

Programación Orientada a Objetos

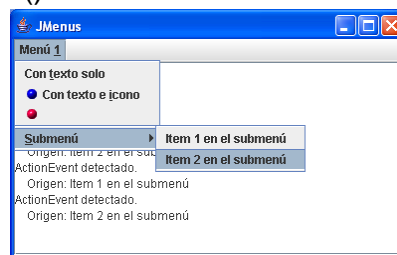
Tema 6: Desarrollo de interfaces gráficas de usuario

Tema 6-2: Conceptos avanzados de SWING

- Tema 6-2: Conceptos avanzados de SWING
- 1. COMPONENTES AVANZADOS
- 2. CUADROS DE DIÁLOGO
- 3. APARIENCIA
- 4. EJEMPLO POO + GUI

- **Menús:**

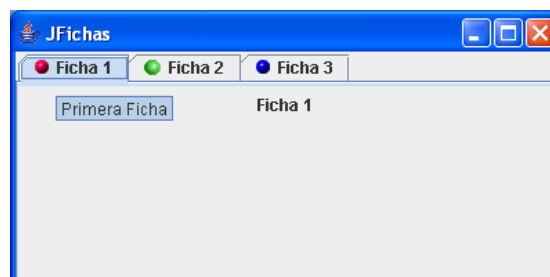
- Los Menús generalmente nos proporcionan el medio principal de manejo de la aplicación.
- Las clases que intervienen en la construcción y manipulación de menús son las siguientes:
 - **JMenuBar**, una barra de menú en una ventana.
 - **JMenu**, es un componente de una barra de menú.
 - **JMenuItem**, representa una opción en un menú. Cuando se selecciona una opción de un menú, se genera un evento de tipo *ActionEvent* y se trata igual que los botones.
- Podemos añadir iconos a las opciones del menú.
- Podemos insertar separadores entre las distintas opciones del menú mediante el método `addSeparator()`.



- **Fichas:**

- Para realizar fichas con Swing utilizaremos la clase **JTabbedPane**. En cada una de las fichas insertaremos un **JPanel** que habremos diseñado previamente.

```
JTabbedPane tabbedPane = new JTabbedPane();
JPanel panel1 = new JPanel();
tabbedPane.addTab("Ficha 1", icono1, panel1, "Primera Ficha");
tabbedPane.setSelectedIndex(0);
JPanel panel2 = new JPanel();
tabbedPane.addTab("Ficha 2", icono2, panel2, "Segunda Ficha");
JPanel panel3 = new JPanel();
tabbedPane.addTab("Ficha 3", icono3, panel3, "Tercera Ficha");
```

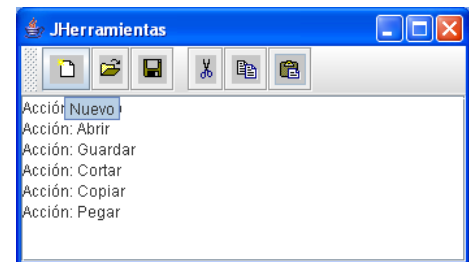




• Barra de Herramientas:

- Las barras de herramientas se crean mediante la clase **JToolBar**. Estas barras tienen la característica especial de ser movibles y acoplables dentro de la interfaz gráfica de usuario.
- Las barras de herramientas son contenedores de otros componentes Swing o de AWT. Generalmente se suelen formar mediante objetos JButton que se construyen a base de iconos representativos de la acción a realizar.
- Al igual que con los menús podemos utilizar el método `addSeparator()` para agregar separadores a la barra de herramientas.

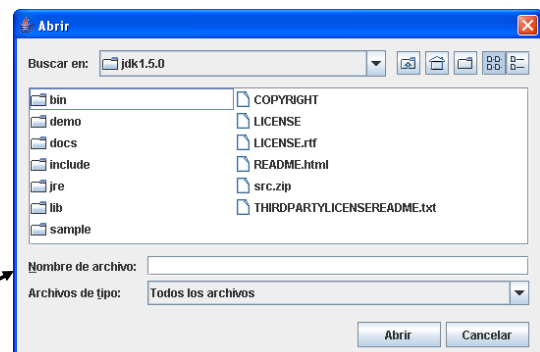
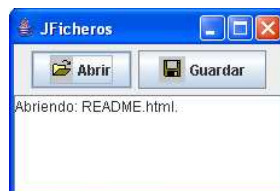
```
JToolBar toolBar = new JToolBar();
JButton boton1 = new JButton(new ImageIcon("new.gif"));
toolBar.add(boton1);
JButton boton2 = new JButton(new ImageIcon("open.gif"));
toolBar.add(boton2);
```



• Selector de ficheros:

- Con la clase **JFileChooser** se permite seleccionar un fichero para ser abierto o guardado, así como la posibilidad de introducir el nombre del fichero. Luego será responsabilidad del programa el llevar a cabo la apertura del fichero o la grabación de datos.
- La ventana que permite la selección de ficheros, suele ser una ventana modal, ya que los cambios que se produzcan en ella, o la selección que se haga, repercutirá en el funcionamiento de la aplicación general.
- Mediante el método `showOpenDialog()` podremos realizar la apertura de un fichero y con `showSaveDialog()` podremos mostrar un diálogo para guardar ficheros.

```
JFileChooser fc = new JFileChooser();
int returnVal = fc.showOpenDialog(this);
int returnVal = fc.showSaveDialog(this);
if (returnVal == JFileChooser.APPROVE_OPTION) {}
```





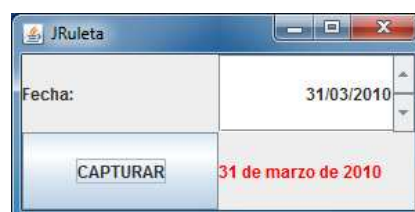
- **Campos de texto con formato:**

- Utilizando la clase **JFormattedTextField** se permite especificar un conjunto válido de caracteres que se pueden escribir en un campo de texto. Esta clase hereda de JTextField.
- Se pueden utilizar para formatear números y fechas o para establecer una máscara (estilo #####-UUU) que debe seguir el texto introducido.
- Se puede acompañar de un validador de entradas creando una subclase de **InputVerifier**.



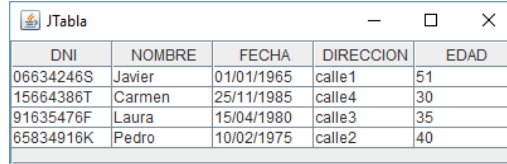
- **Spinners:**

- La clase **JSpinner** se utiliza cuando un campo de texto muestra unos valores con un orden predeterminado. Son similares a las listas y las listas desplegadas ya que permiten al usuario elegir entre un rango de valores pero la diferencia está en que JSpinner no muestra todos los valores posibles sino que el usuario incrementará o decrementará el valor mostrado.
- Existen tres modelos preconfigurados que nos ayudan a controlar un objeto JSpinner:
 - **SpinnerListModel**: Los valores que presenta están almacenados en un array de objetos o en una lista.
 - **SpinnerNumberModel**: Representa secuencias de objetos de tipo numérico.
 - **SpinnerDateModel**: Representa secuencias de objetos de tipo Date.



- **Tablas:**

- Se basan en la clase **JTable** que nos ofrece una forma muy flexible de crear y mostrar tablas.
- Se pueden construir a partir de arrays o vectores de objetos. También se permite la edición de las tablas.



DNI	NOMBRE	FECHA	DIRECCION	EDAD
06634246S	Javier	01/01/1965	calle1	51
15664386T	Carmen	25/11/1985	calle4	30
91635476F	Laura	15/04/1980	calle3	35
65834916K	Pedro	10/02/1975	calle2	40

- **Bordes:**

- Nos permiten rodear una serie de componentes para poder agruparlos. La clase **AbstractBorder** implementa la interfaz **Border** y a partir de ella se crea la jerarquía de clases que representan los distintos bordes que son los que generalmente utiliza el programador.

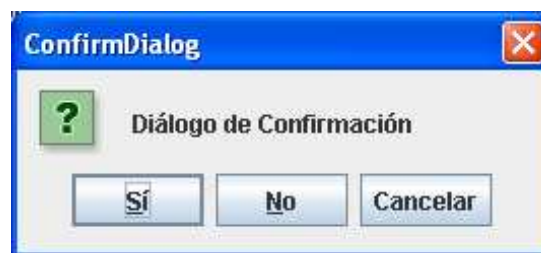


- Los diálogos son aquellos que suelen servir para presentarle al usuario cierta información como por ejemplo un mensaje de advertencia. En Swing se pueden diseñar con la clase **JDialog** o de una forma más sencilla con la clase **JOptionPane**.
- La clase **JOptionPane** nos proporciona cuatro métodos para construir distintos tipos de diálogos, estos métodos son:
 - **showConfirmDialog()**
 - **showInputDialog()**
 - **showMessageDialog()**
 - **showOptionDialog()**

- **showConfirmDialog()**

- Muestra un cuadro de diálogo que hace una pregunta y tiene botones de respuesta Si, No y Cancelar.

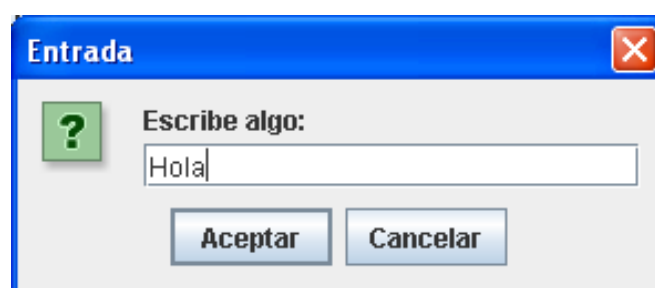
```
int n = JOptionPane.showConfirmDialog(  
    this, "Diálogo de Confirmación",  
    "ConfirmDialog",  
    JOptionPane.YES_NO_CANCEL_OPTION);  
if (n == JOptionPane.YES_OPTION){ textol.setText("SI"); }  
else if (n == JOptionPane.NO_OPTION){ textol.setText("NO"); }  
else if (n == JOptionPane.CANCEL_OPTION) { textol.setText("CANCEL"); }  
else {textol.setText("Ninguna Opción"); }
```



- **showInputDialog()**

- Muestra un cuadro de diálogo que le pide cierta información al usuario mediante un cuadro de texto.

```
String respuesta = JOptionPane.showInputDialog(this, "Escribe algo:");
```



- **showMessageDialog()**

- Muestra un cuadro de diálogo que despliega un mensaje.

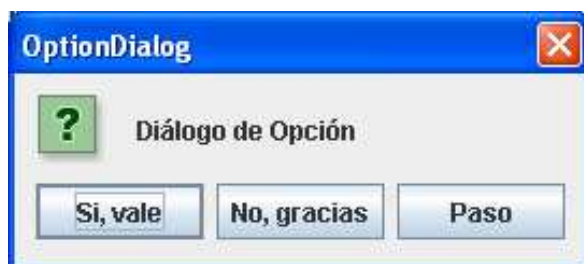
```
JOptionPane.showMessageDialog(this, "Diálogo de  
Mensaje", "MessageDialog", JOptionPane.WARNING_MESSAGE);
```



- **showOptionDialog()**

- Muestra un cuadro de diálogo que combina las características de los tres anteriores.

```
Object[] opciones = {"Si, vale", "No, gracias", "Paso"};  
n = JOptionPane.showOptionDialog(this,  
    "Diálogo de Opción",  
    "OptionDialog",  
    JOptionPane.YES_NO_CANCEL_OPTION,  
    JOptionPane.QUESTION_MESSAGE,  
    null,  
    opciones,  
    opciones[0]);
```





- Cada aplicación Java tiene un objeto **UIManager** que determina el Look-and-Feel, es decir, la apariencia y sensación en pantalla que van a tener los componentes Swing de esa aplicación.
- Java nos permite seleccionar la apariencia de una aplicación asociada a una plataforma, de este modo los programas que se ejecuten en Windows podrán poner esa apariencia y los que se ejecuten en Unix podrán tener apariencia Motif.
- Pero Swing también permite la selección de una apariencia gráfica independiente de la plataforma en que se esté ejecutando la aplicación. La apariencia por defecto de los componentes Swing se denomina Metal, y es propia de Java.
- Para establecer una apariencia se utiliza el método `setLookAndFeel()`.



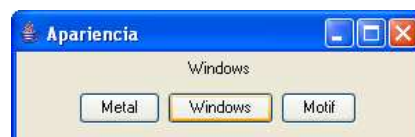
- **Metal:**

```
UIManager.setLookAndFeel("javax.swing.plaf.metal.MetalLookAndFeel");
```



- **Windows:**

```
UIManager.setLookAndFeel("com.sun.java.swing.plaf.windows.WindowsLookAndFeel");
```



- **Motif:**

```
UIManager.setLookAndFeel("com.sun.java.swing.plaf.motif.MotifLookAndFeel");
```





- Aplicación Cobro de Peajes Swing:

The image displays four Swing windows for a toll application:

- Peaje Autopista:** A window titled "Peaje Autopista" with buttons for "Camión" and "Autobús". It shows a summary: "- Total recaudado: 185 - Vehículos: 2" and a "Recaudar" button.
- Peaje Camión:** A window titled "Peaje Camión" with "Peaje" and "Borrar" buttons. It contains input fields for "Matricula:" (1234-ABC), "Peso:" (20000), and "Ejes:" (3).
- Peaje Autobús:** A window titled "Peaje Autobús" with "Peaje" and "Borrar" buttons. It contains input fields for "Matricula:" (2345-CDE), "Peso:" (20000), and "Pasajeros:" (30).
- Summary Windows:** Two windows at the bottom showing the calculated toll. The "Peaje Camión" summary shows: "Vehiculo con matricula: 1234-ABC Peso total: 20000 # Camión - Ejes: 3 - Peaje: 55". The "Peaje Autobús" summary shows: "Vehiculo con matricula: 2345-CDE Peso total: 20000 # Autobús - Pasajeros: 30 - Peaje: 130". Both summary windows have an "Aceptar" button.