DESARROLLO DE INTERFACES

TÉCNICO EN DESARROLLO DE APLICACIONES MULTIPLATAFORMA

Clases y componentes

Introducción

La distribución de este tipo elementos depende de los llamados **Layout. Definen** la **ubicación exacta de los elementos**.

Una aplicación puede presentar más de una ventana.

Encontramos JFrame y **JDialog**.

JDialog establece los llamados **diálogos modales o no modales.**

Área de diseño

Utilizamos el entorno Eclipse:

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

Se crea la **ventana** utilizando la clase contenedora **JFrame** desde el **menú File**.

Source: (**código en Java** del JFrame creado).

Design (Vista de diseño). 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.

REPASO LENGUAJES ORIENTADO A OBJETOS

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 utilizando 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.

Métodos

Los métodos definen el comportamiento de un objeto.

Los métodos pueden recibir argumentos o no.

Devolverán un valor o **realizarán alguna modificación** sobre los atributos de la clase.

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 (get(), set())**.

Los atributos definen las características del objeto.

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.

JFrame y JPanel

JFrame se utiliza, sobre todo, para definir la **pantalla principal** de una aplicación.

Para generar las **pantallas secundarias** es común utilizar **JDialog**, prácticamente idénticas a JFrame.

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.

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.

```
// Panel
JPanel panelSecundario = new JPanel();
// dentro de otro JPanel principal
panel.add(panelPpal);
// etiqueta
JLabel jlabel1 = new JLabel("hola");
// dentro del JPanel secundario
panelSecundario.add(jlabel1);
```

JDialog

Si 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 **ventana 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 DialogHola() {
		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);
			}
		}
	}
```

En este primer diseño aparecen dos **botones**, **Ok** y **Cancel**.

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.

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);

*el método: .**setModal(boolean b)** modificará este atributo.*

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);
```

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 “saludar” desde la ventana principal implementada con una clase JFrame, nos lleva a una segunda ventana también de tipo JDialog, la cual muestra un mensaje en una etiqueta de texto.

Clase Frame:

```
JButton btAbreDialogHola = new JButton("saludar");
btAbreDialogHola.addActionListener(new ActionListener() {
            public void actionPerformed(ActionEvent e) {
                  // en el método actionPerformed lanzamos el DialogHola
                  DialogHola hola = new DialogHola();
                  hola.setVisible(true);
                  dispose();
            }
});
```

Clase JDialog:

```
public DialogHola() {
```

```
            setBounds(100, 100, 450, 300);
            getContentPane().setLayout(new BorderLayout());
            setTitle("Hola");
            setResizable(false);
            contentPanel.setLayout(new FlowLayout());
            contentPanel.setBorder(new EmptyBorder(5, 5, 5, 5));
            getContentPane().add(contentPanel, BorderLayout.CENTER);
            saludo = new JLabel("¡¡¡Holaaaa qué pasaaaaa!!!");
            getContentPane().add(saludo);

            {
                  JPanel buttonPane = new JPanel();
                  buttonPane.setLayout(new FlowLayout(FlowLayout.RIGHT));
                  getContentPane().add(buttonPane, BorderLayout.SOUTH);
                  {
                        JButton okButton = new JButton("OK");
                        okButton.addActionListener(new ActionListener() {
                              public void actionPerformed(ActionEvent e) {
                              }
                        });
                        okButton.setActionCommand("OK");
                        buttonPane.add(okButton);
                        getRootPane().setDefaultButton(okButton);
                  }
                  {
                        JButton cancelButton = new JButton("Cancel");
                        cancelButton.setActionCommand("Cancel");
                        buttonPane.add(cancelButton);
                  }
            }
      }
```

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 específica como visible**. Finalmente, en este ejemplo, se utiliza el método **dispose(),** el cual **cierra la ventana principal** y solo **mantiene abierta la segunda**.

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.

JButton

Permite crear un objeto de tipo botón dentro de una interfaz gráfica en Java.

Algunas propiedades:

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

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.

Ahora crearemos un botón y modificaremos algunas propiedades:

```
JButton btAbreDialogHola = new JButton("saludar");
panelSecundario.add(btAbreDialogHola);
// establecemos tipo y tamaño de letra
btAbreDialogHola.setFont(new Font("Monospaced", Font.BOLD, 12));
// establecemos el color de fondo del botón
btAbreDialogHola.setBackground(Color.MAGENTA);
// determinamos el color del texto
btAbreDialogHola.setForeground(Color.white);
```

JLabel

Se trata de un elemento de texto. P**uede 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

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 está **deshabilitado**.

JTextField

El elemento JTextField, se utiliza como contenedor de **una línea de texto.**

Algunas de sus propiedades:

Background	Color de fondo de la caja de texto
Foreground	Color del 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
horizontalAlignment	Alineación horizontal del texto
Text	Texto que aparece al inicio en la caja

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**.

JRadioButton

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.

En el siguiente ejemplo se han creado dos elementos y se han asociado en un un ButtonGroup llamado bgSexo:

```
JRadioButton rbHombre = new JRadioButton("hombre");
rbHombre.setBounds(31, 201, 149, 23);
panelPrincipal.add(rbHombre);
JRadioButton rbMujer = new JRadioButton("mujer");
rbMujer.setBounds(31, 228, 149, 23);
ButtonGroup bgSexo = new ButtonGroup();
bgSexo.add(rbMujer);
bgSexo.add(rbHombre);
```

JComboBox

Menús desplegables: 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.

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.

Flow Layout

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

```
FlowLayout fl_contentPane = new FlowLayout(FlowLayout.LEFT, 10, 10);
contentPane.setLayout(fl_contentPane);
```

Grid 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.

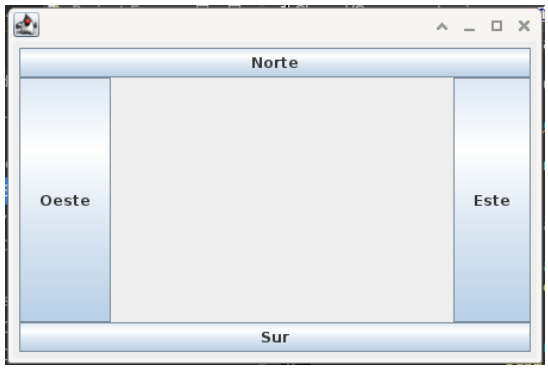
Para la creación de este sistema de rejilla, se utiliza un **constructor** que necesita recibir el valor exacto de **filas** y **columnas** que tendría la interfaz, **GridLayout(int numFilas, int numCol)**.

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.

```
contentPane.setLayout(new GridLayout(2,2));
		lblNewLabel = new JLabel("New label");
		contentPane.add(lblNewLabel);
		lblNewLabel_1 = new JLabel("New label");
		contentPane.add(lblNewLabel_1);
		lblNewLabel_2 = new JLabel("New label");
		contentPane.add(lblNewLabel_2);
		lblNewLabel_3 = new JLabel("New label");
		contentPane.add(lblNewLabel_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**.



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

```
setContentPane(contentPane);
contentPane.setLayout(new BorderLayout(0, 0));
JButton btnNewButton = new JButton("Norte");
contentPane.add(btnNewButton, BorderLayout.NORTH);
JButton btnNewButton_1 = new JButton("Oeste");
contentPane.add(btnNewButton_1, BorderLayout.WEST);
JButton btnNewButton_2 = new JButton("Este");
contentPane.add(btnNewButton_2, BorderLayout.EAST);
JButton btnNewButton_3 = new JButton("Sur");
contentPane.add(btnNewButton_3, BorderLayout.SOUTH);
```

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.