MEDAC
Instituto Oficial de Formación Profesional

DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Clases y componentes

02

ÍNDICE

ÍNDICE

OBJETIVOS

Crear una interfaz gráfica utilizando los asistentes de un editor visual.

Utilizar las funciones del editor para ubicar los componentes del interfaz.

Modificar las propiedades de los componentes para adecuarlas a las necesidades de la aplicación.

Asociar las acciones correspondientes a los eventos.

Analizar el código generado por el editor visual.

Modificar el código generado por el editor visual.

Desarrollar una aplicación que incluye la interfaz gráfica obtenido.

/ 1. Introducción y contextualización práctica

El desarrollo de interfaces gráficas permite la creación del canal de comunicación entre el usuario y la aplicación y, por esta razón, su diseño requiere de especial atención. En la actualidad, las herramientas de desarrollo permiten implementar el código relativo a una interfaz a través de vistas de diseño que facilitan y hacen más intuitivo el proceso de creación.

Existen numerosos componentes, que son tipos de elementos que pueden ser incluidos en una interfaz simulando una comunicación bidireccional, en la que la aplicación recibe diferentes entradas de datos, en función del componente escogido. Por ejemplo, sería posible que el usuario introduzca texto en una caja, seleccione un valor en un menú desplegable o se conecte a otras ventanas a través de la acción sobre un botón, entre otras.

En este tema se verán en detalle los principales tipos de componentes, así como sus características más importantes. La distribución de este tipo elementos depende de los llamados Layout, los cuales permiten definir la ubicación exacta de los elementos.

Una misma aplicación puede presentar más de una ventana. En función de la finalidad de la misma encontramos JFrame y JDialog. La segunda establece los llamados diálogos modales o no modales, elementos clave en el desarrollo de interfaces. La combinación de tipos de ventanas y de elementos de diseño es infinita.

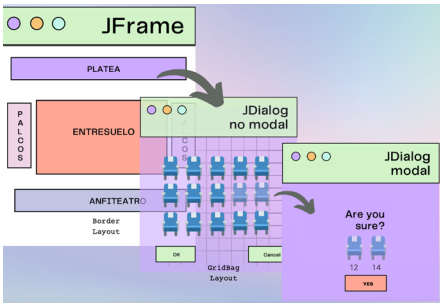

Fig. 1. Diagrama conexión tipos de ventanas.

Escucha el siguiente audio donde contextualizaremos el caso práctico del tema y que encontrarás resuelto en el último apartado de la unidad:

Audio intro. "Los componentes de una interfaz gráfica"

https://bit.ly/3fxF3rW

/ 2. Explotación del área de diseño

Conocer en profundidad todas las funcionalidades del área de la vista de diseño es fundamental para un correcto desarrollo.

En primer lugar, se crea la ventana utilizando la clase contenedora JFrame desde el menú File. Tras crearla, lo primero que aparece es el contenido de la pestaña 'Source', es decir, el código en Java del JFrame creado. Para acceder a la vista de diseño solo hay que pulsar en la pestaña **Design**. Recordemos que las partes principales de la vista de diseño son:

- **Zona de diseño:** sobre la que se sitúan los componentes de la interfaz.
- **Palette:** aquí se encuentran todos los elementos utilizados para la implementación de la interfaz, que son colocados sobre la ventana de la zona de diseño. Cada vez que tomamos uno de estos elementos y lo colocamos en la pestaña 'Source', aparece su código de programación.
- **Components:** mapa de navegación que muestra un resumen de todos los elementos insertados en la zona de diseño.
- **Propiedades:** si se selecciona cualquier componente en esta ventana, se muestran todas las propiedades del elemento que permiten definir su apariencia, entre otras. En cambio, si no se pulsa sobre ningún elemento aparece en blanco.

2.1. Insertar elementos

Para colocar cualquier elemento basta con pulsar sobre él en la paleta de componentes y llevarlo hasta la posición exacta de la zona de diseño. Como se puede ver en la siguiente imagen, la ventana queda dividida en varias zonas de color verde. Si movemos el puntero del ratón sobre ellas, cambiarán a amarillo; para colocar el elemento basta con pulsar sobre la caja amarilla.

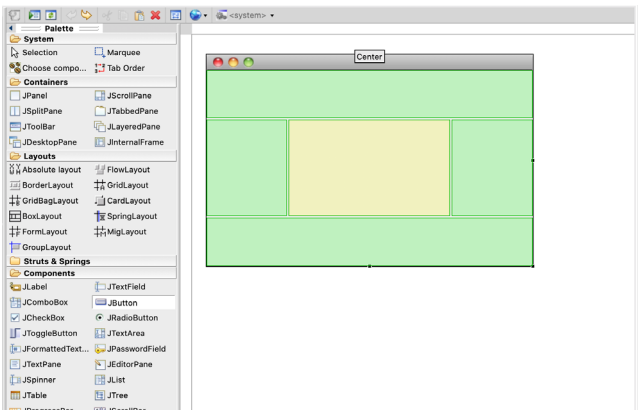

Fig. 2. Área de diseño para inserción de componentes.

2.2. Eliminar elementos

Para eliminar un elemento, ya sea componente, contenedor o de cualquier otro tipo, basta con seleccionarlo desde la zona de diseño o desde el mapa de navegación de Components, pulsar con el botón derecho sobre él y seleccionar Borrar (o Delete). También se puede realizar el borrado desde la pestaña de código (Source), localizando los métodos relativos al componente y eliminarlos. La primera opción es mucho más rápida.

/ 3. Clases, propiedades y métodos en Poo

3.1. Clases

Una clase representa un conjunto de objetos que comparten una misma estructura (ATRIBUTOS) y comportamiento (MÉTODOS). A partir de una clase se podrán instanciar tantos objetos correspondientes a una misma clase como se quieran. Para ello se utilizan los constructores.

Para llevar a cabo la instanciación de una clase y así crear un nuevo objeto, se utiliza el nombre de la clase seguido de un par de paréntesis. Un constructor es sintácticamente muy semejante a un método. El constructor puede recibir argumentos; de esta forma, podrá crearse más de un constructor en función de los argumentos que se indiquen en su definición. Aunque el constructor no haya sido definido explícitamente, en Java siempre existe un constructor por defecto que presenta el nombre de la clase y no recibe ningún argumento.

3.2. Métodos

Los métodos definen el comportamiento de un objeto, esto quiere decir que toda aquella acción que se quiera realizar sobre la clase tiene que estar previamente definido en un método.

Los métodos pueden recibir argumentos o no; además, en función de su definición, devolverán un valor o realizarán alguna modificación sobre los atributos de la clase.

3.3. Propiedades o atributos

Un objeto es una cápsula que contiene todos los datos y métodos ligados a él. La información contenida en el objeto será accesible solo a través de la ejecución de los métodos adecuados, creándose una interfaz para la comunicación con el mundo exterior.

Los atributos definen las características del objeto. Por ejemplo, si se tiene una clase círculo, sus atributos podrían ser el radio y el color. Estos constituyen la estructura del objeto, que posteriormente podrá ser modelada a través de los métodos oportunos.

La estructura de una clase en Java quedaría formada por los siguientes bloques, de manera general: atributos, constructor y métodos.

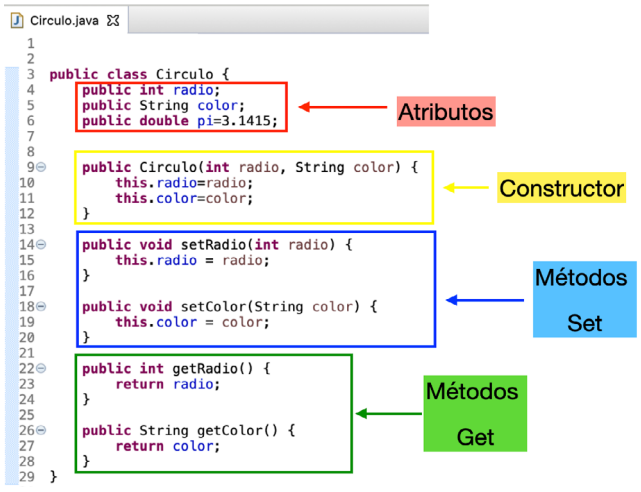

Fig. 3. Estructura básica clase Java.

/ 4. JFrame y JPanel

Como se vio en el tema anterior, una de las clases principales es la clase JFrame, destinada a la creación de la ventana para la interfaz. Es aconsejable utilizar el proceso de creación descrito para poder acceder a la vista de diseño (Design). Esta clase se utiliza, sobre todo, para definir la pantalla principal de una aplicación. Para generar las pantallas secundarias es común utilizar **JDialog**, con unas prestaciones prácticamente idénticas.

Al igual que ocurre con todas las clases en Java, es necesario utilizar un constructor para crear una instancia del objeto, en el caso de Jframe encontramos varios, como Jframe() o Jframe (String nombreVentana), entre otros.

Otros métodos importantes en Jframe son aquellos que permiten establecer las dimensiones exactas de la ventana, la acción de cierre y habilitar su visibilidad.

Complementando a JFrame, tenemos un panel o lienzo denominado **JPanel**. Este es un bloque «invisible» que se sitúa sobre una ventana. Consiste en un contenedor de componentes sobre el que vamos a ubicar todos los elementos necesarios, sin tener que colocarlos directamente sobre la ventana JFrame.

Para crear un panel, desde el menú **File, New**, pulsamos **Other** y se busca en la carpeta **WindowBuilder** el tipo **JPanel.**

Fig. 4. JPanel y JFrame.

Gracias a los paneles podemos tener más organizada la interfaz gráfica. La distribución de estos paneles constituye un sistema de capas o Layout que se verá más adelante.

El resto de los componentes se colocan dentro del lienzo creado de la forma descrita en el apartado 2.1. Por ejemplo, si se crea un nuevo JPanel dentro del cual se coloca una etiqueta de texto, el código implementado sería de la forma:

```
//Crea un JPanel
JPanel panelSecundario = new JPanel();
//Se coloca dentro de otro JPanel principal
panel.add(panelPpal);
//Crea una etiqueta
JLabel jlabel1 = new JLabel("hola");
//La coloca dentro del JPanel secundario
panelSecundario.add(jlabel1);
```

Código 1. Creación e inserción de un elemento JPanel.

/ 5. JDialog

Las aplicaciones que solo utilicen una pantalla implementarán su interfaz solo con un elemento JFrame; pero cuando la aplicación que se está desarrollando presenta más de una ventana, las de tipo secundario se crearán utilizando JDialog, puesto que esta sí permite tener un elemento padre, es decir, un elemento principal a partir del cual se accede a la venta secundaria.

Las ventanas tipo JDialog siempre quedarán situadas por encima de sus padres, ya sean de tipo JDialog o JFrame.

La creación de este tipo de ventanas se realiza de forma similar a la de tipo JFrame: desde el menú File y New, seleccionamos Other y, a continuación, dentro de la carpeta WindowBuilder, pulsamos sobre JDialog.

```
public holaDialog() {
    setBounds(100, 100, 450, 300);
    getContentPane().setLayout(new BorderLayout());
    contentPanel.setLayout(new FlowLayout());
    contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
    getContentPane().add(contentPanel, BorderLayout.CENTER);
    {
        JPanel buttonPane = new JPanel();
        buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
        getContentPane().add(buttonPane, BorderLayout.SOUTH);
        {
            JButton okButton = new JButton("OK");
            okButton.setActionCommand("OK");
            buttonPane.add(okButton);
            getRootPane().setDefaultButton(okButton);
        }
        {
            JButton cancelButton = new JButton("Cancel");
            cancelButton.setActionCommand("Cancel");
            buttonPane.add(cancelButton);
        }
    }
}
```

Código 2. Método creación de la ventana JDialog de la Figura 5.

En la figura anterior se muestra el código por defecto que se genera al crear un JDialog de la forma descrita. En este primer diseño aparecen dos botones, Ok y Cancel.

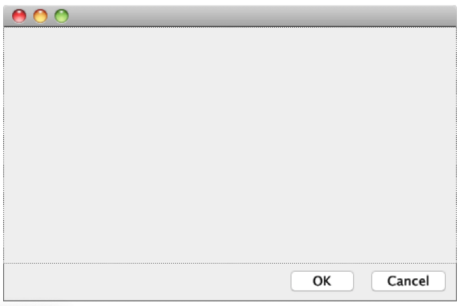

Fig. 5. Ventana creada código 2. JDialog.

Los **diálogos modales** son aquellos que no permiten que otras ventanas de diálogo se abran hasta que la que se encuentra abierta se haya cerrado, por ejemplo, un programa que queda a la espera de la selección de una opción para poder continuar, como la selección del número de asiento en una aplicación para la compra de billetes de tren.

Por lo tanto, los diálogos de tipo no modal sí permitirán que haya tantos JDialog abiertos como se desee. Para indicar a cuál de estos tipos pertenecen, utilizamos el flag de modal del constructor de JDialog, indicando a true para modal, y false para no modal.

```
JDialog ventanaSec = new JDialog(f, “Dialog”, true);
```

Código 3. Creación de un elemento JDialog de tipo modal.

/ 6. Conexión entre ventanas. Eventos

Para poder crear una conexión entre dos o más ventanas, en primer lugar, es necesario crearlas todas, ya sean de tipo Jframe o JDialog. El paso de una ventana a otra se produce tras la ocurrencia de un evento. Habitualmente, la pulsación sobre un botón.

Tras la creación de las ventanas se sitúan los botones de conexión y se modifican sus propiedades de apariencia. Este elemento puede situarse dentro de un layout o de un JPanel. Para crear el evento escuchador asociado a este botón, basta con hacer doble clic sobre él y, de forma automática, se generará el siguiente código en la clase de la ventana de la interfaz donde estamos implementando el botón conector.

```
JButton btnNewButton = new JButton("Púlsame");
btnNewButton.addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
        }
});
panel.add(btnNewButton);
```

Código 4. Evento implementado para conexión de JButton.

Es importante tener en cuenta que, siempre que se utilicen nuevas ventanas, hay que ponerlas visibles utilizando el método setVisible(boolean visibilidad), donde el valor que recibe por parámetro será true en el caso de hacerla visible y false en el caso contrario.

En el siguiente ejemplo, al pulsar el botón Jugar desde la ventana principal implementada con una clase JFrame, nos lleva a una segunda ventana también de tipo JFrame, la cual muestra un mensaje en una etiqueta de texto.

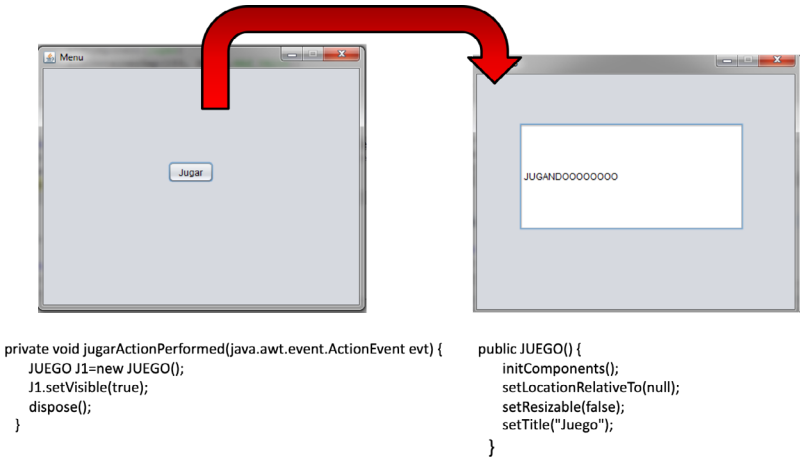

Fig. 6. Conexión de ventanas con ActionListener.

Cuando se detecta la pulsación del botón como evento, desde la clase principal se crea una nueva instancia del objeto Juego, también de tipo JFrame y se especifica como visible. Finalmente, en este ejemplo, se utiliza el método dispose(), el cual cierra la ventana principal y solo mantiene abierta la segunda.

/ 7. Componentes

Existe un amplio abanico de componentes. En este apartado se verán los que constituyen la gran parte de la interfaz. Los componentes son los elementos que se sitúan en la ventana, directamente sobre el JFrame y JDialog o sobre un JPanel. Hay que prestar especial atención a aquellas propiedades que tienen un mismo nombre, puesto que no realizan la misma acción en todos los elementos.

7.1. JButton

Permite crear un objeto de tipo botón dentro de una interfaz gráfica en Java. Las propiedades de este elemento son:

background	El color de fondo del botón. Se muestra solo si es opaco
enabled	True/false determina si el botón está activo o no
font	Fuente del tipo de letra y tamaño
foreground	Color del texto
horizontalAlignment **verticalAlignment**	Alineación vertical y horizontal del texto con respecto al botón
text	Texto que aparece dentro del botón
icon	Carga imagen como fondo del botón

Tabla 1. Propiedades JButton.

Para utilizar este componente es necesario importar los paquetes: **import javax.swing e import java.awt.event.** El segundo se utiliza para que, al pulsar el botón, se detecte la ocurrencia de un evento y se derive una acción.

En el siguiente ejemplo, se crea un botón y, desde el menú de propiedades, se modifican de la siguiente forma:

El código que implementa las propiedades anteriormente modificadas es el siguiente. Si se cambia algún valor desde la pestaña Source, estos también se reflejarán en las propiedades de la vista de diseño del componente que describen.

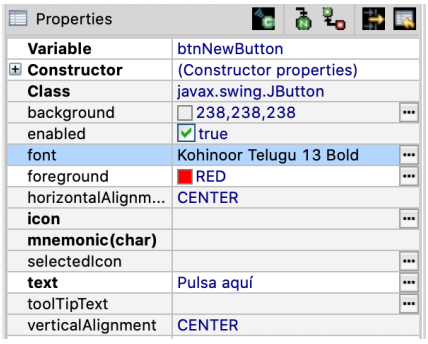

Fig. 7. Propierties JButton en Vista diseño Eclipse.

```
JButton btnNewButton = new JButton("Pulsa aquí");
panel.add(btnNewButton);
btnNewButton.setFont(new Font("Kohinoor Telugu", Font.
BOLD, 13));
btnNewButton.setForeground(Color.RED);
```

Código 5. Código equivalente propiedades modificadas Figura 7.

Vídeo 1. "Modificación de propiedades desde Design y Source"
https://bit.ly/395mCIK

7.2. JLabel

Este elemento es uno de los más sencillos de aplicar y que, al mismo tiempo, más utilidad reporta. No solo se trata de un elemento de texto, sino que este contenedor puede llegar a albergar también imágenes, iconos o texto. Sus propiedades características son:

background	El color de la etiqueta si está deshabilitada
enabled	Habilita la etiqueta
font	Fuente del tipo de letra y tamaño
foreground	Color del texto si etiqueta habilitada
horizontalAlignment verticalAlignment	Alineación vertical y horizontal del texto con respecto a la caja de la etiqueta
text	Texto que aparece dentro de la etiqueta
icon	Permite cargar una imagen

Tabla 2. Propiedades JLabel.

Hay que prestar especial atención a los valores **background y foreground**, ambos definen el color del texto: el primero cuando está habilitado y el segundo cuando no lo está.

7.3. JTextField

El elemento JTextField, se utiliza como contenedor de una línea de texto, el tamaño queda definido por el valor del atributo 'columns'. No se trata de un valor exacto en cuanto a número de caracteres, sino que está definiendo su ancho, por lo tanto, en función del carácter que se escriba, la capacidad variará.

background	Color de fondo de la caja de texto
columns	Tamaño de la caja de texto
Enabled	Habilita el campo de texto
editable	Permite al usuario modificar el contenido
Font	Fuente del tipo de letra y tamaño
Foreground	Color del texto
thorizontalAlignment	Alineación horizontal del texto
Text	Texto que aparece al inicio en la caja

Tabla 3. Propiedades JTextField.

7.4. JCheckBox

Los elementos de tipo casilla o CheckBox son elementos que se presentan junto a una pequeña caja cuadrada y que pueden ser marcados por el usuario. Presenta unas propiedades similares a los casos anteriores, añadiendo algunos nuevos atributos como 'selected', el cuál puede ser de valor true o false. El primero indicará que la casilla se muestre marcada por defecto y, si es false, aparecerá sin marcar.

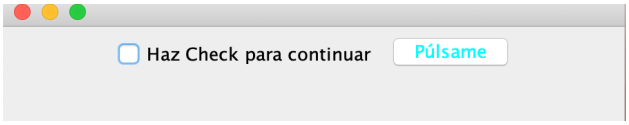

Fig. 8. Interfaz ejemplo con JCheckBox y JButton.

7.5. JRadioButton

Los elementos de tipo JRadioButton se utilizan habitualmente en el desarrollo de interfaces para indicar varias opciones, de las que solo se podrá escoger una, es decir, que resultarán excluyentes. Las propiedades que presenta son iguales a la del elemento 'JCheckBox'.

Ahora bien, cuando insertamos un elemento JRadioButton en una interfaz su funcionamiento va a ser muy parecido al de un elemento de tipo check. Para conseguir un comportamiento excluyente, es necesario utilizar un objeto tipo **ButtonGroup.**

La creación de un elemento ButtonGroup nos permite asociar a este grupo tantos elementos como se deseen; de esta forma, todos aquellos que queden agrupados resultarán excluyentes entre sí, puesto que pertenecen al mismo grupo.

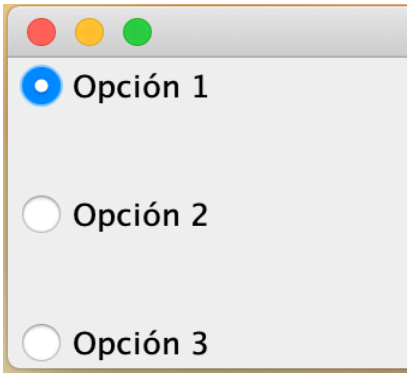

Fig. 9. JRadioButton excluyentes con ButtonGroup.

En el siguiente ejemplo se han creado tres elementos y se han asociado en un un ButtonGroup llamado bg.

```
public holaMundo() {
//Se define el comportamiento de la ventana
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
setBounds(100, 100, 450, 300);
//Creación de JRadioButton y se ubican
JRadioButton r1 = new JRadioButton("Opción 1");
getContentPane().add(r1, BorderLayout.NORTH);
JRadioButton r2 = new JRadioButton("Opción 2");
getContentPane().add(r2, BorderLayout.WEST);
JRadioButton r3 = new JRadioButton("Opción 3");
getContentPane().add(r3, BorderLayout.SOUTH);
//Se agrupan los JRadioButton creados
ButtonGroup bg=new ButtonGroup();
bg.add(r1);
bg.add(r2);
bg.add(r3);
}
```

Código 6. Ejemplo JRadioButton excluyentes.

7.6. JComboBox

Finalmente, otro elemento común en la creación de interfaces son los **menús desplegables**, que se crean a través del componente **JComboBox**. Presenta unas propiedades muy parecidas al resto de componentes descritos.

Para insertar los valores que se mostrarán en el combo utilizando la vista de diseño, desde propiedades, seleccionamos 'model' y se abrirá una nueva ventana en la que se escriben en líneas separadas los valores del combo. El valor máximo de elementos mostrados en el combo queda establecido en la propiedad maximumRowCount. La propiedad selectedIndex permite al desarrollador indicar cuál es el valor que mostraría por defecto de entre todos los recogidos, siendo 0 la primera posición.

/ 8. Layout manager

Un **layout manager** (manejador de composición) permite adaptar la distribución de los componentes sobre un contenedor, es decir, son los encargados de colocar los componentes de una interfaz de usuario en el punto deseado y con el tamaño preciso. Sin los layout, los elementos se colocan por defecto y ocupan todo el contenedor. El uso de los layout nos permite modificar el tamaño de los componentes y su posición de forma automática.

8.1. Flow Layout

Flow Layout sitúa los elementos uno al lado del otro, en una misma fila. Permite dar valor al tipo de alineación (**setAlignment**), así como la distancia de separación que queda entre los elementos: en vertical (**setVgap**) y en horizontal (**setHgap**).

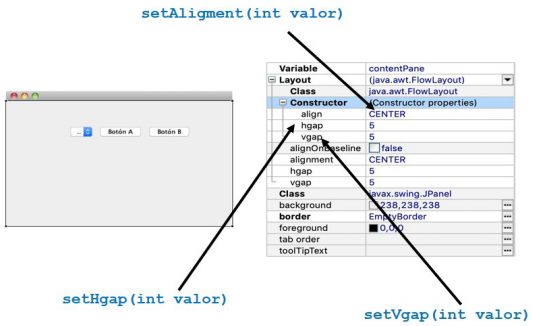

Fig. 10. Propiedades Flow Layout.

8.2. Grid Layout

Este nuevo layout permite colocar los componentes de una interfaz siguiendo un patrón de columnas y filas, simulando una rejilla. Al igual que en el caso anterior, es posible modificar el valor de la separación entre componentes. Las propiedades de este elemento incorporan la column y row, que definen el número exacto de columnas y filas.

Para la creación de este sistema de rejilla, se utiliza un constructor que necesita recibir el valor exacto de filas y columnas que tendría la interfaz, **GridLayout(int numFilas, int numCol).**

Cualquiera de los elementos 'Layout' presentan como propiedad común el valor de vgap y hgap, que definen la distancia entre elementos que se crea tanto en vertical como en horizontal.

Audio 1. "Tipos de Layouts"
https://bit.ly/2CgAsfg

8.3. Border Layout

BorderLayout permite colocar los elementos en los extremos del panel contenedor y en el centro. Para situar a cada uno de los elementos desde la vista de diseño basta con colocarlos en la posición deseada.

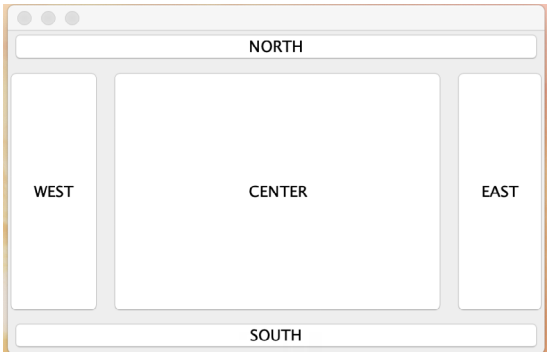

Fig. 11. Propiedades Border Layout y ejemplo botones en Border Layout.

Ahora bien, desde el código se sitúan atendiendo a su situación (NORTH, SOUTH, EAST, WEST, CENTER).

```
public interfazBorderLayout() {
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setBounds(100, 100, 450, 300);
	getContentPane().setLayout(new BorderLayout(10, 5));

	JButton btnNewButton = new JButton("WEST");
	getContentPane().add(btnNewButton, BorderLayout.WEST);

	JButton btnNewButton_1 = new JButton("NORTH");
	getContentPane().add(btnNewButton_1, BorderLayout.NORTH);

	JButton btnNewButton_2 = new JButton("EAST");
	getContentPane().add(btnNewButton_2, BorderLayout.EAST);

	JButton btnNewButton_3 = new JButton("SOUTH");
	getContentPane().add(btnNewButton_3, BorderLayout.SOUTH);

	JButton btnNewButton_4 = new JButton("CENTER");
	getContentPane().add(btnNewButton_4, BorderLayout.CENTER);
}
```

Código 7. Código de creación Figura 12 con JButton y JBorderLayout.

8.4. GridBag Layout

A diferencia del tipo Grid Layout visto, este permite un diseño más flexible, donde cada uno de los componentes que se coloquen tendrán asociado un objeto tipo GridBagConstraints.

Tras la inserción de este layout, será posible ubicar el elemento de una forma mucho más precisa, seleccionado la posición exacta de la rejilla. Por ejemplo, en este caso, se situará en la columna 2 y fila 2.

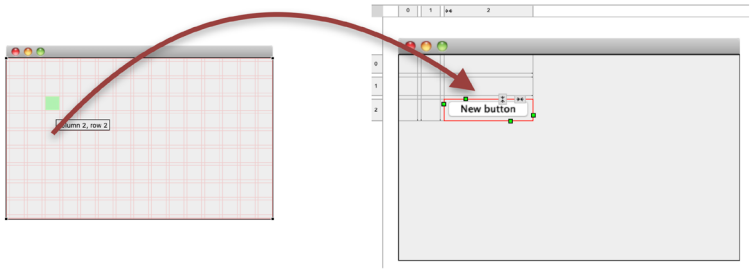

Fig. 12. Ejemplo botones en GridBag Layout

Vídeo 2. "Uso de Layouts"
https://bit.ly/3fyTcVR

/ 9. Caso práctico 1: "Acceso a una nueva ventana con usuario y contraseña"

Planteamiento: En este ejercicio se implementará una interfaz formada por dos ventanas conectadas. Si en la primera ventana el usuario y contraseña solicitados son correctos, se dará acceso a una segunda ventana de bienvenida.

El JFrame principal estará formado por:

- Dos **JLabel** 'Usuario' y 'Contraseña'.
- Un **JTexField** para introducir el nombre.
- Un **JPassWordField** para introducir la contraseña.
- Dos **JButton** 'Inicio' y 'Salir'.

El botón 'Inicio' tendrá implementada la función del evento escuchador, para que, al hacer clic, se compruebe si los datos son correctos y así abrir la ventana de 'Bienvenida'.

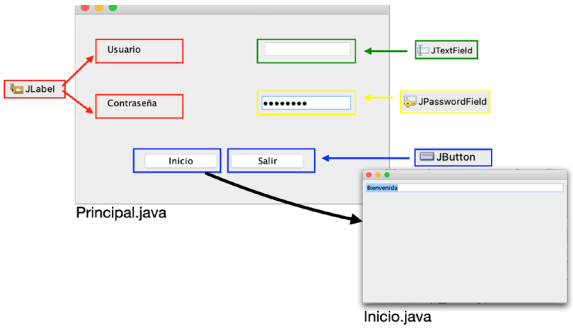

Fig. 13. Diseño a implementar Caso Práctico 1.

Nudo: Para implementar un **evento escuchador** asociado al botón Inicio, será necesario:

- Crear el botón 'Inicio' de tipo JButton.
- Asociar el evento escuchador.
- Recuperar el contenido de los campos del nombre de usuario y contraseña.
- Comprobar si son correctos y si, en este caso, corresponden con 'admin' y '1234'.
- Si es correcto, se crea una segunda ventana llamada 'acceso' de la clase Inicio.java y se hace visible.
- Si no es correcto, se muestra un mensaje de tipo 'Dialog' indicando que el usuario o contraseña no es correcto.

Desenlace:

```
JButton boton_inicio = new JButton("Inicio");
boton_inicio.addActionListener(new ActionListener() {
 public void actionPerformed(ActionEvent e) {
  String user,pwd;
  user=textUsuario.getText();
  pwd=String.copyValueOf(textContraseña.getPassword());
  if(user.contentEquals("admin")&&pwd.equals("1234")) {
     Inicio acceso = new Inicio();
     acceso.setVisible(true);
  }else {
    JOptionPane.showInternalMessageDialog(null, "El usuario o contraseña
es incorrecto");
  }
 }
});
```

Código 8. Código desenlace Caso Práctico 1.

El código completo de las clases Principal.java e Inicio.java se encuentran en el Anexo 1 del tema.

/ 10. Caso práctico 2: "Interfaz de reproductor de música"

Planteamiento: Para poder adaptar la distribución de los componentes de una interfaz, se dispone de los manejadores de composición, mejor conocidos como Layout. En este caso, vamos a practicar con el tipo GridLayout que permite colocar componentes de una interfaz siguiendo un patrón de columnas y filas, similar a una rejilla.

Nudo: Utilizando la composición de tipo GridLayout, se va a diseñar la interfaz de un reproductor de música. Para ello, será necesario:

- Crear el JFrame llamado 'reproductor'.
- Implementar el JPanel con la disposición 'GridLayout', estableciendo el número apropiado de columnas y filas mediante GridLayout(int numFilas, int numCol).

- Añadir los nueve botones al JPanel.
- Añadir el JPanel de tipo 'GridLayout' al reproductor de tipo JFrame.
- Realizar el empaquetado de los componentes de la ventana.
- Activar la visibilidad de la ventana.

```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
public class reproductor {
	public static void main(String[] args) {
		JFrame frame=new JFrame("reproductor");
		JPanel p1=new JPanel(new GridLayout(3,3));
		JButton button = new JButton("ON/OFF");
		p1.add(button);
		button = new JButton("PLAY");
		p1.add(button);
		button = new JButton("RECORD");
		p1.add(button);
		button = new JButton("PISTA ANTERIOR");
		p1.add(button);
		button = new JButton("PAUSE");
		p1.add(button);
		button = new JButton("PISTA POSTERIOR");
		p1.add(button);
		button = new JButton("REBOBINAR ATRÁS");
		p1.add(button);
		button = new JButton("STOP");
		p1.add(button);
		button = new JButton("REBOBINAR DELANTE");
		p1.add(button);
		frame.add(p1, BorderLayout.CENTER);
		frame.pack();
		frame.setVisible(true);
	}
}
```

Código 9. Código desenlace "Caso Práctico 2".

Desenlace: El resultado final de implementar el código anterior es una interfaz como la que se muestra en la imagen, compuesta por una matriz de seis botones dispuestos en tres filas y tres columnas.

/ 11. Resumen y resolución del caso práctico de la unidad

En este tema hemos estudiado el **paquete Swing**, el cual contiene todas las clases necesarias para programar todo tipo de componentes visuales. Swing es una extensión para AWT, un kit de herramientas de widgets utilizados para el desarrollo de interfaces gráficas en Java.

Aunque el número de elementos que se incorporan en la paleta (**Palette**) es muy amplio, en este tema se han descrito algunos de los más usuales, analizando sus principales propiedades. El abanico de elementos permite crear infinitas combinaciones que se adecuarán en cada caso a las especificaciones finales de la aplicación. Algunos de los elementos más importantes son **JButton, JRadioButton, JCheckBox, JLabel o JComboBox.**

Por lo otro lado, hemos visto que tener presentes las diferencias existentes entre **JFrame y JDialog** es fundamental, no en cuanto a su implementación, sino al uso que se les va a dar.

En próximos temas se verá más en detalle el análisis de eventos, pero ya hemos adelantado el código necesario sobre el que se establecerá la acción asociada ante la **pulsación de un botón,** por ejemplo, crear la **conexión necesaria** entre dos ventanas, ya sean de tipo JFrame o JDialog a vista de diseño o desde el código utilizando los métodos adecuados.

Finalmente, hemos comprobado que la elección de una buena combinación de elementos y el diseño de su distribución en el lienzo de la interfaz **utilizando los elementos de tipo Layout** determinará la usabilidad del diseño de una aplicación.

Resolución del caso práctico de la unidad.

Con los conocimientos adquiridos a lo largo del tema, podemos responder a las preguntas de diseño sobre la interfaz de compra de entradas de teatro que se proponía al principio del tema.

En cuanto a la mejor opción para distribuir los distintos tipos de entradas, a través del uso de BorderLayout será posible colocar los elementos en los extremos y en el centro del panel contenedor, de forma que la platea esté en la disposición NORTH, el anfiteatro en SOUTH, el palco 1 en EAST, el palco 2 en WEST y el entresuelo en CENTER.

Por otro lado, para simular la colocación de las butacas, los componentes que representan estos elementos se dispondrán utilizando GridBagLayout, que nos permite ubicar los elementos de una forma precisa, seleccionando la posición exacta, como si se tratase de una rejilla completa por filas y columnas.

Finalmente, en el último paso de confirmación de compra, es recomendable utilizar una ventana con diálogo modal, es decir, que no permita que otras ventanas se abran hasta que la que se encuentra abierta no se haya cerrado.

/ 12. Bibliografía

Fernández, A., García-Miguel, B. y García-Miguel, D. (2020). Desarrollo de Interfaces. (1.a ed.). Madrid: Síntesis.