Tema 7: Interfaces gráficas de usuario I

Componentes gráficos

1. Introducción

Bibliotecas usadas para crear interfaces gráficas de usuario en Java:

- AWT (Abstract Window Toolkit)
- Swing (más moderna y potente)

Contenedores principales para nuestros componentes:

- JWindow (rectángulo plano sin controles de ventana ni título)
- JFrame (ventana típica)

1.1 Creación de un marco de tamaño fijo

Instrucciones principales:

- Hacemos que nuestra clase herede de JFrame.

```
public class PrimeraVentana extends JFrame
```

- Lo primero que tenemos que hacer en el constructor es llamar a al constructor de la superclase (podemos llamarlo sin parámetros o pasarle el título de la ventana como parámetro).

```
super("Ventana Java");
```

- Establecemos el tamaño de la ventana en 350x200 píxeles.

```
setSize(350, 200);
```

- Establecemos que la operación por defecto al pulsar el botón de cierre de la ventana sea cerrar la aplicación.

```
setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

- Hacemos el JFrame visible (por defecto está oculto).

```
setVisible(true);
```

- Creamos un objeto de tipo PrimeraVentana dentro del método main( ).

```
PrimeraVentana ventana = new PrimeraVentana();
```

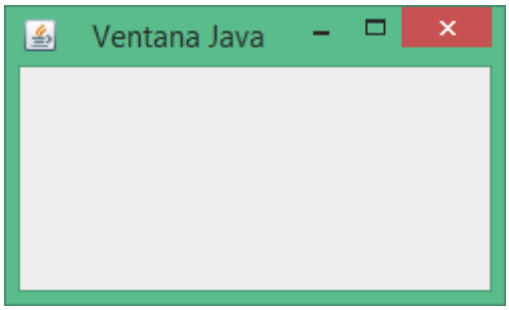
Ejemplo:

```
import javax.swing.*;

public class PrimeraVentana extends JFrame {
	public PrimeraVentana() {
		super("Ventana Java");
		setSize(350, 200);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setVisible(true);
	}

	public static void main(String[] arguments) {
		PrimeraVentana ventana = new PrimeraVentana();
	}

}
```



1.2 Añadir botones y un layout

Podemos añadir botones y cualquier otro componente al contenedor JFrame creándolos como objetos de clase JButton y usando el método add del JFrame:

```
	JButton btnPlay = new JButton("Play");
	add(btnPlay);
```

Hemos de configurar un layout, que es un modo de colocarse los componentes dentro de la ventana. En este caso usamos el más sencillo (Flowlayout) que hace que se coloquen formando una fila horizontal, y si no caben, empiecen a colocarse en la fila de abajo.

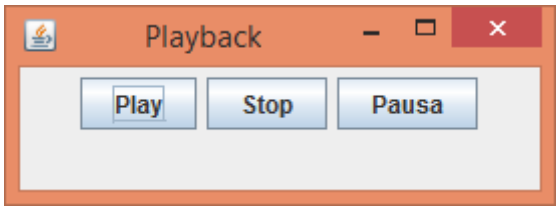
```
	FlowLayout dis = new FlowLayout();
	setLayout(dis);
```

Ejemplo:

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

public class Playback extends JFrame {
	public Playback() {
		super("Playback");
		setSize(275, 100);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setVisible(true);
		FlowLayout dis = new FlowLayout();
		setLayout(dis);
		JButton btnPlay = new JButton("Play");
		JButton btnStop = new JButton("Stop");
		JButton btnPausa = new JButton("Pausa");
		add(btnPlay);
		add(btnStop);
		add(btnPausa);
		setVisible(true);
	}

	public static void main(String[] arguments) {
		Playback pb = new Playback();
	}
}
```



1.3 Etiquetas y campos de texto

Crear una etiqueta:

- `JLabel lblPagina = new JLabel("Dirección página web: ", JLabel.LEFT);`

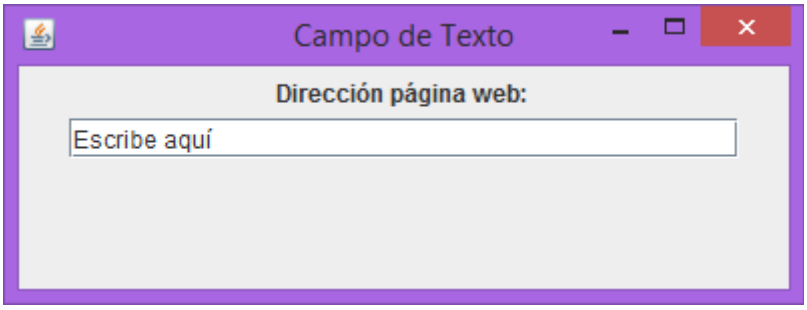
Crear un campo de texto:

- `JTextField txtPagina = new JTextField("Escribe aquí", 30);`

Ejemplo:

```
public class CampoTexto extends JFrame {
```

```
	public CampoTexto() {
		super("Campo de Texto");
		setSize(400, 150);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JLabel lblPagina = new JLabel("Dirección página web: ",
				JLabel.LEFT);
		JTextField txtPagina = new JTextField("Escribe aquí", 30);
		FlowLayout flo = new FlowLayout();
		setLayout(flo);
		add(lblPagina);
		add(txtPagina);
		setVisible(true);
	}
```



1.4 Checkbox y agrupar controles

Crear un checkbox:

- `JCheckBox chkCursoJava = new JCheckBox("Curso de Java", true);`

Los grupos de botones nos sirven para agrupar casillas de texto (JCheckBox) o botones de radio (JRadioButton). De manera que sólo pueda haber uno pulsado por grupo.

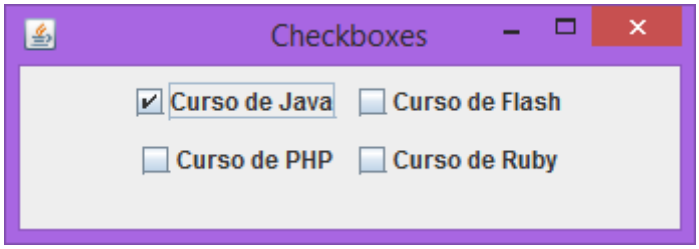
Crear un grupo de botones y añadirle componentes:

- `ButtonGroup cursos = new ButtonGroup();`
- `cursos.add(chkCursoJava);`

Ejemplo:

```
public class CajasChequeo extends JFrame {
	public CajasChequeo() {
		super("Checkboxes");
		setSize(345, 120);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JCheckBox chkCursoJava = new JCheckBox("Curso de Java", true);
		JCheckBox chkCursoFlash = new JCheckBox("Curso de Flash", true);
		JCheckBox chkCursoPHP = new JCheckBox("Curso de PHP", false);
		JCheckBox chkCursoRuby = new JCheckBox("Curso de Ruby", false);
```

```
        FlowLayout dis = new FlowLayout();
        ButtonGroup cursos = new ButtonGroup();
        cursos.add(chkCursoJava);
        cursos.add(chkCursoFlash);
        cursos.add(chkCursoPHP);
        cursos.add(chkCursoRuby);
        setLayout(dis);
        add(chkCursoJava);
        add(chkCursoFlash);
        add(chkCursoPHP);
        add(chkCursoRuby);
        setVisible(true);
    }
```



1.5 Listas desplegables

Un combobox es una lista desplegable. Para crearlo:

- JComboBox<String> cmbCursos = new JComboBox<String>()

Y para añadirle elementos (cadenas de texto, en este caso):

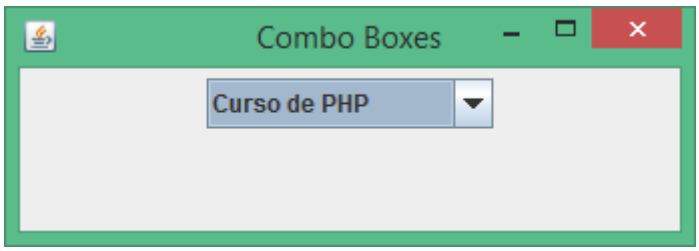
- cmbCursos.addItem("Curso de Java");

Ejemplo:

```
public class ComboBoxes extends JFrame {
    public ComboBoxes() {
        super("Combo Boxes");
        setSize(345, 120);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JComboBox<String> cmbCursos = new JComboBox<String>();
        FlowLayout dis = new FlowLayout();
        cmbCursos.addItem("Curso de Java");
        cmbCursos.addItem("Curso de C++");
        cmbCursos.addItem("Curso de PHP");
        cmbCursos.addItem("Curso de Flash");
        cmbCursos.addItem("Curso de JavaScript");
        cmbCursos.addItem("Curso de Java");
        setLayout(dis);
        add(cmbCursos);
```

```
		setVisible(true);
	}

	public static void main(String[] arguments) {
		ComboBoxes apli = new ComboBoxes();
	}
}
```



1.6 Cambiar el tamaño de un componente

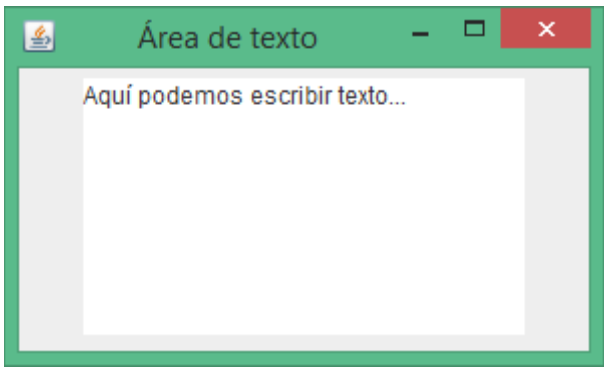
Si queremos cambiar el tamaño de uno de los controles que vamos a añadir a un JFrame, podemos usar su método *setPreferredSize()*. A este método hay que pasarle un objeto de tipo *Dimension*, que sirve para especificar unas medidas en píxeles.

Por ejemplo, si queremos hacer un poco más grande el combobox anterior (250x20 px), podemos escribir:

- cmbCursos.setPreferredSize(new Dimension(250, 20));

1.7 Área de texto

Un área de texto permite crear un casilla formada por varias filas y columnas para introducir textos largos.



Para crearla le indicamos nº de filas y columnas:

- JTextArea txtComentarios = new JTextArea(8, 40);

Ejemplo:

```
public class AreaTexto extends JFrame {
	public AreaTexto() {
		super("AreaTexto");
		setSize(500, 180);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		JTextArea txtComentarios = new JTextArea(8, 40);
		FlowLayout dis = new FlowLayout();
		setLayout(dis);
		add(txtComentarios);
		setVisible(true);
	}

	public static void main(String[] arguments) {
		AreaTexto app = new AreaTexto();
	}
}
```

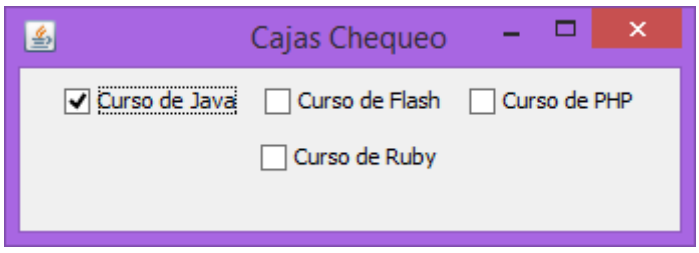
2. Look and feel

2.1 Aspecto visual de las ventanas

Para trabajar con el aspecto visual de nuestras ventanas, usamos la clase UIManager. Para establecer un tema concreto, usamos el método *UIManager.setLookAndFeel()*.

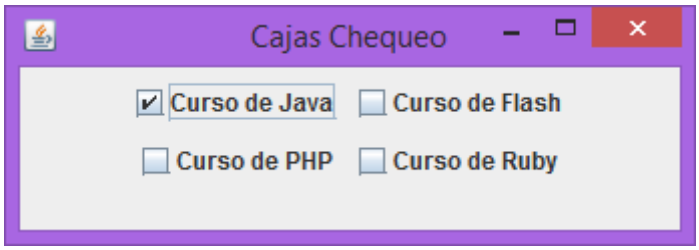
Para establecer el tema por defecto del sistema operativo utilizamos:

- `UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());`



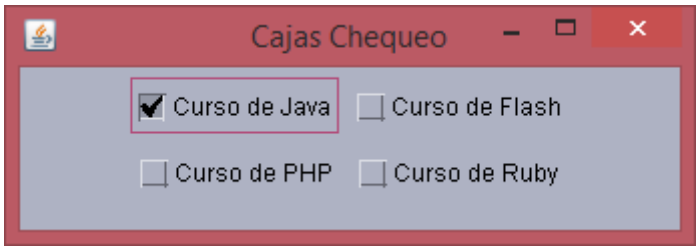
Para establecer el tema por defecto de Java (igual para todas las plataformas):

- `UIManager.setLookAndFeel(UIManager.getCrossPlatformLookAndFeelClassName());`



Para establecer otros temas podemos pasar al método *setLookAndFeel()* una cadena con el nombre del tema. Por ejemplo *Motif*:

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



O *Nimbus*:

- `UIManager.setLookAndFeel("javax.swing.plaf.nimbus.NimbusLookAndFeel");`

2.2 Actualizar el árbol de componentes

Una vez establecido el nuevo *look and feel*, para que todos los componentes de nuestra ventana lo apliquen, hay que llamar al siguiente método:

- `SwingUtilities.updateComponentTreeUI(this);`

En general todo este código vendría en el constructor del JFrame después de la llamada al constructor de la superclase, *super( )*:

```
try {
	UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
	SwingUtilities.updateComponentTreeUI(this);
} catch (Exception e) {
	System.out.println("No se ha podido configurar el look and feel:" +
		e.getMessage());
}
```

2.3 Aspectos visuales disponibles en nuestro sistema

Podemos ver una lista de los temas (look and feels) disponibles en nuestro sistema con el siguiente código. En él se crea un array de objetos de tipo *UIManager.LookAndFeelInfo* que contendrán cada uno información de un tema:

```
public class DetectLook extends JFrame {
	public DetectLook() {
		UIManager.LookAndFeelInfo[] laf = UIManager.getInstalledLookAndFeels();
		for (int i = 0; i < laf.length; i++) {
			System.out.println("Class name: " + laf[i].getClassName());
			System.out.println("Name: " + laf[i].getName() + "\n");
		}
	}

	public static void main(String[] arguments) {
		DetectLook dialog = new DetectLook();
	}
}
```

3. Cuadros de diálogo

Los cuadros de diálogo permiten mostrar mensajes al usuario sin tener que crear nosotros un JFrame a propósito. Todos se generan llamando a un método estático *show()* de la clase JOptionPane.

Por ejemplo:

- JOptionPane.showMessageDialog(): Muestra un mensaje al usuario y, debajo, un botón *Aceptar* que cierra el panel.

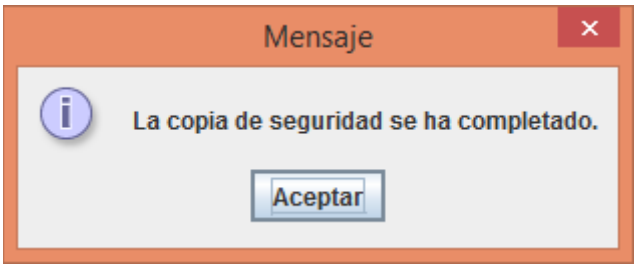
El primer parámetro de los métodos *show()* es el objeto dentro del cual debe mostrarse el panel. En general lo dejaremos en *null* para que nuestro mensaje se muestre centrado en la pantalla.

3.1 MessageDialog

Se trata de una ventana que simplemente nos muestra un mensaje de confirmación.

Ejemplos:

- ```
  JOptionPane.showMessageDialog(null,
                      "La copia de seguridad se ha completado.");
  ```
  ```



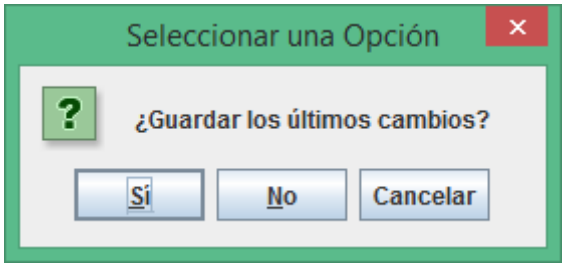
- `JOptionPane.showMessageDialog(null,`
  `"No se ha podido guardar el archivo XML.", "Error de E/S",`
  `JOptionPane.ERROR_MESSAGE);`

3.2 ConfirmDialog

Se crea al llamar al método JOptionPane.showConfirmDialog( ).

Para un diálogo simple, pasamos como argumentos el objeto contenedor del cuadro (o null) y el mensaje que queremos que se muestre. Aparecerán los botones *Sí*, *No*, *Cancelar*.

- `JOptionPane.showConfirmDialog(null, "¿Guardar los últimos cambios?");`



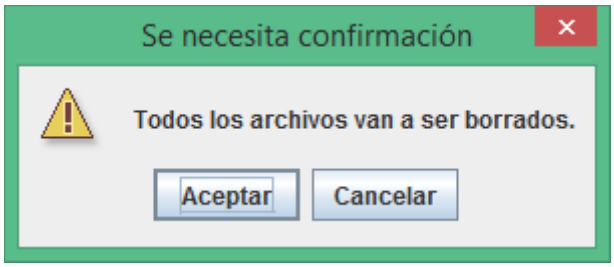
Para un diálogo más complejo podemos pasar estos parámetros:

1. Objeto contenedor
2. Mensaje para el usuario
3. Título de la ventana
4. Botones de la ventana
   a) JOptionPane.YES_NO_CANCEL_OPTION
   b) JOptionPane.YES_NO_OPTION
   c) JOptionPane.YES_OPTION
   d) Etc.
5. Icono (indica el tipo de mensaje)
   a) JOptionPane.WARNING_MESSAGE

b) JOptionPane.ERROR_MESSAGE

c) Etc.

- ```
  JOptionPane.showConfirmDialog(null,
                          "Todos los archivos van a ser borrados.",
                          "Se necesita confirmación", JOptionPane.OK_CANCEL_OPTION,
                          JOptionPane.WARNING_MESSAGE);
  ```

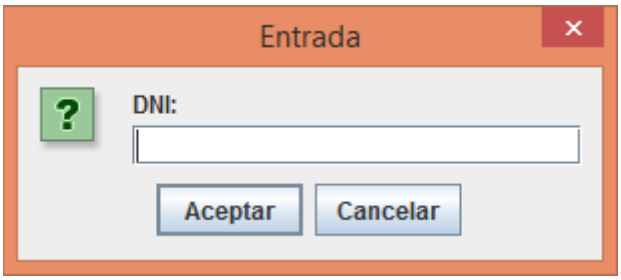


3.3 InputDialog

Muestra una casilla donde podemos escribir información.

Para un diálogo simple pasamos como argumentos el objeto contenedor y un String con el mensaje a mostrar.

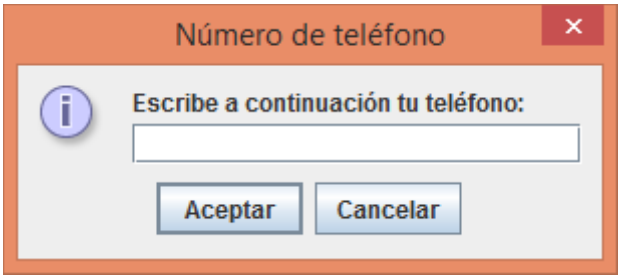
- ```
  String respuesta = JOptionPane.showInputDialog(null, "DNI:");
  ```

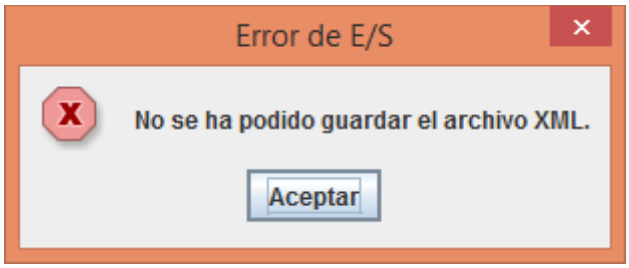


Para un diálogo más complejo podemos pasar los siguientes parámetros:

1. Objeto contenedor.
2. Mensaje a mostrar.
3. Título de la ventana.
4. Icono
   a) JOptionPane.QUESTION_MESSAGE
   b) Etc.

- ```
  String respuesta = JOptionPane.showInputDialog(null,
                     "Escribe a continuación tu teléfono:", "Número de teléfono",
                     JOptionPane.INFORMATION_MESSAGE);
  ```





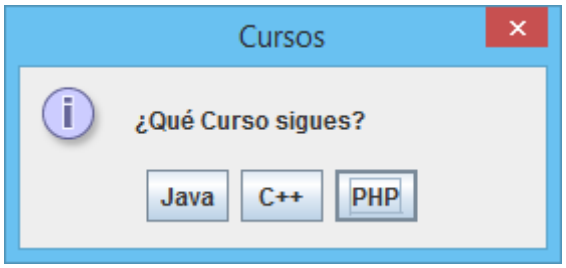
3.4. OptionDialog

Este control nos permite personalizar completamente los botones que aparecerán en el cuadro. Para ello hay que crear un array de Strings con los mensajes de cada botón y pasárselo como argumento.

Lista de argumentos:

1. Objeto contenedor (*null* si queremos que el panel se centre respecto a la pantalla)
2. Mensaje
3. Título de la ventana
4. Botones (se pone a cero para personalizar)
5. Icono (error, advertencia, información)
6. Icono personalizado (poner null)
7. Array de Strings
8. String con el mensaje por defecto (el que aparece resaltado).

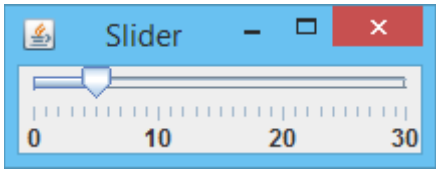
- ```
  String[] cursos = { "Java", "C++", "PHP" };
            int response = JOptionPane.showOptionDialog(null,
            "¿Qué Curso sigues?", "Cursos", 0,
            JOptionPane.INFORMATION_MESSAGE, null, cursos, cursos[2]);
  ```



4. Componentes avanzados de Swing

4.1 JSlider

Muestra una escala sobre la que podemos deslizar un indicador. La versión más larga del constructor nos permite indicar:



- Orientación (JSlider.Horizontal, JSlider.Vertical).
- Valor mínimo.
- Valor máximo.
- Valor inicial.

Además, podemos modificar algunas propiedades con los siguientes métodos:

- *setMajorTickSpacing(10)*: Las marcas grandes irán de 10 en 10.
- *setMinorTickSpacing(1)*: Las marcas pequeñas irán de 1 en 1.
- *setPaintTicks(true)*: Se mostrarán las marcas.
- *setPaintLabels(true)*: Se mostrarán los números.

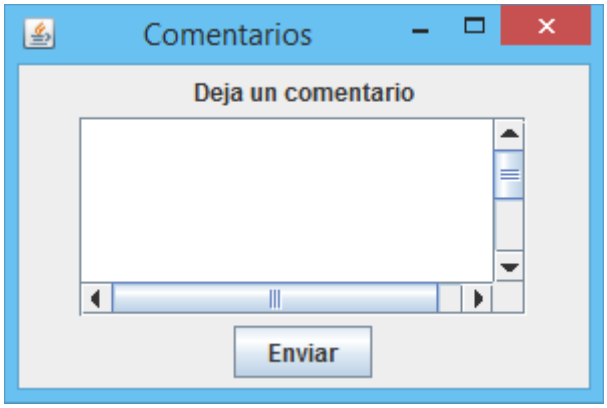
Ejemplo:

```
public Sliders() {
        super("Ejemplo Slider");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        JSlider miSlider = new JSlider(JSlider.HORIZONTAL, 0, 30, 5);
        miSlider.setMajorTickSpacing(10);
        miSlider.setMinorTickSpacing(1);
        miSlider.setPaintTicks(true);
```

```
        miSlider.setPaintLabels(true);
        add(miSlider);
        pack();
        setVisible(true);
}
```

4.2 JscrollPane (barras de desplazamiento)

Permite agregar barras de scroll a otro componente.



Para conseguir esto seguimos los siguientes pasos:

1. Crear el componente al que le queremos agregar barras laterales (un JTextArea, por ejemplo).

   ```
   JTextArea txtComentarios = new JTextArea(6, 20);
   ```

2. Crear un objeto JScrollPane y pasarle al constructor el objeto JTextArea.

   ```
   JScrollPane scrPanelBarras = new JScrollPane(txtComentarios);
   ```

3. Añadir al panel dentro del JFrame el objeto JScrollPane, con la orden:

   ```
   getContentPane().add(scrPanelBarras);
   ```

Ejemplo:

```
public AreaTextoScroll() {
       super("AreaTextoScroll");
       setSize(300, 150);
       setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
       JTextArea txtComentarios = new JTextArea(6, 20);
       FlowLayout dis = new FlowLayout();
       setLayout(dis);
       scrPanelBarras = new JScrollPane(txtComentarios);
       getContentPane().add(scrPanelBarras);
       setVisible(true);
}
```

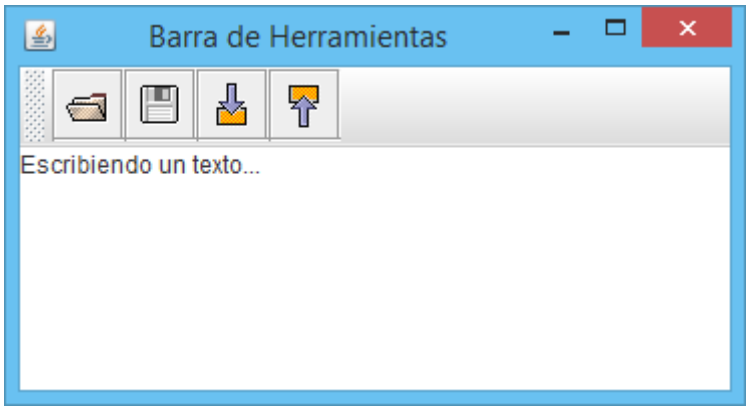
4.3 ImageIcon

Nos permite poner imágenes en etiquetas o botones. Se crea así:

```
ImageIcon imgCargar = new ImageIcon("Ruta al archivo GIF, PNG o JPG");
```

4.4 Barras de herramientas (JToolBar)

Una barra de herramientas se crea con la clase JToolBar.



Cada botón de la barra se forma con ImageIcon (que recibe un archivo JPG, PNG o GIF) y un JButton:

```
ImageIcon imgCargar = new ImageIcon("iconos/load.gif");
JButton btnCargar = new JButton(imgCargar);
```

Después se crea el JToolBar y se le añaden todos los botones:

```
JToolBar barra = new JToolBar();
barra.add(btnCargar);
```

En una ventana con barra de herramientas usaremos BorderLayout en lugar de FlowLayout. Así colocaremos la barra en la posición NORTH habitualmente, y el resto de componentes en la posición CENTER:

```
add("North", barra);
add("Center", txtEditar);
```

Ejemplo:

```
public BarraHerramientas() {
        super("Barra de Herramientas");
        setSize(370, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        BorderLayout borde = new BorderLayout();
        setLayout(borde);

        // Construimos cada botón de la barra
        ImageIcon imgCargar = new ImageIcon("iconos/load.gif");
        JButton btnCargar = new JButton(imgCargar);
        ImageIcon imgGuardar = new ImageIcon("iconos/save.gif");
        JButton btnGuardar = new JButton(imgGuardar);
        ImageIcon imgAlta = new ImageIcon("iconos/subscribe.gif");
        JButton btnAlta = new JButton(imgAlta);
        ImageIcon imgBaja = new ImageIcon("iconos/unsubscribe.gif");
        JButton btnBaja = new JButton(imgBaja);

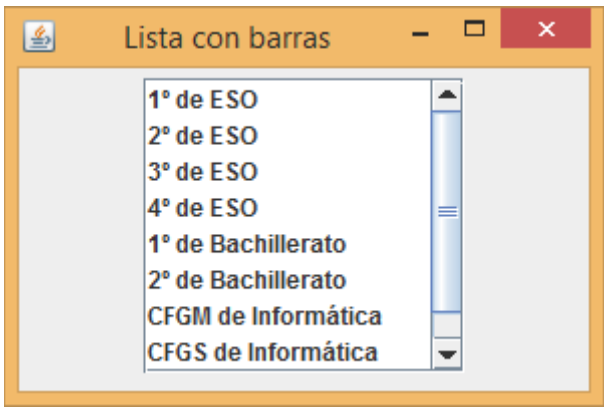
        // Creamos la barra y le añadimos los botones
        JToolBar barra = new JToolBar();
        barra.add(btnCargar);
        barra.add(btnGuardar);
        barra.add(btnAlta);
        barra.add(btnBaja);

        // construir area texto
        JTextArea txtEditar = new JTextArea(3, 10);

        // Agregar la barra de herramientas y el text area al marco.
        add("North", barra);
        add("Center", txtEditar);
        setVisible(true);
}
```

4.5 Listas (JList)

Podemos mostrar una lista de elementos con un control JList. La diferencia con un JComboBox es que la lista permanece siempre desplegada.



Creamos un array de Strings con las palabras que se mostrarán, y lo pasamos al constructor del JList:

```
String categorias[] = { "1º de ESO", "2º de ESO", "3º de ESO"}
JList lista = new JList(categorias);
```

Ejemplo de JList con barras de scroll:

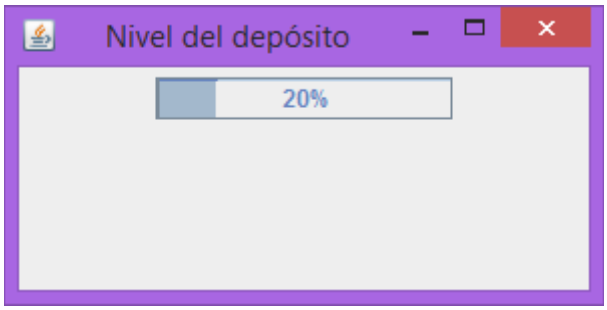
```
public Lista() {
	super("Lista con barras");
	setSize(300, 200);
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setLayout(new FlowLayout());

	String categorias[] = { "1º de ESO", "2º de ESO", "3º de ESO",
			"4º de ESO", "1º de Bachillerato", "2º de Bachillerato",
			"CFGM de Informática", "CFGS de Informática",
			"FP básica de Informática" };

	JList lista = new JList(categorias);
	JScrollPane panelScroll = new JScrollPane(lista);

	add(panelScroll);
	setVisible(true);
}
```

4.6 Barras de progreso (JProgressBar)



Para crear una barra de progreso declaramos un objeto JProgressBar pasándole como argumentos los valores inicial y final de la misma:

```
JProgressBar prgDeposito = new JProgressBar(0, 2000);
```

Para mostrar un valor lo asignamos con la propiedad *setValue*:

```
prgDeposito.setValue(400);
```

Podemos hacer que se muestre el porcentaje recorrido:

```
prgDeposito.setStringPainted(true);
```

Ejemplo:

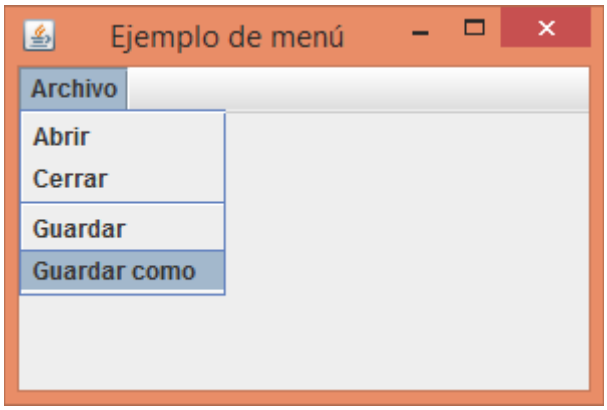
```
public BarraProgreso() {
	super("Nivel del depósito");
	setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
	setSize(300, 150);
	setLayout(new FlowLayout());

	// Creamos una barra de progreso con valores de 0 a 2000.
	// Su valor inicial es 400. Se mostrará también el porcentaje.
	JProgressBar prgDeposito = new JProgressBar(0, 2000);
	prgDeposito.setValue(400);
	prgDeposito.setStringPainted(true);

	add(prgDeposito);

	setVisible(true);
}
```

4.7 Barras de menú (JMenuItem, JMenu, JMenuBar)



Para añadir una barra de menú a una ventana (JFrame) trabajaremos con tres clases:

- JMenuItem: Cada opción que se muestra en un menú desplegable (por ejemplo, Nuevo, Abrir, Guardar, ...).
- JMenu: Cada menú desplegable (por ejemplo, Archivo, Editar, Ver, ...).
- JMenuBar: La propia barra de menú.

```
public Menu() {
	super("Ejemplo de menú");
```

```
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(300, 200);

        setLayout(new FlowLayout());

        // Creamos los elementos del menú Archivo.
        JMenuItem itemAbrir = new JMenuItem("Abrir");
        JMenuItem itemCerrar = new JMenuItem("Cerrar");
        JMenuItem itemGuardar = new JMenuItem("Guardar");
        JMenuItem itemGuardarComo = new JMenuItem("Guardar como");

        // Creamos el menú Archivo y le añadimos sus elementos.
        JMenu menuArchivo = new JMenu("Archivo");
        menuArchivo.add(itemAbrir);
        menuArchivo.add(itemCerrar);
        menuArchivo.addSeparator(); // Línea horizontal para separar
        menuArchivo.add(itemGuardar);
        menuArchivo.add(itemGuardarComo);

        // Creamos la barra de menú y le añadimos el menú Archivo.
        JMenuBar barraMenu = new JMenuBar();
        barraMenu.add(menuArchivo);

        // Establecemos barraMenu como barra de menú de nuestra ventana.
        setJMenuBar(barraMenu);

        setVisible(true);
}
```

Referencias

- Look and feel que funciona bastante bien:
  - http://www.jtattoo.net/