Unidad 2

Lenguaje de programación en diseño de interfaces

Desarrollo de interfaces

Índice

2.1. Programación orientada a objetos
2.1.1. Clases
2.1.2. Propiedades o atributos
2.1.3. Métodos

2.2. Programación de eventos

2.3. Librerías de componentes
2.3.1. AWT
2.3.2. Swing

2.4. Herramientas de edición de interfaces

2.5. Contenedores
2.5.1. Layout manager
2.5.2. FlowLayout
2.5.3. GridLayout
2.5.4. BorderLayout
2.5.5. GridBagLayout

2.6. Componentes
2.6.1. JButton
2.6.2. JLabel
2.6.3. JTextField
2.6.4. JCheckBox
2.6.5. JRadioButton
2.6.6. JComboBox

2.7. Diálogos

Introducción

En este tema nos introduciremos en la fabricación de las ventanas de una interfaz.

Veremos que tipo de elementos podemos emplear para facilitar nuestra labor, como el empleo de la orientación a objetos, el empleo de las librerías como las de java o el uso de herramientas de edición como es Eclipse.

Centrándonos en este último descubriremos que son los Layout, en todos sus diferentes tipos y que cualidades y características posee cada uno.

Finalmente cerraremos con el estudio de algunos de los elementos que podemos encontrar en los editores y que nos facilitarán nuestro trabajo al ser elementos ya conformados y listos para su uso, como son los botones, desplegables, etc.

Al finalizar esta unidad

- Conoceremos el uso de los programas orientados a objetos para la construcción de interfaces gráficas.
- Estudiaremos las diferentes funciones de los editores gráficos.
- Describiremos los diferentes elementos que podemos encontrar en herramientas de diseño como Eclipse.
- Distinguiremos algunas de las propiedades relevantes de los elementos empleados.
- Sabremos desarrollar algunos ejemplos de ventanas sencillas.

2.1. Programación orientada a objetos

Como ya hemos visto las interfaces gráficas son el puente que conecta a los usuarios con las computadoras, por lo que su importancia es máxima. Con el fin de optimizar estas se emplean diferentes recursos, uno de ellos es la programación orientada a objetos, la cual permite que los elementos de una interfaz tengan sus propias identidades y comportamientos propios.

Con el fin de que este tipo de programación funcione se emplea el uso de *layout*, los cuales permiten colocar los componentes y moverlos cuando es necesario.

La combinación de ventanas, ya que puede haber más de una, mediante JFrame para la ventana principal y JDialog para las ventanas secundarias.

2.1.1. Clases

La clase es la denominación que reciben los diferentes tipos de objetos, los cuales cuentan con la misma formación estructural y metodológica. Desde la clase podemos extraer tantas instancias como sea necesario para nuestro trabajo.

La instanciación se realiza mediante los constructores, el nombre de cada una de estas será el nombre de la clase seguido del de la instancia entre paréntesis.

2.1.2. Propiedades o atributos

Los atributos definirán, preferiblemente inequívocamente, cada una de las instancias de una clase mediante una serie de valores. Por ejemplo, ante una clase "Usuario" los atributos pueden ser "cuenta", "contraseña" y "DNI". Los valores de estos tres atributos juntos identificarán de manera inequívoca la instancia que necesitamos.

Si empleamos java la estructura de cada clase contaría con atributos, constructory métodos.

2.1.3. Métodos

El método u operación de una clase es la acción que puede realizar, esta acción puede ser tanto una realización física como una entrega de datos, en este último caso es necesario incluir el tipo de datos entregado como un atributo.

La devolución de datos generalmente se hace mediante números, ya sea con el valor exacto si este debe expresarse en números o empleándolos como código, normalmente usando el "0" como un valor positivo y el resto de números para otros valores. Los resultados de los métodos pueden llegar a modificar los valores de los atributos.

2.2. Programación de eventos

Un evento puede ser cualquier entrada de información como puede ser la pulsación de un botón, lo cual puede provocar que la página se cambie. Para que esto suceda es necesario enlazar ambas ventanas, una vez ya hechas, mediante elementos de conexión como son los botones, los cuales se pueden encontrar en JPanel o *layout*.

El evento debe crearse en el botón y conectarlo a la que será la ventana emergente, en el caso de emplear programas como NetBeans esto se puede realizar fácilmente mediante los comandos preprogramados, en el caso de querer realizarlo de forma manual se deberá incorporar el código pertinente.

Quedando el código de este botón como:

```
JButton btnNewButton = new JButton("New button");
btnNewButton.addActionListener(new ActionListener() {
    private Ventana ventana1;

    public void actionPerformed(ActionEvent e) {
            ventana1 = new Ventana();
            ventana1.setVisible(true);
```

Imagen 2. Ejemplo del código de una acción

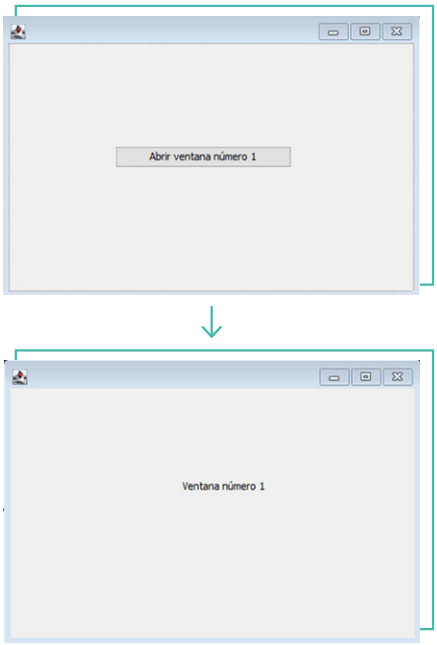

Imagen 1. Ejemplo de botón con ventana emergente

2.3. Librerías de componentes

Lenguajes como Java pueden llevar incorporadas librerías propias, con las cuales poder reutilizar código y, por tanto, ahorrándonos mucho tiempo. En el caso del desarrollo de interfaces las librerías llegan a ser fundamentales. En el caso de Java podemos encontrar las librerías de AWT y Swing.

2.3.1. AWT

AWT, Abstract Window Toolkit, permite la creación de interfaces gráficas con el paquete java.awt, al darnos acceso a las clases Component y Container, las cuales definen los controles y la pantalla de la aplicación en desarrollo respectivamente.

2.3.2. Swing

Una evolución de la anterior AWT que incorpora nuevas herramientas al tiempo que elimina algunas limitaciones, dándonos una mayor libertad de trabajo. Nos permite crear interfaces al añadir elementos desde un directorio al tiempo que el código asociado se genera automáticamente.

Para poder trabajar con estas librerías es necesario importarlas, lo cual se realiza con el comando import más el nombre de la librería:

```
import javax.swing.*;
import java.awt.*;
```

	Comparación AWT y SWING			
	Componentes	Eventos manejados por...	Apariencia en S.O.	Apariencia
AWT	Del S.O.	S.O.	Depende del S.O.	Estática
SWING	Propios	Java	Siempre igual	Personalizada

2.4. Herramientas de edición de interfaces

Existen múltiples herramientas con las que podemos trabajar, una de las más asequibles en cuanto a nivel, la cual además es gratuita, es Eclipse, la cual podemos descargar desde su página www.eclipse.org.

Es necesario, ya que se trabaja con Java, poseer Java Development Kit, que podemos descargar desde Oracle.

2.5. Contenedores

Los componentes, en especial en el diseño de interfaces, siempre deben estar localizados y distribuidos de la mejor manera posible, esta tarea se facilita en gran medida mediante el empleo de contenedores, los cuales no son más que componentes que pueden contener otros componentes.

2.5.1. Layout manager

Traducido como manejador de componentes, los *layout manager* permiten modificar la localización de los componentes y su tamaño, con el fin de que estos se coloquen donde son necesarios, ya que en caso de no especificarse por un *layout manager* los componentes ocuparán íntegramente el contenedor.

Existen distintos *layout* de los cuales podemos hacer uso en función de nuestras necesidades.

2.5.2. FlowLayout

Nos permite colocar componentes en una misma fila y, por lo tanto, contiguos, también nos permite modificar diversos valores de propiedades que afectan a su ubicación como son la alineación y la separación entre elementos vertical y horizontal, los cuales se señalan mediante las entradas `setAlignment`, `setVgap` y `setHgap` respectivamente.

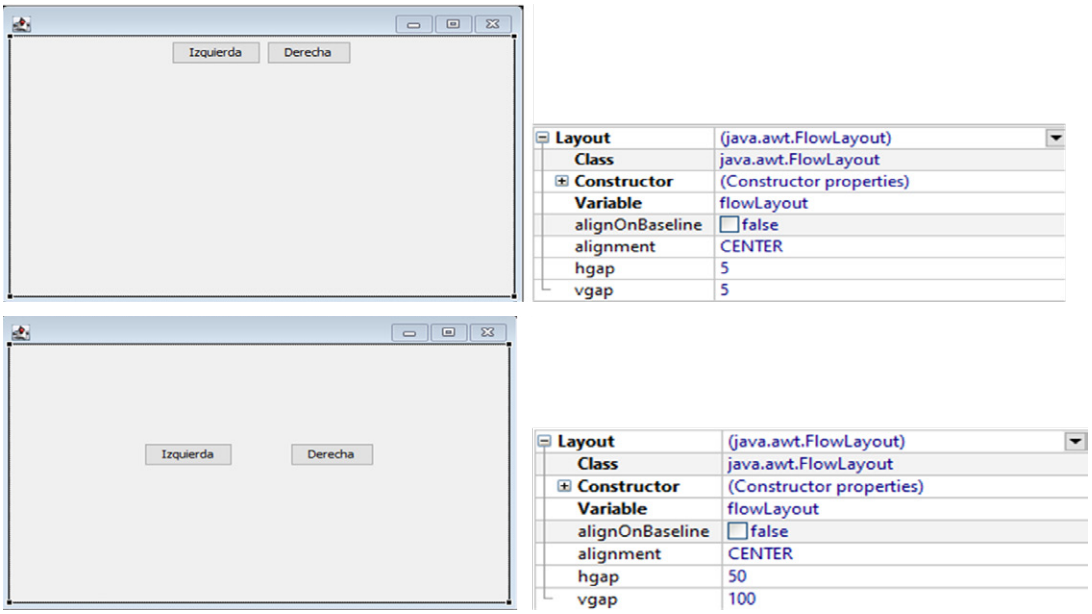

Imagen 3. Ejemplo de FlowLayout

2.5.3. GridLayout

Permite colocar comandos siguiendo el patrón de una tabla, yendo de izquierda a derecha y de arriba hacia abajo.

A los valores de separación anteriores, `vgap` y `hgap`, añade nuevas propiedades como `cols`, o `columns` y `rows` que, respectivamente, señalan el número de columnas y filas, cuyo valor debe expresar en un número exacto y entero.

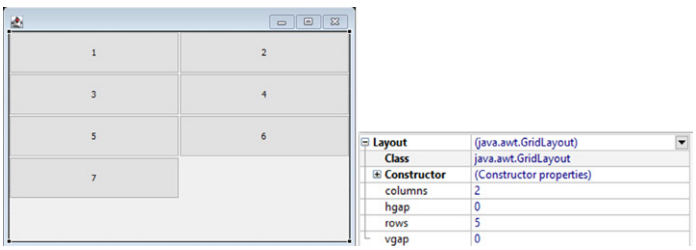

Imagen 4. Ejemplos de GridLayout

2.5.4. BorderLayout

Permite colocar elementos en el centro de la ventana, así como en los cuatro bordes, Empezando por el superior y girando en el sentido de las agujas del reloj se denominan NORTH, SOUTH, EAST y WEST, con CENTER como el que se encuentra en el centro.

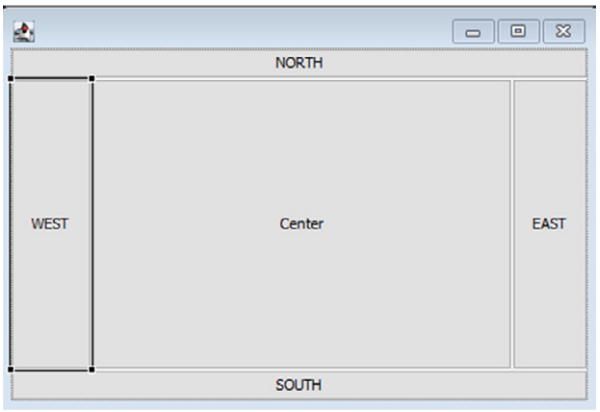

Imagen 5. Ejemplos de BorderLayout

2.5.5. GridBagLayout

En mucho más flexible que GridLayout, permitiendo colocar cada componente en la celda deseada de una cuadricula ya hecha, al tiempo que se le asocia otro componente de tipo `GridBagConstraints`. Las columnas y filas de la cuadrícula se encuentran numeradas comenzando desde cero.

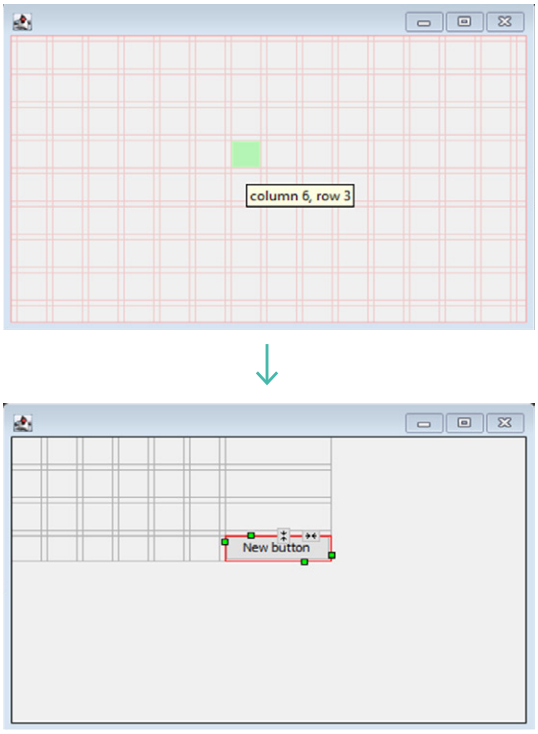

Imagen 6. Ejemplo de GridBagLayout

2.6. Componentes

Existen una gran multitud de componentes con los que trabajar, pero en este tema solo mostraremos los más relevantes entre ellos.

2.6.1. JButton

En Java permite crear un objeto tipo botón para una interfaz el cual contendrá las siguientes propiedades con relación a su apariencia.

Propiedades de JButton	
Propiedad	**Explicación**
`font`	Fuente y tamaño de la letra
`foreground`	Color del texto
`horizontalAlignment`	Alineación del texto horizontal
`verticalAlignment`	Alineación del texto vertical
`text`	Texto del botón
`background`	Si es opaco el color de fondo del botón
`enabled`	Valor, activo o no, del botón (true/false)
`icon`	Imagen como fondo del botón

2.6.2. JLabel

JLabel permite la inclusión de texto, iconos o imágenes que no reaccionan ante eventos, por lo que se emplean para añadir elementos de información no reactivos a una ventana, para colocar información que complemente la de un botón cercano, etc.

Puede contener las siguientes propiedades.

Propiedades de JLabel	
Propiedad	**Explicación**
`enabled`	Habilitar la etiqueta creada
`foreground`	Color del texto
`horizontalAlignment`	Alineación horizontal del texto
`verticalAlignment`	Alineación vertical del texto
`text`	Texto
`icon`	Imagen
`font`	Fuente y tamaño de letra
`background`	Color de la etiqueta deshabilitada

2.6.3. JTextField

Sirve como contenedor para una línea de texto, con el valor, no necesariamente exacto, marcado por `columns`. La capacidad variará en función de la línea de texto escrita.

Propiedades de JTextField	
Propiedad	**Explicación**
`enabled`	Habilitar la etiqueta creada
`foreground`	Color del texto
`horizontalAlignment`	Alineación horizontal del texto
`columns`	Tamaño de la caja de texto
`text`	Texto
`editable`	Permite modificar el contenido
`font`	Fuente y tamaño de letra
`background`	Color de la etiqueta deshabilitada

2.6.4. JCheckBox

Son elementos que incorporan una casilla, cuadrada, que puede ser seleccionada por el usuario.

A las propiedades anteriores añade `selected` con los valores `true` o `false` en función de si la casilla se marca o no.

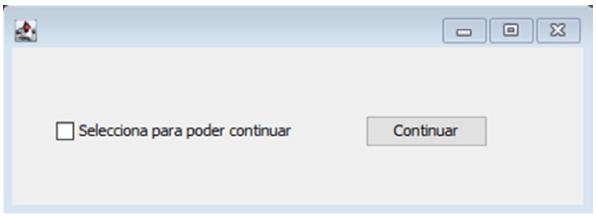

Imagen 7. Ejemplo de JCheckBox

2.6.5. JRadioButton

Permite la selección de opciones como JCheckBox, pero con múltiples opciones de las que solo se podrá seleccionar una, ya que son mutuamente excluyentes, lo cual se consigue con el ButtonGroup el cual agrupo diferentes opciones.

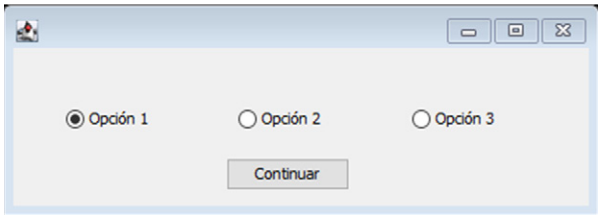

Imagen 8. Ejemplo de JRadioButton

2.6.6. JComboBox

Los JComboBox, similar al elemento exterior, nos permite seleccionar una única opción entre varias, pero esta vez desde un desplegable.

Podemos añadir las diferentes opciones desde la propiedad `model`, con el número máximo de opciones determinado por maximumRowCount. La propiedad selectedIndex definirá el valor que se mostrará de forma predeterminada.

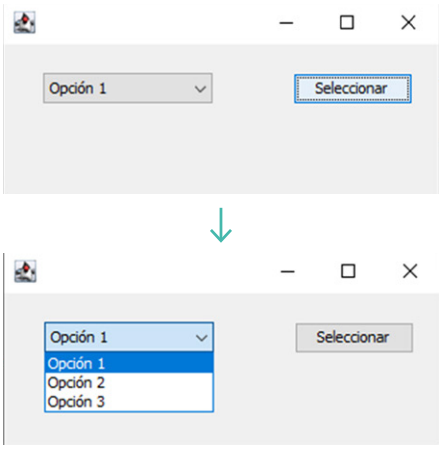

Imagen 9. Ejemplo de JComboBox

2.7. Diálogos

Es normal crear ventanas secundarias mediante JDialog, ya que permite enlazar un elemento padre a una ventana secundaria.

Las ventanas JDialog siempre se situarán por encima de su ventana padre sin importar el tipo de esta.

Su creación no difiere de la del resto de ventanas, siguiendo la dirección: File, New, Other, WindowBuilder y JDialog.

Este tipo de ventana incluirá por defecto los botones de Ok y Cancel.

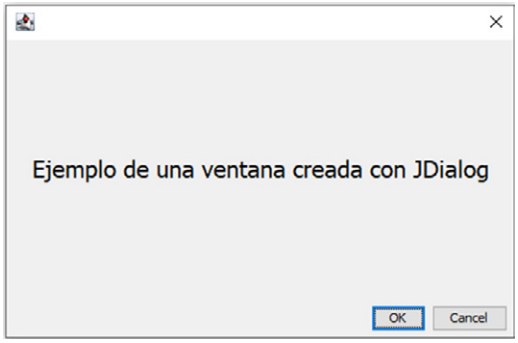

Imagen 10. Ejemplo de ventana JDialog

Podemos encontrar dos tipos de estas ventanas:

- **Diálogos modales:** evitan la apertura de otras ventanas hasta que la actual se cierre.
- **Diálogos no modales:** no afecta de ningún modo a otras ventanas, aunque se encuentre abierta.

La propiedad `modal` determinará según su valor, `true` o `false`, si la ventana JDialog es modal o no modal respectivamente.

www.universae.com