Introducción a interfaz gráfica de usuario

La parte visual de un programa es muy importante, sobre todo, a la hora de interactuar con el usuario. Java presenta múltiples herramientas para crear interfaces de usuario de forma rápida y sencilla. En este módulo, nos centraremos en dos bibliotecas fundamentales que son útiles para crear interfaces gráficas de usuario: AWT y Swing.

Introducción a interfaz gráfica de usuario

Introducción a la programación de aplicaciones gráficas

La interfaz gráfica es uno de los elementos fundamentales para comunicarnos con el usuario final de nuestras aplicaciones. Resulta fundamental conocer bien el problema a solucionar y contar con todas las herramientas disponibles para brindar la mejor solución en el menor tiempo a nuestros clientes. También, es sumamente importante poder brindarles una interfaz clara, sencilla de utilizar, intuitiva y, además, que sea estéticamente agradable para conseguir un producto profesional y bien logrado.

El desarrollo de la parte gráfica no es trivial y para poder enfrentarnos a este desafío contamos con diferentes bibliotecas. Además, según el IDE que utilicemos para programar en Java, contaremos con herramientas más sencillas para desarrollar las diferentes pantallas involucradas en nuestra aplicación.

Introducción a interfaz gráfica de usuario con AWT

AWT (Abstract Windows Toolkit) es la primera biblioteca que incorporó Java para construir interfaces gráficas de usuario. Este paquete ha estado presente desde la primera versión (la 1.0), aunque con la 1.1 sufrió un

cambio notable. En la versión 1.2 se incorporó también a Java una librería adicional, Swing, que podríamos considerar que es una versión de AWT con vitaminas, ya que incorpora mejoras sustanciales con respecto a esta primera librería.

Para poder comprender mejor cómo funciona Swing, que será la librería de nuestro interés, comenzaremos primero analizando AWT. Así, luego comprenderemos cuáles fueron las mejoras que se introdujeron.

La biblioteca AWT

Esta biblioteca es la primera disponible para el desarrollo de interfaces gráficas. Contiene una multitud de clases e interfaces que permiten la definición y la gestión de interfaces gráficas. En realidad, esta biblioteca utiliza las funcionalidades gráficas del sistema operativo. Por lo tanto, no es el código presente en esta biblioteca el que asegura el resultado gráfico de los diferentes componentes. Este código hace de intermediario con el sistema operativo. Su uso ahorra bastantes recursos, pero presenta varios inconvenientes.

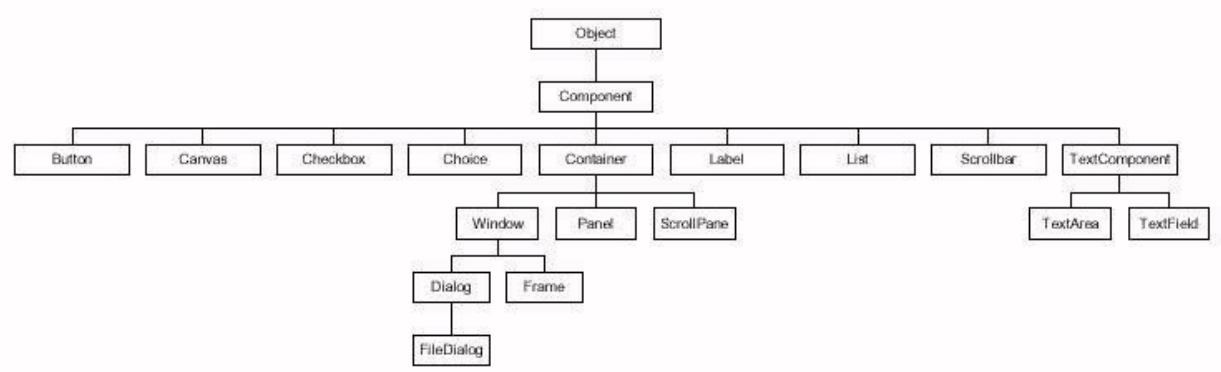
- Al estar relacionado el aspecto visual de cada componente con la representación que el sistema operativo hace de él, puede resultar delicado desarrollar una aplicación que tenga una apariencia coherente en todos los sistemas. El tamaño y la posición de los diferentes componentes son los dos elementos que se ven principalmente afectados por este problema.

- Para que esta biblioteca sea compatible con todos los sistemas operativos, los componentes que contiene están limitados a los más corrientes (botones, zonas de texto, listas, etc.).

Los elementos que proporciona esta librería para diseñar aplicaciones visuales son un grupo de controles que derivan de la clase Component y forman parte del paquete java.awt.

En la siguiente figura, podemos ver los más comunes:

Figura 1: Controles



Fuente: elaboración propia

Tal vez nos llamen la atención algunos controles que están más desarrollados que otros, por ejemplo, el Container. En este caso, los *containers* son aquellos elementos que pueden contener al resto de los controles para hacerlos accesibles a los usuarios. Cuando desarrollemos una

interfaz gráfica, iremos armando diferentes sectores de la pantalla o incluso diferentes pantallas, mediante la visualización de diferentes *containers* con sus correspondientes componentes internos.

Supongamos que queremos desarrollar una interfaz gráfica para el logueo de un usuario en el sistema de una cierta empresa. Para ello, tendremos que diseñar una pantalla inicial en la cual se pida la identificación del usuario y, en el caso de que la validación sea correcta, se mostrará la siguiente pantalla; pero en caso de validación incorrecta, se deberá tomar otra medida visual.

Veamos cómo utilizar un *container.*

Para agregar un componente dentro de un contenedor utilizamos el método add(...) del contenedor. Por ejemplo, si queremos añadir un botón a un panel:

```
Button boton = new Button("Pulsame");

Panel = new Panel();

...

panel.add(boton);
```

Un detalle no menor es que en Java, por lo menos, si estamos programando mediante un editor de texto o en los IDE más tradicionales, no tenemos plantillas predeterminadas, ni contamos con una aplicación gráfica para

arrastrar componentes y ubicarlos para armar la interfaz que queremos para el usuario. En Java, la interfaz gráfica se arma directamente mediante código, y es allí en donde tenemos que configurar todas las opciones de aspecto, ubicación, etc. que queremos para nuestra aplicación.

Veamos una descripción de los controles más utilizados de la librería AWT:

Controles librería AWT

Component

La clase padre Component no se puede utilizar directamente. Es una clase abstracta, que proporciona algunos métodos útiles para sus subclases.

Botones

Para emplear la clase **Button**, en el constructor, simplemente indicamos el texto que queremos que tenga:
Button boton = new Button("Pulsame");

Etiquetas

El uso de **Label** es muy similar al botón: se crea el objeto con el texto que deseemos:
Label etiq = new Label("Etiqueta");

Áreas de dibujo

La clase **Canvas** se emplea para heredar de ella y crear componentes personalizados.
Accediendo al objeto Graphics de los elementos, podremos darle la apariencia que queramos: dibujar líneas, pegar imágenes, etc.

```
Panel p = new Panel();
p.getGraphics().drawLine(0, 0, 100, 100);
p.getGraphics().drawImage(...);
```

Casillas de verificación

Checkbox se emplea para marcar o desmarcar opciones. Podremos tener controles aislados o grupos de ***checkboxes*** en un objeto ***CheckboxGroup***, de forma que solo una de las casillas del grupo pueda marcarse cada vez.

```
Checkbox cb = new Checkbox
        ("Mostrar subdirectorios",
        false);
System.out.println ("Esta marcada: " +
```

cb.getState());

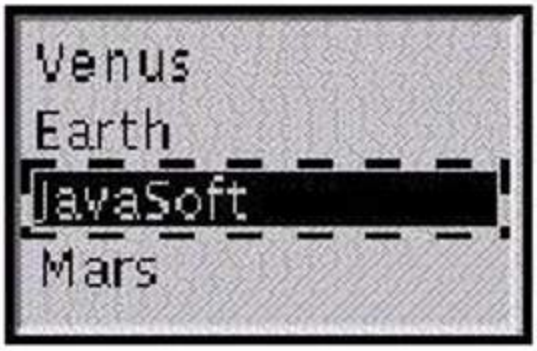
Listas

Para utilizar una **lista desplegable** (objeto Choice), se crea el objeto y se añaden, con el método **addItem(...)**, los elementos que queramos a la lista.

```
Choice ch = new Choice();
ch.addItem("Opcion 1");
ch.addItem("Opcion 2");
...
int i = ch.getSelectedIndex();
```

Para utilizar **listas fijas** (objeto **List**), en el constructor, indicamos cuántos elementos son visibles. También, podemos indicar si se permite seleccionar varios elementos a la vez. AWT dispone de muchos de los métodos que tiene Choice para añadir y consultar elementos.

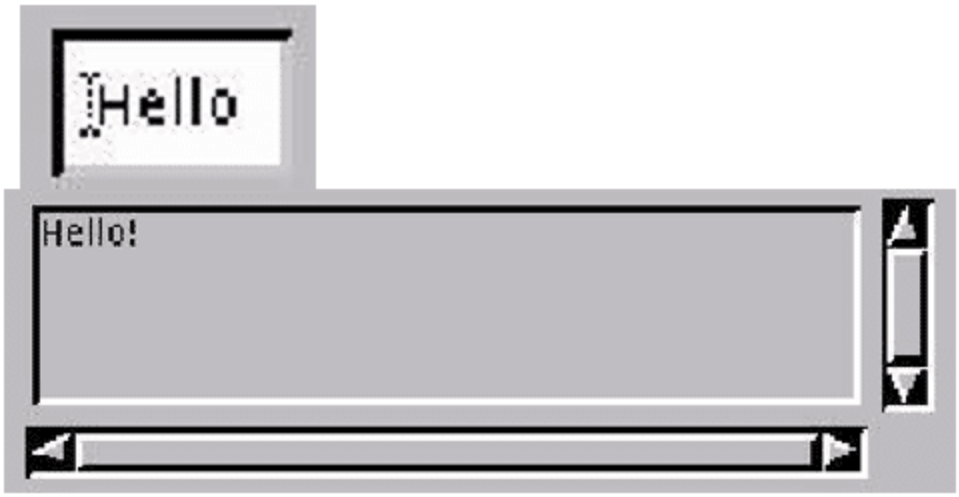
```
List lst = new List(3, true);
lst.addItem("Opcion 1");
lst.addItem("Opcion 2");
```



Cuadros de texto

Al trabajar con **TextField o TextArea,** se indica opcionalmente en el constructor el número de columnas (y filas en el caso de TextArea) que se quieren en el cuadro de texto.

```
TextField tf = new TextField(30);
TextArea ta = new TextArea(5, 40);
...
tf.setText("Hola");
ta.appendText("Texto 2");
String texto = ta.getText();
```



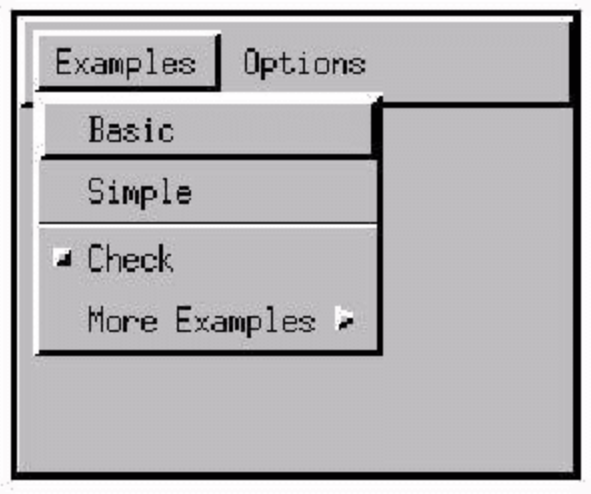
Menús

Para utilizar menús, se emplea la clase **MenuBar** (para definir la barra de menú), **Menu** (para definir cada menú), y **MenuItem** (para cada opción en un menú). Un menú podrá contener, a su vez, submenús (objetos de tipo Menu). También, está la clase **CheckboxMenuItem** para definir opciones de menú que son casillas que se marcan o desmarcan.

```
MenuBar mb = new MenuBar();
Menu m1 = new Menu "Menu 1");
Menu m11 = new Menu ("Menu 1.1");
Menu m2 = new Menu ("Menu 2");
MenuItem mi1 = new MenuItem ("Item 1.1");
MenuItem mi11=new MenuItem ("Item 1.1.1");
CheckboxMenuItem mi2 =
  new CheckboxMenuItem("Item 2.1");
mb.add(m1);
mb.add(m2);
m1.add(mi1);
m1.add(m11);
m11.add(mi11);
```

m2.add(mi2);

Mediante el método setMenuBar(...) de frame, podremos añadir un menú a una ventana.
Frame f = new Frame();
f.setMenuBar(mb);



Fuente: elaboración propia

Gestores de disposición

Para colocar los controles Java en los contenedores, se hace uso de un determinado **gestor de disposición**. Dicho gestor indica cómo se colocarán los controles en el contenedor, siguiendo una determinada distribución. Para establecer qué gestor queremos, se emplea el método setLayout(...) del contenedor. Por ejemplo:

Panel panel = new Panel();

panel.setLayout(new BorderLayout());

Veremos ahora los gestores más importantes.

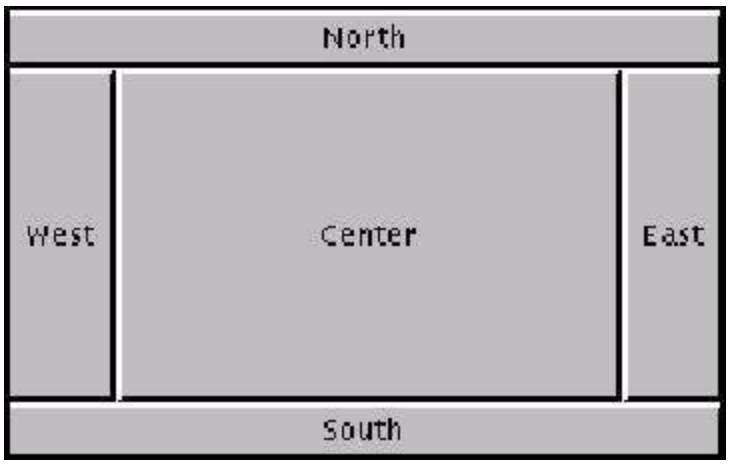
Gestores

BorderLayout (gestor por defecto para contenedores tipo Window)

Divide el área del contenedor en cinco zonas: norte (NORTH), sur (SOUTH), este (EAST), oeste (WEST) y centro (CENTER). De esta forma, al colocar los componentes, deberemos indicar en el método add(...) en qué zona colocarlo.

```
panel.setLayout(new BorderLayout());
Button btn = new Button("Pulsame");
panel.add(btn, BorderLayout.SOUTH);
```

Cuando incluimos un componente en una zona, se colocará sobre el que existiera anteriormente en dicha zona (lo tapa).



FlowLayout (gestor por defecto para contenedores de tipo Panel)

Con este gestor, se colocan los componentes en fila, uno detrás de otro, con el tamaño preferido (preferredSize) que se les haya dado. Si no caben en una fila, se utilizan varias.

```
panel.setLayout(new FlowLayout());
panel.add(new Button("Pulsame"));
```

GridLayout

Este gestor sitúa los componentes en forma de tabla, dividiendo el espacio del contenedor en celdas del mismo tamaño, de forma que el componente ocupa todo el tamaño de la celda.
Se indica en el constructor el número de filas y de columnas. Luego, colocamos el componente por orden (rellenando filas de izquierda a derecha).

```
panel.setLayout(new GridLayout(2,2));
panel.add(new Button("Pulsame"));
panel.add(new Label("Etiqueta"));
```

Sin gestor

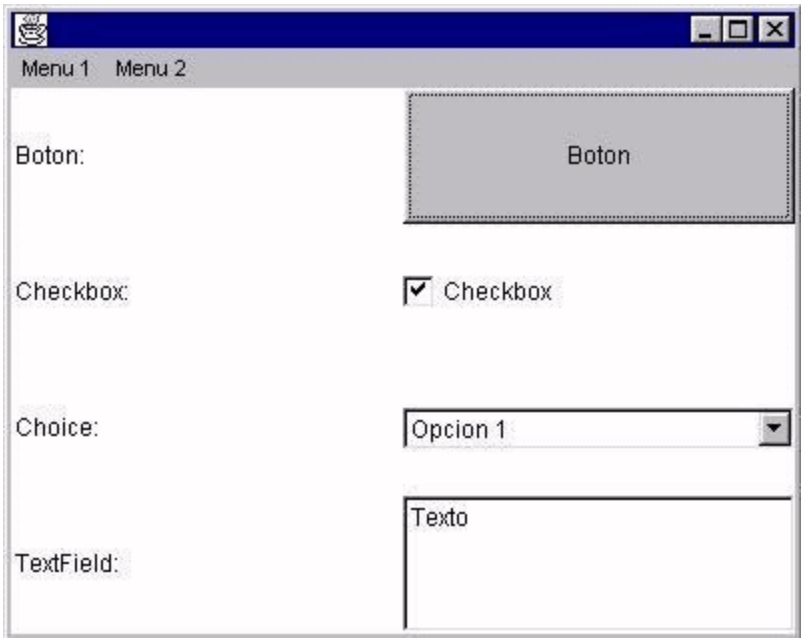
Si especificamos un gestor **null,** podremos colocar a mano los componentes en el contenedor, con métodos como **setBounds(...) , o setLocation(...).**

```
panel.setLayout(null);
Button btn = new Button ("Pulsame");
btn.setBounds(0, 0, 100, 30);
panel.add(btn);
```

Fuente: elaboración propia

Veremos el aspecto de algunos componentes de AWT y el uso de gestores de disposición en el siguiente ejemplo:

Figura 2: Ejemplo



Fuente: elaboración propia

```
import java.awt.*;

import java.awt.event.*;

/**

 * Este es un ejemplo en el que se muestran algunos de los controles

 * de AWT

 */

public class EjemploAWT extends Frame
```

```
{

        /**

          * Constructor

          */

        public EjemploAWT()

        {

                setSize(400, 300);

                setLayout(new GridLayout(4, 2));



                // Menus



                MenuBar mb = new MenuBar();

                Menu m1 = new Menu ("Menu 1");

                Menu m11 = new Menu ("Menu 1.1");
```

```
Menu m2 = new Menu ("Menu 2");

MenuItem mi1 = new MenuItem ("Item 1.1");

MenuItem mi11 = new MenuItem ("Item 1.1.1");

CheckboxMenuItem mi2 = new CheckboxMenuItem("Item 2.1");

mb.add(m1);

mb.add(m2);

m1.add(mi1);

m1.add(m11);

m11.add(mi11);

m2.add(mi2);

setMenuBar(mb);



// Componentes
```

```
Label lblBoton = new Label("Boton:");

Label lblCheck = new Label("Checkbox:");

Label lblChoice = new Label("Choice:");

Label lblText = new Label("TextField:");



Button btn = new Button ("Boton");

Checkbox chk = new Checkbox ("Checkbox", true);

Choice ch = new Choice();

ch.addItem("Opcion 1");

ch.addItem("Opcion 2");

TextField txt = new TextField("Texto");
```

```
add(lblBoton);

add(btn);

add(lblCheck);

add(chk);

add(lblChoice);

add(ch);

add(lblText);

add(txt);



// Evento para cerrar la ventana

addWindowListener(new WindowAdapter()

{

        public void windowClosing (WindowEvent e)
```

```
                {

                        System.exit(0);

                }

        });

}



/**

  * Main

  */

public static void main (String[] args)

{

        EjemploAWT e = new EjemploAWT();

        e.show();
```

```
        }

}
```

El código nos muestra cómo se crea una clase, que es una ventana principal (heredada de *frame*), y define un gestor que es un *GridLayout,* con cuatro filas y dos columnas. En ellas, vamos colocando etiquetas (Label), botones (Button), casillas de verificación (Checkbox), listas desplegables (Choice) y cuadros de texto (TextField). Además, se crea un menú con diferentes opciones.