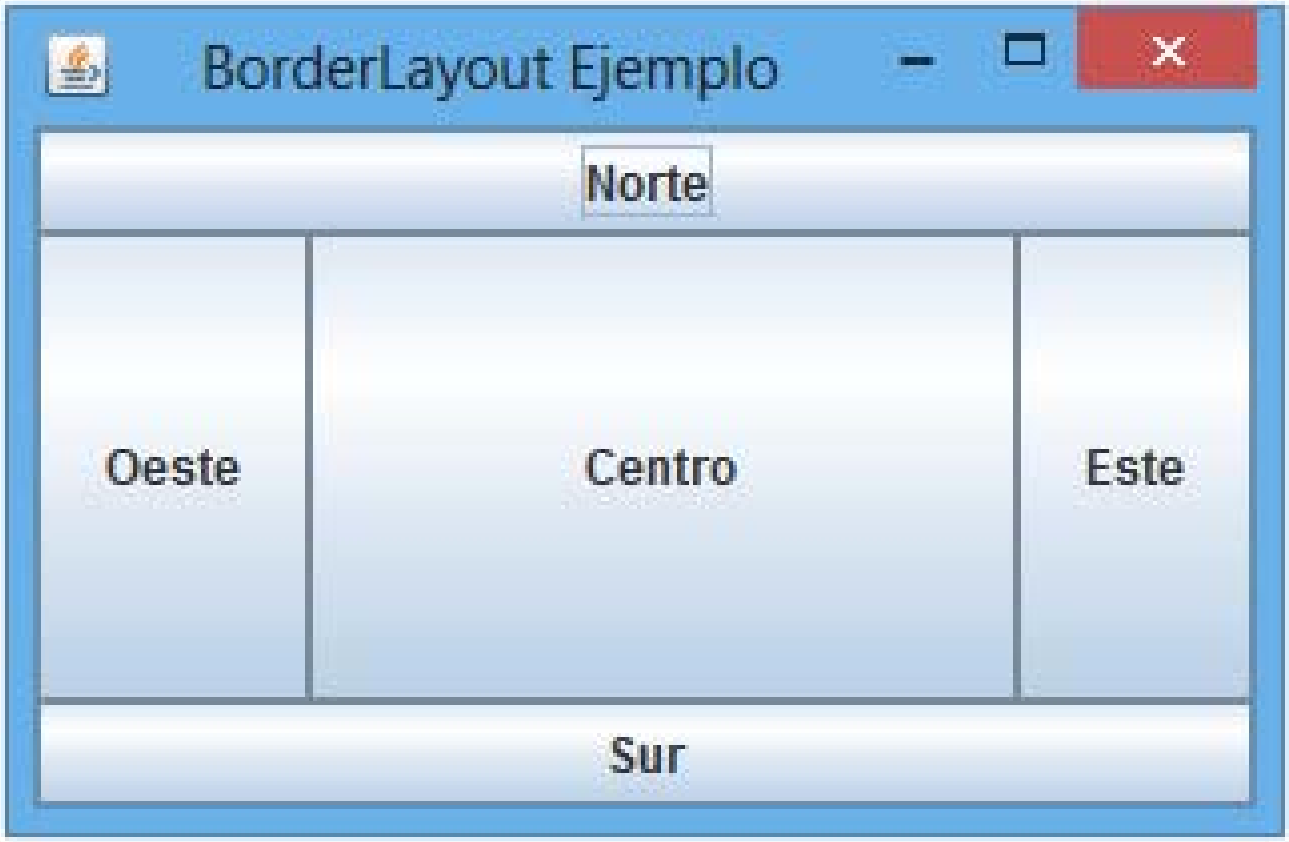

FlowLayout Ej...
Primero
Segundo
El Tercero es muy largo
Cuarto
Quinto

BorderLayout

Este administrador de disposición divide el contenedor en cinco zonas:

- Una central.
- Norte.
- Sur.
- Este.
- Oeste.

BorderLayout



GridLayout

Este administrador de disposición va colocando los componentes de izquierda a derecha, y de arriba abajo, según una matriz de celdas cuyo tamaño se especifica en el constructor.

GridLayout


GridLayout Ejemplo
Primero
Segundo
El Tercero es muy largo
Cuarto
Quinto
Sexto

BoxLayout

El **BoxLayout** permite tanto orientación horizontal como vertical.

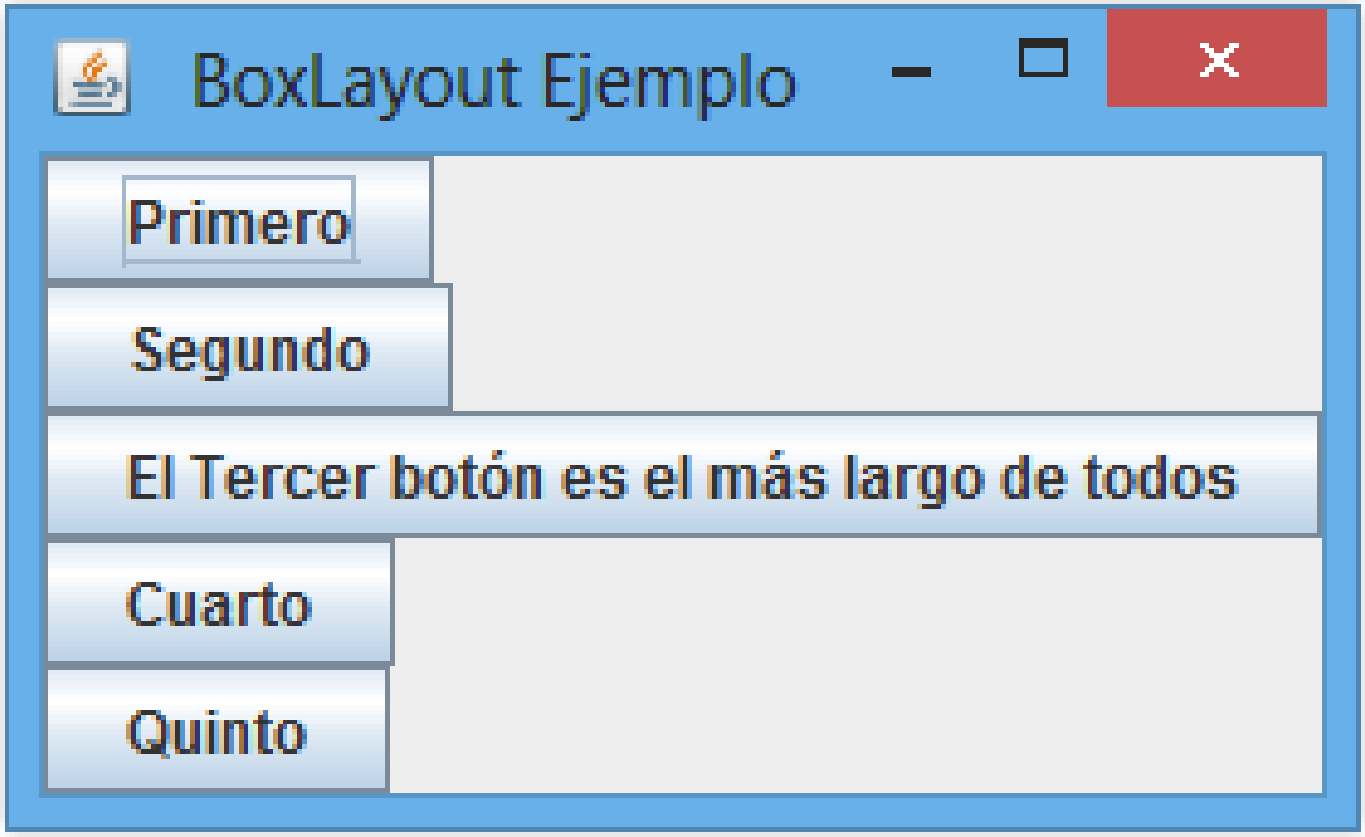
El constructor del **BoxLayout** es más complejo que el del **FlowLayout**.

Podemos distribuir los componentes arriba, abajo, izquierda o derecha.

Debemos pasarle el contenedor al que lo estamos añadiendo, es decir, el parámetro panelIzquierdo.

También debemos pasarle si queremos orientación vertical **BoxLayout.Y_AXIX** u orientación horizontal **BoxLayout.X_AXIS**.

BoxLayout


BoxLayout Ejemplo
Primero
Segundo
El Tercer botón es el más largo de todos
Cuarto
Quinto

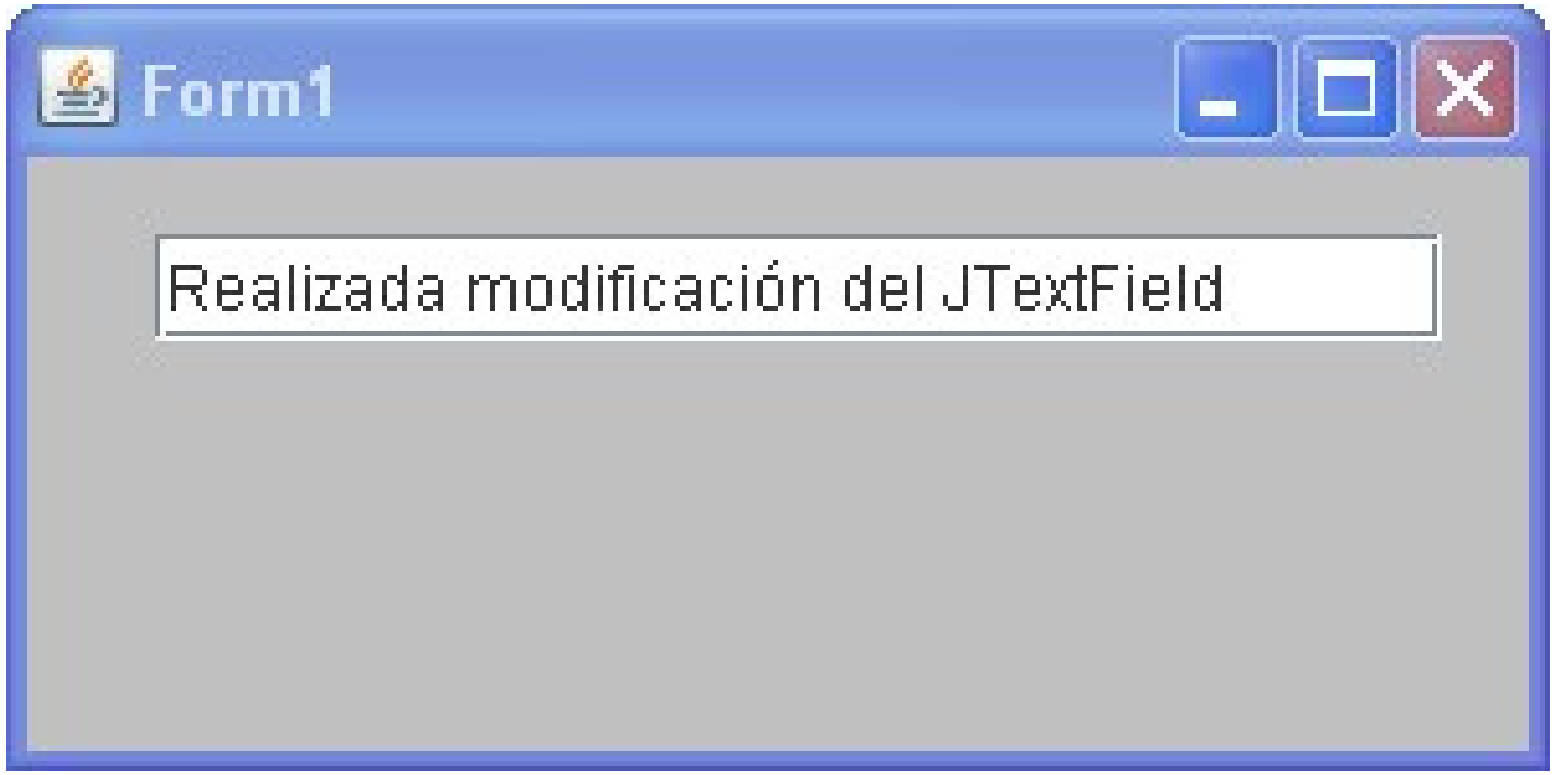
Controles de un Formulario

Una petición de datos se realizará a través de un objeto prefabricado de la clase JTextField.

Mostraremos el siguiente código y la salida respectiva de entrada de datos, con un control visual en Java.

```java
//(1) PAQUETE
import java.awt.*;
import javax.swing.*;

//(2) FORMULARIO
public class Formulario extends JFrame {

    //(3) CONTROLES DEL FORMULARIO
    JPanel jpanel = (JPanel) this.getContentPane();
    JTextField jtextfield = new JTextField();

    //(4) CONSTRUCTOR DEL FORMULARIO
    public Formulario() {

        //(5) PROPIEDADES DEL CONTENEDOR
        jpanel.setLayout(null);
        jpanel.setBackground(Color.lightGray);
```

```
        //(6) PROPIEDADES DE LOS CONTROLES
        jtextfield.setBounds(new Rectangle(25, 15, 250, 21));
        jtextfield.setText("Ejemplo de Interfaz Gráfica");
        jtextfield.setEditable(false);
        jtextfield.setHorizontalAlignment(JTextField.LEFT);

        //(7) ADICION DE LOS CONTROLES AL CONTENEDOR
        jpanel.add(jtextfield, null);

        //(8) PROPIEDADES DEL FORMULARIO
        setSize(300,150);
        setTitle("UNED GUI");
        setVisible(true);
    }

    //(9) METODOS DEL FORMULARIO
    public static void main(String arg[]) {
        new Formulario();
    }
}
```


Form1
Realizada modificación del JTextField

1-Paquete

En este punto se usa la sentencia **`import`**, para declarar los paquetes que emplearemos en nuestro programa.

`JTextField`, **`JPanel`**, **`JFrame`**, son clases que pertenecen al paquete **`javax.swing`**, así como la clase **`Color`** pertenece al paquete **`java.awt`**.

Los paquetes forman parte de java, lo único que hacemos nosotros es cargarlos para poder usarlos en nuestro programa.

2-Formulario

`Formulario extends JFrame.`

La idea de esta sentencia, es hacer que nuestro programa tenga el comportamiento de un formulario (ventana Windows) y para ello debemos heredar (**`extends`**) de **`JFrame`**, sus particularidades.

`JFrame`, es una clase que tiene todas las características propias de una ventana en Windows.

A partir de este punto, nuestro programa deja de ser un programa de consola DOS y pasa a ser un programa visual tipo ventana Windows.

3-Controles del formulario

Aquí, se crean los objetos de los controles visuales que se mostrarán en el formulario.

El primer objeto que vemos es **`jpanel`**, que es un nombre cualquiera y pertenece a la clase **`JPanel`**.

El objeto **`jpanel`**, es lo que se llama un ***contenedor*,** que como su propio nombre indica, va a contener a los demás controles visuales.

Los controles visuales no se ponen directamente en el formulario, sino en el contenedor, colocado éste encima del formulario.

El siguiente objeto es **`jtextfield`**, perteneciente a la clase **`JTextField`.**

Este objeto **`jtextfield`**, contiene el control visual para pedir un dato al usuario y tiene la apariencia de una caja para ingresar texto (**`textbox`**).

4-Constructor del formulario

Es una estructura igual a un método, que se inicia con una apertura de llave “{“ y termina con la clausura de la llave “}”.

Entre dichas llaves se procede a dar a los objetos, que representan a los controles visuales, los atributos.

También añade los controles visuales al contenedor, además de establecer los atributos del formulario.

5-Propiedades del contenedor del formulario

Explicaremos las siguientes instrucciones relacionadas al contenedor:

`jpanel.setLayout(null)`.

Esta instrucción significa, que al pasarle **`null`** como parámetro al método **`setLayout`,** nuestro contenedor, representado a través del objeto **`jpanel`**, no administrará la forma de colocar los controles en el

contenedor, sino que dejará que esa labor la realice el programador a través de coordenadas.

`jpanel.setBackground(Color.lightGray)`.

Al pasarse **`Color.lightGray`**, como parámetro del método **`setBackground`**, le decimos al contenedor, representado en el objeto **`jpanel`**, que tome un color de fondo gris suave.

Si quisiéramos usar otro color de fondo, como el color verde, usaríamos el parámetro **`Color.green`**, y de igual manera para otros colores.

6-Propiedades de los controles.

Aquí estableceremos a través de propiedades, la apariencia de nuestros controles visuales.

En este ejemplo, el control visual será una caja de texto para ingresar datos, representados en el objeto **`jtextfield`**, para lo cual explicaremos las instrucciones siguientes:

`jtextfield.setBounds(new Rectangle(25, 15, 250, 21)).`

Cada uno de los valores del parámetro (25,15,250,21) es igual a (**`x,y,width,height`**), donde **`x,y`**, es la coordenada en la que se ubica el control, dentro del contenedor del formulario.

`width` es el ancho que tendrá el control, o sea, 250 pixeles.

`height` sería la altura, en este caso de 21 píxeles.

`jtextfield.setText("Realizada modificación del JTextField").`

En esta instrucción, queda claro que la cadena “Realizada modificación del **`JTextField`**”, que se le pasa como parámetro al método **`setText`,** aparece dentro de la caja de texto durante la ejecución del programa.

`jtextfield.setEditable(false)`.

Si esta instrucción, está establecida en **`true`**, permite que se pueda escribir sobre el **`JTextField`**.

Si está establecida en **`false`**, impide que el usuario pueda modificar el contenido del **`JTextField`**.

`jtextfield.setHorizontalAlignment(JTextField.LEFT).`

El parámetro **`LEFT`**, permite que el texto en la caja de texto, se alinee a la izquierda, el parámetro **`CENTER`** al centro y el **`RIGHT`** a la derecha.

7-Adición de los controles al contenedor

`jpanel.add(jtextfield, null).`

El método **`add`** pertenece a la clase **`Jpanel.`**

Este método, es usado para añadir un control al contenedor, representado con el objeto **`jpanel`**.

El primer parámetro, el objeto **`jtextfield`**, se añade al contenedor y el segundo parámetro, **`null,`** indica que la posición, dónde colocar el controlador

`jtextfield` dentro del contenedor, será determinado por el programador, a través de coordenadas que fueron explicadas en el punto 5 y 6.

8-Propiedades del formulario

Un formulario tiene una apariencia visual por defecto, por ejemplo el tamaño, el color de fondo, entre otros.

Estas propiedades, las podemos cambiar, a través de una serie de métodos, como los siguientes:

`setSize(300,150).`

Ubica la esquina superior izquierda del formulario en la pantalla, en la coordenada

`(300,150) = (x,y) = (columna,fila)`

`setTitle("Form1").`

La cadena “Form1”, como parámetro del método **`setTitle`**, significa que se establecerá como título del formulario la cadena “**`Form1`**”.

`setVisible(true).`

Este parámetro **`true`**, del método **`setVisible`**, determina que el formulario sea visible en la pantalla, ya que si ponemos **`false`**, el formulario está en la pantalla de forma invisible.

9-Métodos del formulario

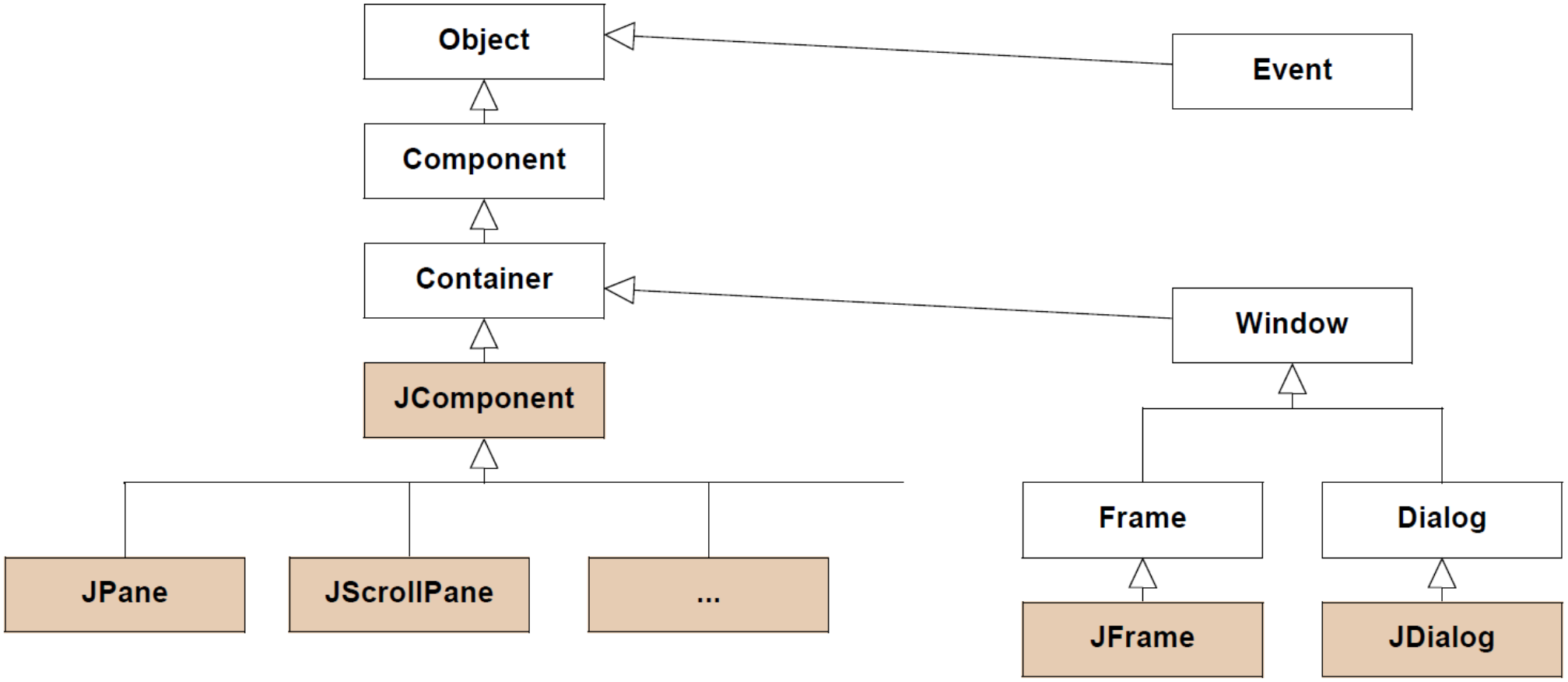
En este punto se definen los métodos que se necesitan para realizar las tareas que diseñan el formulario.

También es el lugar del método **`main`**, que hace que nuestra clase **`Formulario`** se pueda ejecutar, ya que si dejamos de poner el método **`main`**, no podrá ejecutarse, porque dentro de este método, es dónde creamos un objeto del formulario, a través de la instrucción **`new Programa ().`**

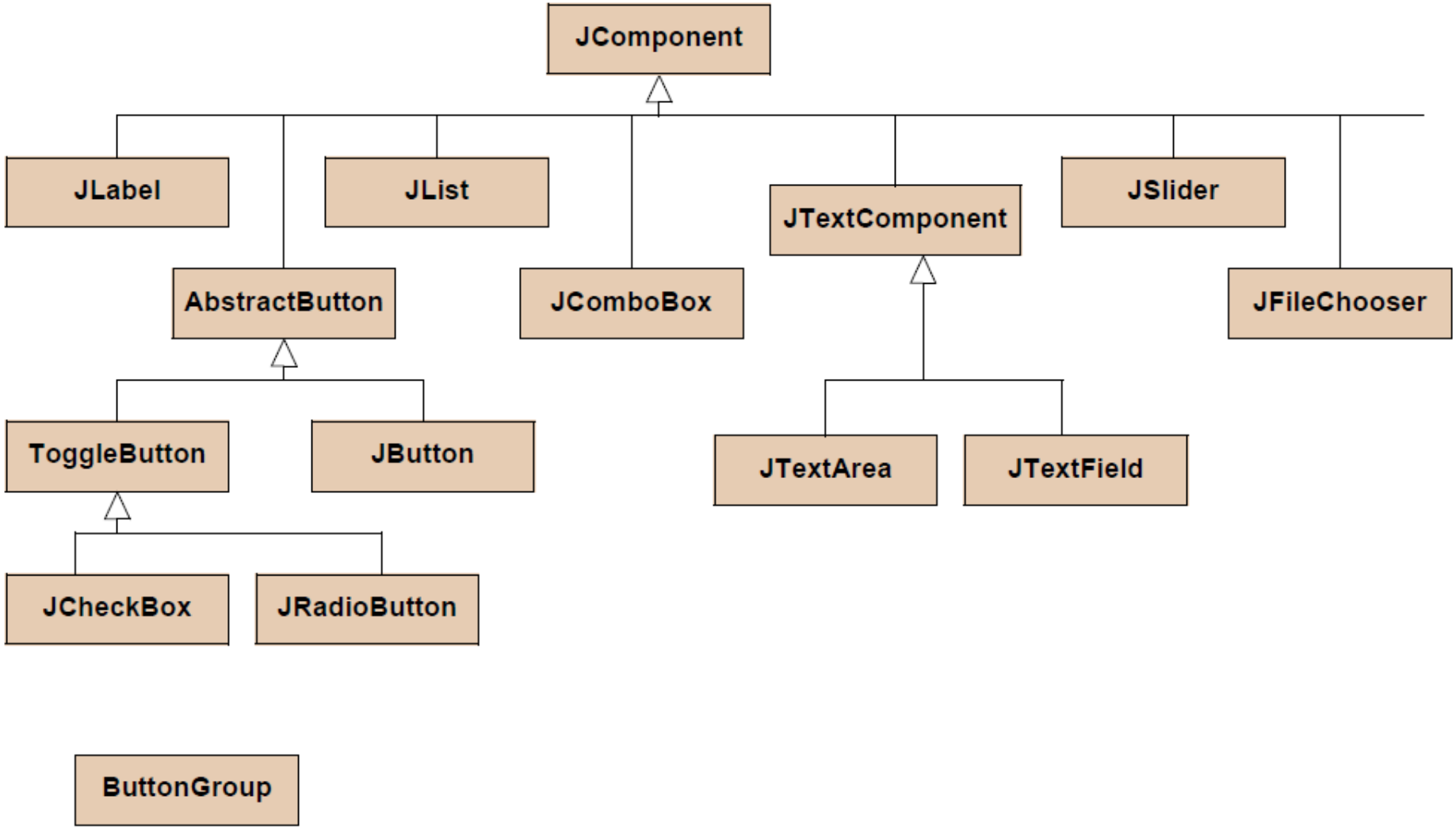
Por tanto debe quedar claro que el formulario se crea siempre y cuando creemos un objeto de la clase **`Formulario`**, lo cual declaramos como **`new Formulario();`**

Jerarquía de Clases Swing y AWT

Ventanas, contenedores y eventos



Componentes Swing



Programación Orientada a Eventos

Eventos en Java

En la programación dirigida por eventos el programa "*espera*" la llegada de un evento.

Es una orden del usuario.

Cuando el evento se produce realiza la acción correspondiente y vuelve a esperar.

El usuario se comunica con una aplicación de interfaz gráfica mediante la interacción con la interfaz.

Por ejemplo, se pulsa el ratón.

Como resultado de estas acciones se generan ***eventos***.

El tratamiento de estos eventos mediante la ejecución de un método en el que se responde a la acción del usuario es la forma de obtener la interactividad de la aplicación.

Esto se conoce como programación basada en eventos, o dirigida por eventos.

Para que los componentes funcionen correctamente también vamos a necesitar objetos que gestionen los eventos. (ActionListener)

Pulsación de tecla, movimiento de ratón, pulsación del botón del ratón, ...

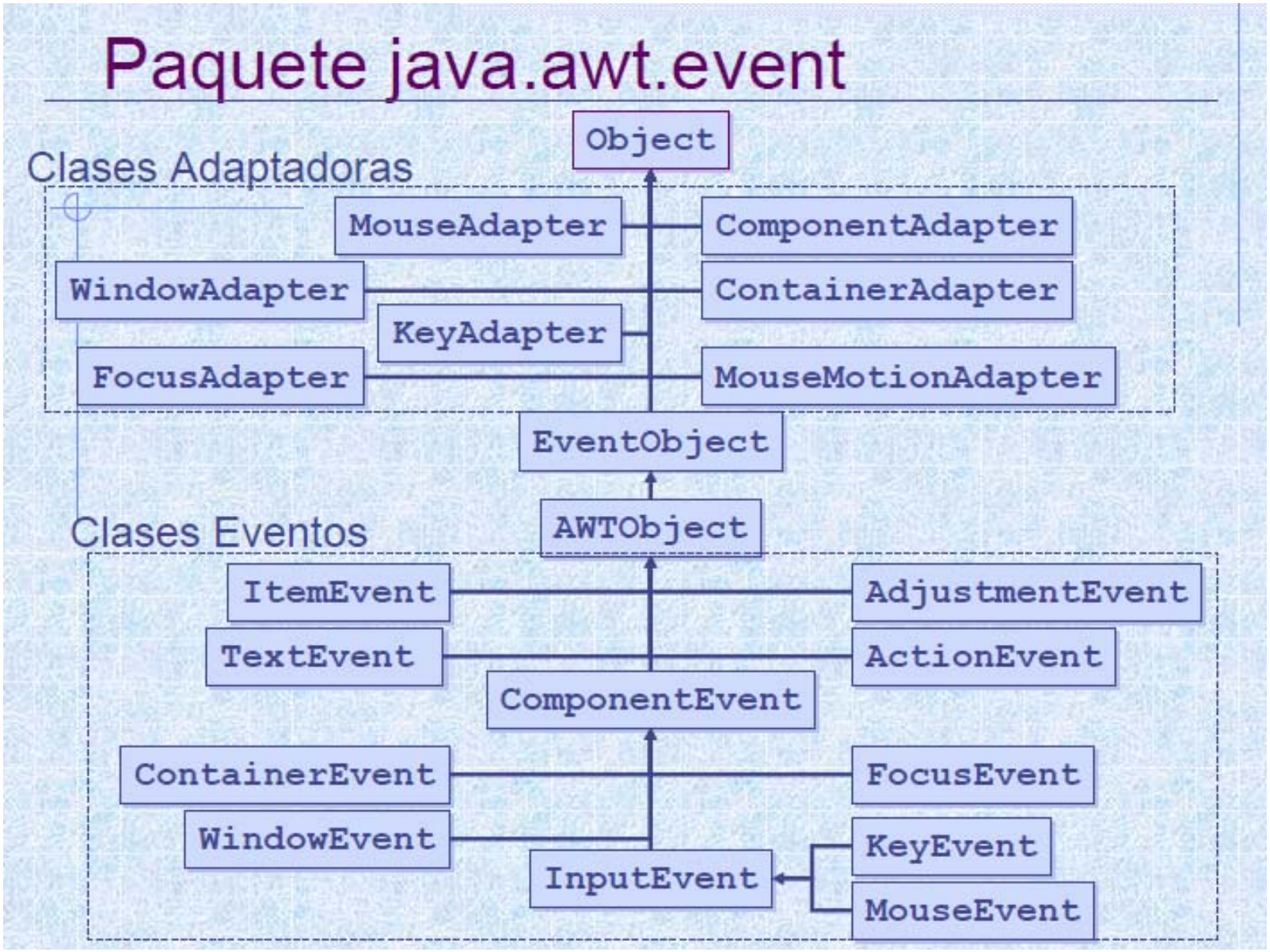
El paquete java.awt.event

El paquete **`java.awt.event`** proporciona clases base para generar eventos.

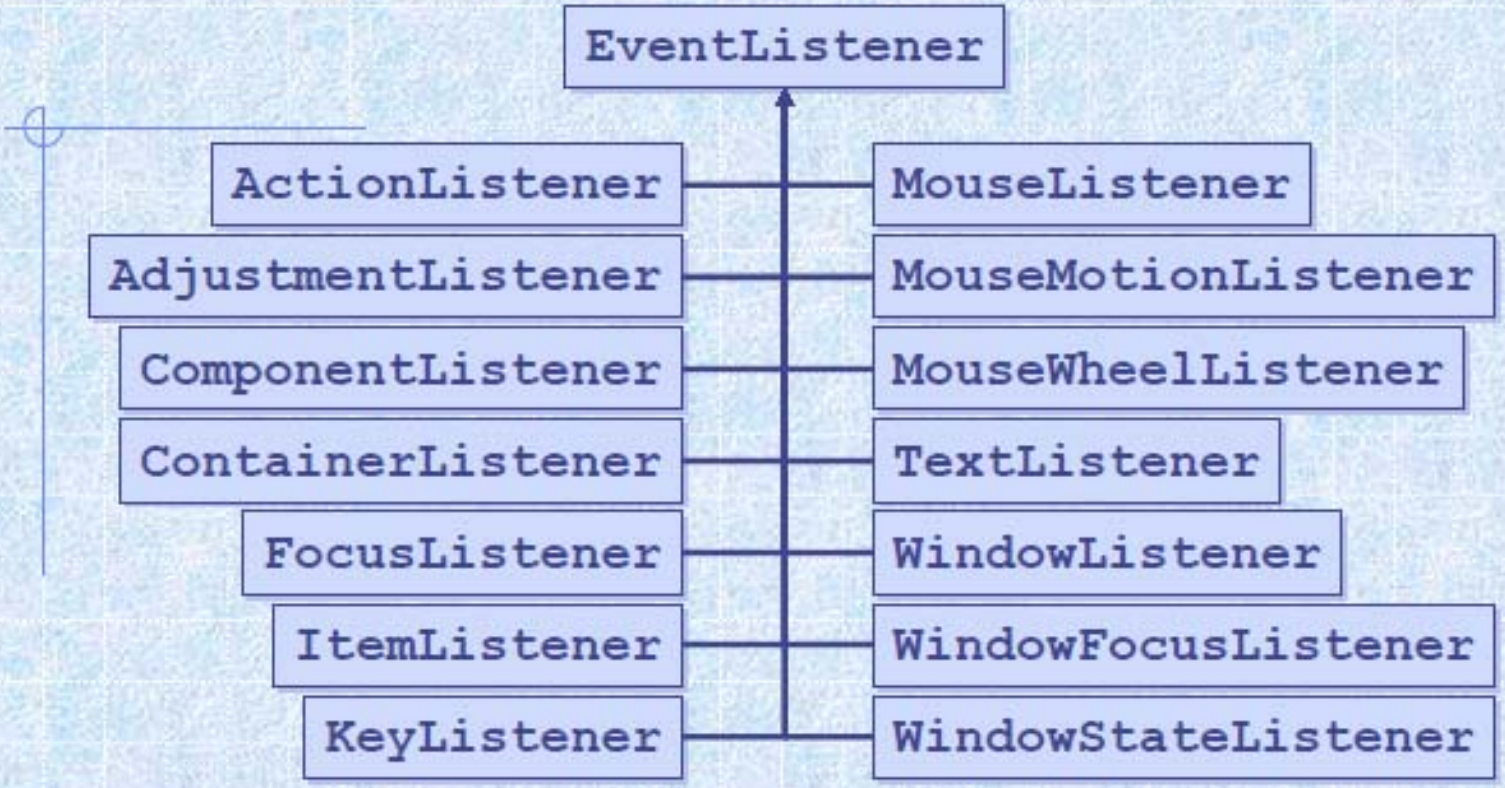
Contiene una serie de interfaces para implementar objetos oyentes.

Adicionalmente proporciona un grupo de clases adaptadoras para facilitar la codificación.

Paquete java.awt.event



Paquete java.awt.event



Interfaces que deben implementar los oyentes

Captura y tratamiento de eventos

El tratamiento de eventos se realiza mediante el ***modelo de delegación de eventos***.

El tratamiento de un evento que ocurre en un objeto no se realiza en ese mismo objeto, ***objeto fuente***, sino que se delega en otro objeto diferente, se conoce como ***oyente***.

Este modelo se basa en la posibilidad de envío de eventos desde el objeto fuente donde se producen a los objetos oyentes que los gestionan.

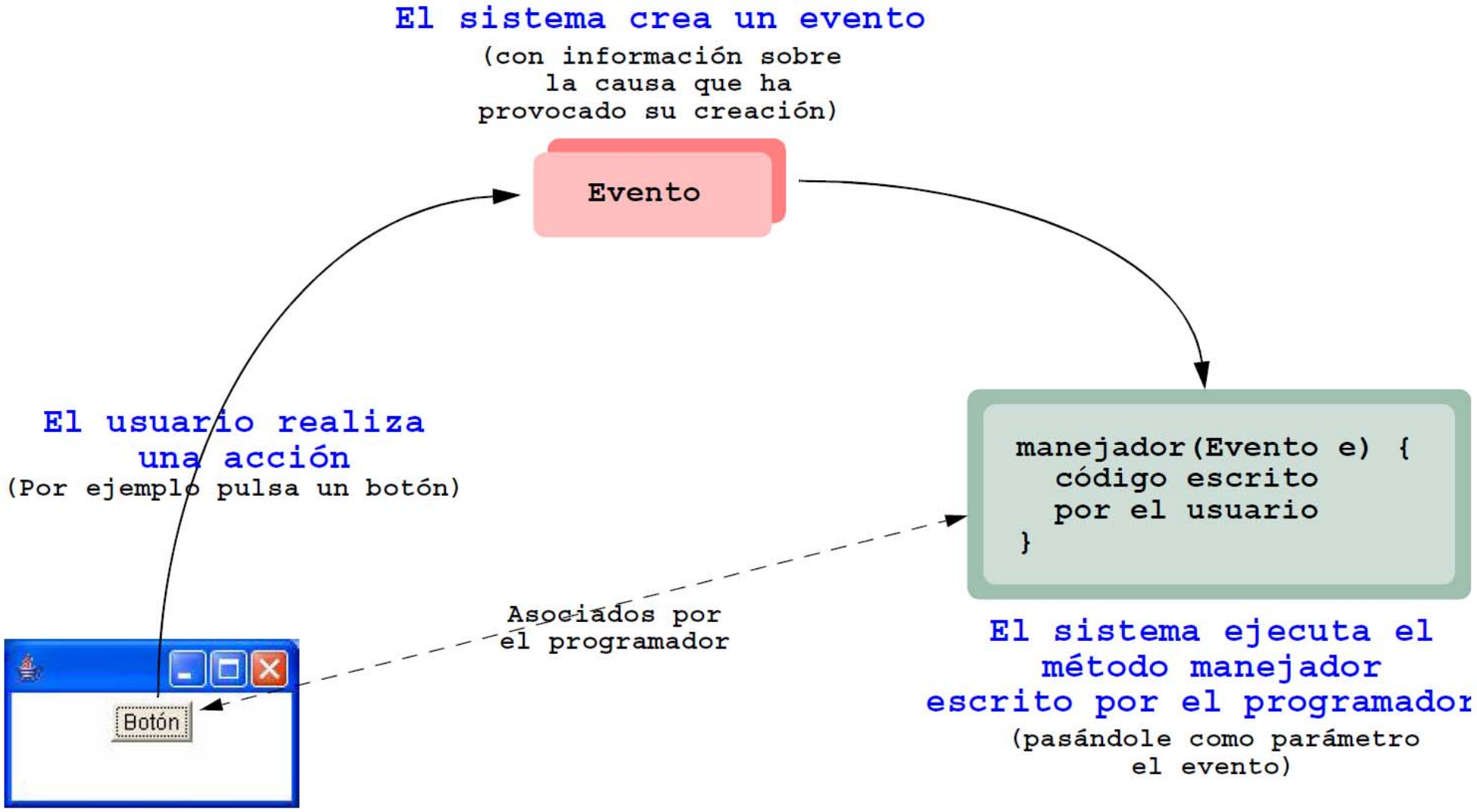
La forma de hacerlo es la siguiente:

El objeto fuente registra qué objetos oyentes están interesados en recibir algún evento específico para comunicárselo una vez que éste se produzca.

Cuando ocurre el evento, el objeto fuente se lo comunica a todos los oyentes registrados.

La comunicación entre la fuente y el oyente se realiza mediante la invocación de un método del oyente proporcionando como argumento el evento generado.

Generación de Eventos



Action Event

Ejemplos de eventos y sus escuchadores	
Acción que lanza un evento	Tipo de escuchador
El usuario hace un click, presiona Return en un área de texto o selecciona un menú	`ActionListener`
El usuario escoge un frame (ventana principal)	`WindowListener`
El usuario hace un click sobre una componente	`MouseListener`
El usuario pasa el mouse sobre una componente	`MouseMotionListener`
Una componente se hace visible	`ComponentListener`
Una componente adquiere el foco del teclado	`FocusListener`
Cambia la selección en una lista o tabla	`ListSelectionListener`