Diseño y Desarrollo de Sistemas de Información

Breves nociones para crear una aplicación sencilla con una interfaz gráfica usando la librería Swing

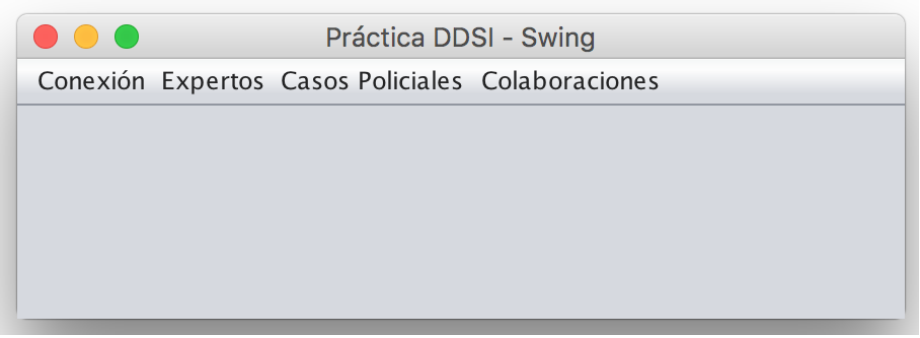
- El objetivo de estas diapositivas es mostrar, de forma breve y concisa, la forma de crear una aplicación en Java con interfaz gráfica usando la librería Swing, que ya viene incorporada en NetBeans

- Por supuesto, existen multitud de librerías para diseñar interfaces (por ejemplo, JavaFX) pero, en este caso, nos centraremos en esta librería por ser suficientemente estable y estar muy extendida

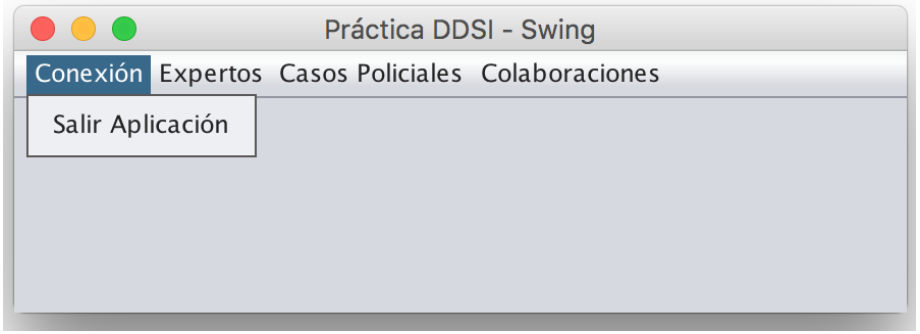
- **Esto no es un manual de Swing**. Sólo son unas nociones muy básicas para poder interactuar, desde Java y mediante una interfaz gráfica, con una base de datos

- Además, lo haremos en modo "diseño", lo que quiere decir que gran parte del código será generado de forma automática

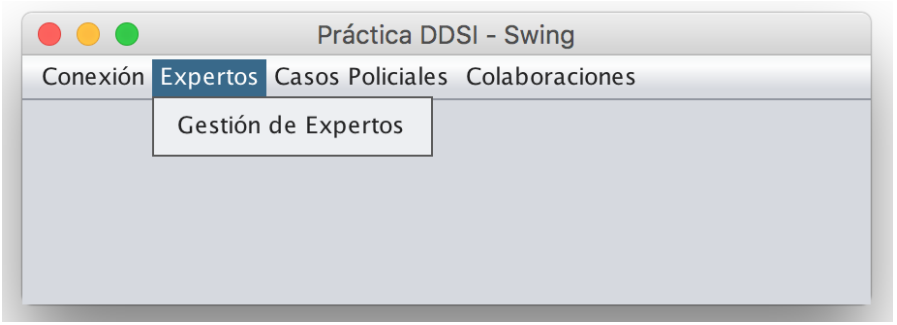
- Diseñaremos una ventana principal con menús parecida a esta



- Dentro del menú "Conexión" tendremos la opción "Salir de la aplicación"



- Y dentro del menú "Expertos" tendremos la opción "Gestión de Expertos"



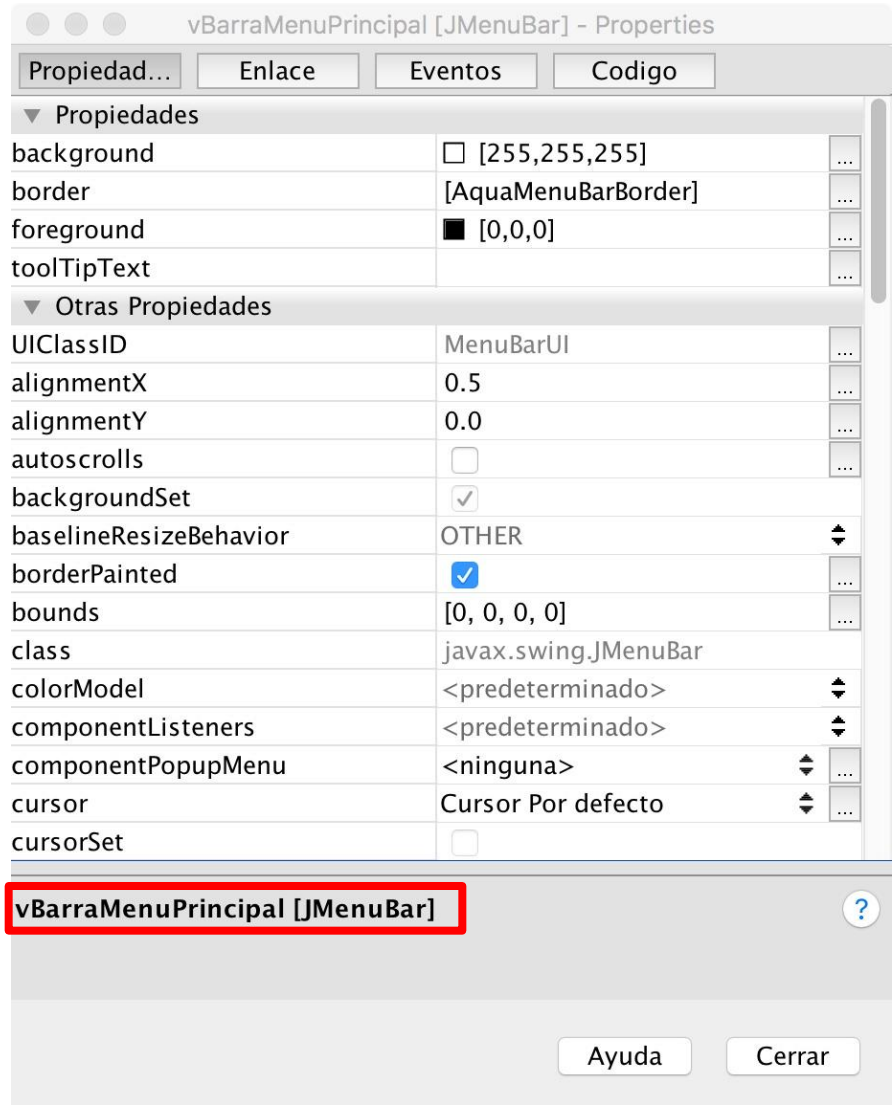
- En la capa de "Aplicación" seleccionamos un nuevo "Formulario JFrame"

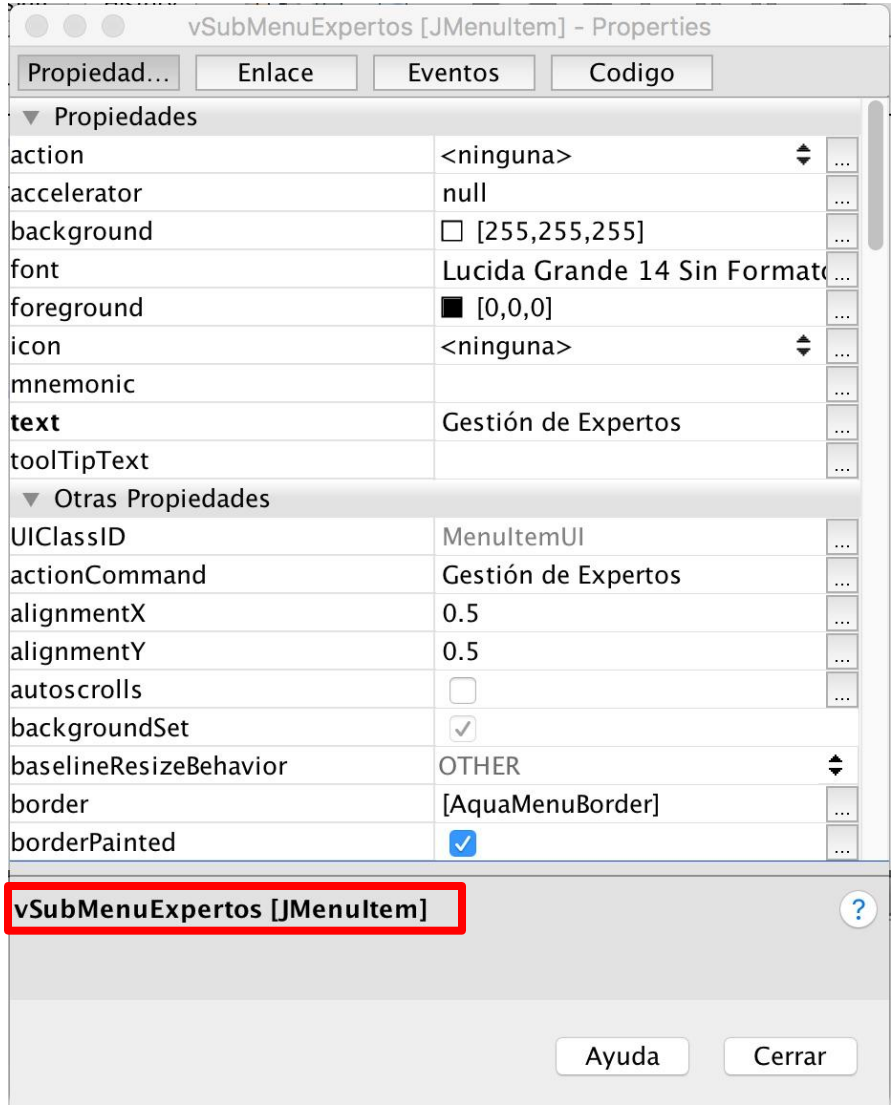
> Las ventanas *Frames* se utilizan normalmente como ventanas independientes de alto nivel, como la interfaz de usuario principal de la aplicación. La mayoría de las aplicaciones Swing se crean a partir de este formulario

- Una vez elegido el nombre del fichero y la ubicación, aparecerá un panel "en blanco" al que podremos ir añadiendo componentes desde la "Paleta de Componentes"

- En nuestro ejemplo, los componentes del menú son:
  - **Barra de menú (JMenuBar)**. Para crear la barra principal
  - **Menú (JMenu)**. Para crear los elementos principales del menú
  - **Elemento de Menú (JmenuItem)**. Para crear las opciones de los elementos principales del menú

- Una vez situados los componentes, podemos definir y modificar todas sus características y propiedades





- Una de las primeras acciones que debemos hacer es asignar al componente un nombre de variable "entendible". Se puede hacer desde el menú de propiedades o desde el menú contextual con el botón derecho

- La principal característica del funcionamiento de una interfaz gráfica es la gestión de eventos

- Esto se hace de forma muy sencilla desde la pestaña "Eventos" de las propiedades del componente al que se le quiere incorporar un evento

- Por ejemplo, si se quiere asignar un evento de "click" de ratón al elemento del menú "Salir", en su cuadro de propiedades seleccionamos **ActionPerformed**. Esto generará un código parecido a este (suponemos que el nombre de la variable que le hemos puesto es **vSubMenuSalir**):

```
 private void vSubMenuSalirActionPerformed(java.awt.event.ActionEvent evt) {

}
```

- En el método generado insertaremos el código que se desea ejecutar cuando se pulse con el ratón. Por ejemplo:

```
private void vSubMenuSalirActionPerformed(java.awt.event.ActionEvent evt) {
    try {
        co.desconexion();
    }
    catch (SQLException ex) {
        JOptionPane.showMessageDialog(null, ex.getMessage());
    }
    finally {
        this.dispose();
    }
}
```

- Un ejemplo del código de la ventana principal:

1/2

```
package Aplicacion;

import Persistencia.conexionOracle;
import java.sql.SQLException;
import javax.swing.JOptionPane;

public class ventanaPrincipal extends javax.swing.JFrame {
    static conexionOracle co = null;

    public ventanaPrincipal() {
        try {
            co = new conexionOracle();
            initComponents();
        } catch (Exception ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
            System.exit(1);
        }
    }

 private void vSubMenuSalirActionPerformed(java.awt.event.ActionEvent evt) {
     try {
         co.desconexion();
     }
     catch (SQLException ex) {
         JOptionPane.showMessageDialog(null, ex.getMessage());
     }
     finally {
         this.dispose();
     }
 }
```

2/2

```
  private void vSubMenuExpertosActionPerformed(java.awt.event.ActionEvent evt) {
        try {
            ventanaExpertos vE = new ventanaExpertos(co);
            vE.setVisible(true);
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
    }

// Aquí viene el código generado automáticamente para inicializar los componentes
// y mostrarlos según el diseño realizado

public static void main(String args[]) {

        /* Set the Nimbus look and feel */

        /* Create and display the form */
        java.awt.EventQueue.invokeLater(new Runnable() {
            public void run() {
                new ventanaPrincipal().setVisible(true);
            }
        });
    }

// Variables declaration - do not modify
    private javax.swing.JMenuBar vBarraMenuPrincipal;
    private javax.swing.JMenu vMenuCasos;
    private javax.swing.JMenu vMenuColaboraciones;
    private javax.swing.JMenu vMenuConexion;
    private javax.swing.JMenu vMenuExpertos;
    private javax.swing.JMenuItem vSubMenuExpertos;
    private javax.swing.JMenuItem vSubMenuSalir;
// End of variables declaration
}
```

Código generado automáticamente para mostrar la ventana principal

- Para la gestión de los expertos vamos a crear otro fichero Java de tipo "Formulario JFrame" al que llamaremos "ventanaExpertos.java"

- Añadiremos los siguientes componentes:
  - **Tabla (JTable)**. Es un contenedor para mostrar el contenido de una tabla o consulta a la base de datos. De forma automática, se crea también un componente de tipo **JScrollPane**, que es donde se sitúa la tabla
  - **Campo de Texto (JTextField)**. Para escribir el nombre del país y realizar el filtro
  - **Etiqueta (JLabel)**. Para describir el contenido del campo de texto
  - **Botones (JButton)**. Lo usaremos para activar el filtro o mostrar todos los expertos

- Posible aspecto de la ventana de Expertos

- Una de las formas más rápidas y sencillas de utilizar un JTable teniendo toda su funcionalidad consiste en instanciar, como modelo de datos, un **DefaultTableModel** y luego un **JTable** , pasándole el modelo en el constructor. Por ejemplo:

```
DefaultTableModel modeloTExpertos = new DefaultTableModel();
JTable tExpertos = new JTable(modeloTExpertos);
```

- O también:

```
DefaultTableModel modeloTExpertos = new DefaultTableModel();
JTable tExpertos = new JTable();
tExpertos.setModel(modeloTExpertos);
```

- A partir de ese momento, todo se maneja con el modelo.
- **DefaultTableModel** tiene todos los métodos necesarios para modificar los datos de la tabla que contiene, añadir filas o columnas, asignarle un nombre a cada columna, etc.

- Un ejemplo de "diseño" para dibujar la tabla EXPERTO

```
private void dibujarTablaExpertos() {
    tExpertos.setModel(modeloTExpertos);
    String[] columnasTabla = {"Código","Nombre","País","Sexo","Especialidad"};
    modeloTExpertos.setColumnIdentifiers(columnasTabla);

    // Para no permitir el redimensionamiento de las columnas con el ratón
    tExpertos.getTableHeader().setResizingAllowed(false);

    // Así se fija el ancho de las columnas
    tExpertos.getColumnModel().getColumn(0).setPreferredWidth(25);
    tExpertos.getColumnModel().getColumn(1).setPreferredWidth(140);
    tExpertos.getColumnModel().getColumn(2).setPreferredWidth(80);
    tExpertos.getColumnModel().getColumn(3).setPreferredWidth(8);
    tExpertos.getColumnModel().getColumn(4).setPreferredWidth(122);
  }
```

Objeto de tipo *JTable*

Objeto de tipo *DefaultTableModel*

- Un ejemplo de cómo mostrar los datos de la tabla EXPERTO a partir de una lista de "expertos"

```
private void rellenarTablaExpertos(ArrayList<experto> expertos) {
        Object[] columna = new Object[5];
        int numRegistros = expertos.size();
        for (int i = 0; i < numRegistros; i++)  {
            columna[0] = expertos.get(i).getCodExperto();
            columna[1] = expertos.get(i).getNombre();
            columna[2] = expertos.get(i).getPais();
            columna[3] = expertos.get(i).getSexo();
            columna[4] = expertos.get(i).getEspecialidad();
            modeloTExpertos.addRow(columna);
        }
}
```

- Un ejemplo para "vaciar" el contenido de la tabla EXPERTO

```
private void vaciarTablaExpertos() {
    while (modeloTExpertos.getRowCount() > 0)
        modeloTExpertos.removeRow(0);
}
```

- Métodos para solicitar los datos y rellenar la tabla EXPERTO

```
private void pideExpertos() throws SQLException {
        ArrayList<experto> lExp = mExp.listaExpertos();
        rellenarTablaExpertos(lExp);
}

private void pideExpertosPorPais() throws SQLException {
        String paisSeleccionado = txtPais.getText();
        ArrayList<experto> lExp = mExp.listaExpertosPorPais(paisSeleccionado);
        rellenarTablaExpertos(lExp);
}
```

Objeto de tipo *manejaExperto*

Objeto de tipo *JTextField*

- Métodos para gestionar los eventos de los botones

```
private void bListarExpertosPaisActionPerformed(java.awt.event.ActionEvent evt) {
        vaciarTablaExpertos();
        try {
            pideExpertosPorPais();
        }
         catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
}


private void bListarExpertosActionPerformed(java.awt.event.ActionEvent evt) {
        vaciarTablaExpertos();
        try {
            pideExpertos();
        }
         catch (SQLException ex) {
            JOptionPane.showMessageDialog(null, ex.getMessage());
        }
}
```

- En resumen, el "esqueleto" del código de la ventana de Expertos podría quedar así

1/2

```
public class ventanaExpertos extends javax.swing.JFrame {
    manejaExperto mExp = null;

    // Se sobreescribe el método isCellEditable para hacer que las filas
    // no se puedan editar al hacer doble click

   DefaultTableModel modeloTExpertos = new DefaultTableModel() {
        @Override
        public boolean isCellEditable(int row, int column) {
                return false;
        }
    };


 public Ventana_Expertos(conexionOracle c) throws SQLException {
        mExp = new manejaExperto(c);
        initComponents();
        dibujarTablaExpertos();
        pideExpertos();
    }

// Aquí viene el código generado automáticamente para inicializar los componentes
// y mostrarlos según el diseño realizado
```

```
private void bListarExpertosPaisActionPerformed(java.awt.event.ActionEvent evt) {
   // Código para gestionar el evento del botón bListarExpertosPais
}

// Resto de métodos


// Variables declaration - do not modify
    private javax.swing.JScrollPane PanelExpertos;
    private javax.swing.JButton bListarExpertos;
    private javax.swing.JButton bListarExpertosPais;
    private javax.swing.JLabel etPais;
    private javax.swing.JTable tExpertos;
    private javax.swing.JTextField txtPais;
// End of variables declaration

}
```

2/2