

# UT2. Diseño de Interfaces

Con Netbeans

José Jarones Bueno

# Interfaces : Introducción



**Una interfaz:** es una capa intermedia que nos permite interactuar con la máquina.



Java ofrece una serie de librerías para realizar interfaces gráficas. Estas librerías son conocidas por las siglas GUI: Graphical User Interface.

Así que podemos definir GUI como: Una interfaz de usuario que permite a los usuarios interactuar con los dispositivos electrónicos a través de iconos e indicadores visuales, en lugar de utilizar interfaces basadas en comandos de texto.

# Interfaces : ¿Opciones?



En Java, existen tres principales implementaciones de GUI:

**AWT:** Utiliza el GUI nativo del sistema que tenga el SO. No funcionará igual en todos los sistemas, aunque se intenta que así sea. Por ejemplo, GUIs desarrolladas en AWT que utilicen componentes muy modernos, puede que no funcionen igual en unos equipos o en otros, o incluso que no funcionen en determinados equipos porque su SO no lo soporta.

**SWING:** Es una interfaz que se monta sobre la Máquina Virtual de Java. Desde el punto de vista del sistema operativo, los botones de una interfaz de swing no son más que píxeles dentro de un programa, no tienen semántica. Utilizar SWING y AWT al mismo tiempo *no está soportado*. Es mucho más portable. **Su paquete principal es `java.swing`**

**JAVAFX.** Una plataforma para crear aplicaciones de escritorio y así como Rich Internet Applications que se ejecutan sobre una gran variedad de dispositivos. La intención es que suplante a Swing como la librería de interfaces estándar de Java, pero de momento, conviven.

# Interfaces : Terminología Widgets y Applets



**Widget:** Una pequeña parte gráfica de nuestro programa que nos permite llevar a cabo una funcionalidad.

En Swing podemos considerar que los Botones, las etiquetas, Diálogos... son componentes. Su agrupación formando funcionalidad, Widgets.

**Applet:** Es un programa en Java que corre dentro de otra aplicación que no es nativa en Java, normalmente desde una página Web.

El navegador tiene que ser capaz de utilizar la máquina virtual de java para poder reproducir la aplicación.

En la actualidad han quedado totalmente en desuso porque se han visto superadas por las tecnologías web (HTML, CSS, Javascript).

# Interfaces : JFrame



**JFrame:** Es la clase principal para pintar una ventana en SWING.

Un JFrame define una ventana que sirve de contenedor para todos los componentes que formarán parte de la interfaz.

## Métodos Básicos de JFrame

Método	Acción
setVisible( boolean)	Muestra o esconde la ventana
setBounds( x, y, ancho, alto)	Determina la posición y la dimensión de la ventana
setDefaultCloseOperation(int operation)	Determina la operación que va a suceder cuando el usuario cierre esta ventana

Más información sobre JFrame [AQUÍ](#)

# Interfaces : JFrame



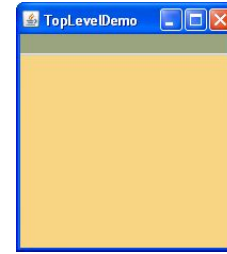
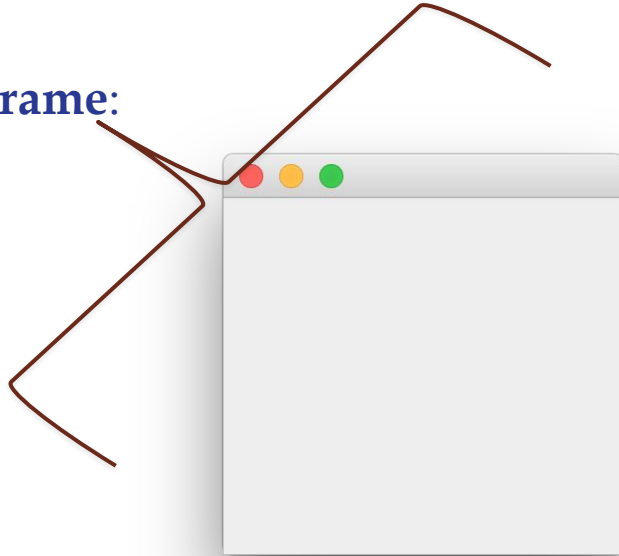
## JFrame:

- Es incompatible con Frame (la clase de AWT).
- Por defecto, los JFrames vienen con un elemento, un JRootPane como único hijo.
  - Un JRootPane, es el panel que contiene todo lo que no es un menú de la ventana.
  - Añadir un componente a un JFrame es lo mismo que añadirlo a un JRootPane.
  - Por defecto, un JRootPane tiene siempre como Layout BorderLayout.
- Un JFrame tiene información sobre qué tiene que pasar cuando se cierra la ventana.

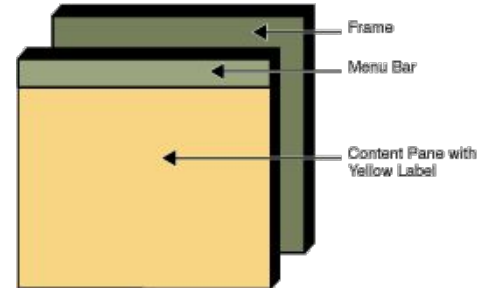
# Interfaces : JFrame



JFrame:



JRootPane



# Interfaces : JFrame



`setBounds(x, y, width, height)`

para especificar la posición y el tamaño de un componente GUI si establece layout en null. Entonces (x, y) es la coordenada de la esquina superior izquierda de ese componente.



# Interfaces : JFrame



## EJERCICIO DE CLASE:

Crea manualmente un programa en Java que inicialice un JFrame.

## PASOS:

1. Crear la clase *Principal* y *VentanaPrincipal*.
  2. *VentanaPrincipal* tendrá como atributo un JFrame.
  3. En el constructor de *VentanaPrincipal* se instanciará el JFrame, se le dotará de unas dimensiones con `setBounds(x, y, width, height)` y se le dotará de un método de salida `setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);`
  1. Se creará un método en *VentanaPrincipal* denominado *inicializar()* que hará que el JFrame sea visible. (true)
  2. Desde el principal nos declararemos una *VentanaPrincipal* y llamaremos a su método *inicializar()*.
- VER SIGUIENTE DIAPOSITIVA**

# Interfaces : JFrame



## EJERCICIO 1:

```
public class Principal {  
    Run | Debug  
    public static void main(String[] args) {  
        EventQueue.invokeLater(new Runnable() {  
            public void run() {  
                try {  
                    VentanaPrincipal ventanaPrincipal = new VentanaPrincipal();  
                    ventanaPrincipal.inicializar();  
                } catch (Exception e) {  
                    e.printStackTrace();  
                }  
            }  
        });  
    }  
}
```

# Componentes



## Componentes SWING más comunes:

Son los elementos que podemos colocar dentro de nuestra interfaz.

Componente	Cómo se implementa
Etiquetas	JLabel
Botones	JButton
Cajas de texto	JTextField -> una línea JTextArea -> varias líneas
Casillas de verificación	JCheckBox
Botones de opción	JRadioButton
Listas	JList
Barras de desplazamiento	JScrollBar
Cuadros de diálogo	JOptionPane

# Contenedores



Se utilizan para ayudarnos a colocar correctamente los componentes. Estos se organizan en una jerarquía de contenedores.

La raíz general de ese contenedor siempre será el contenedor superior definido por la ventana → **JFrame**.

Para acceder al contenedor raíz de una ventana se utiliza el método **getContentPane**. Éste nos devuelve un objeto de la clase **JRootPane**, es un panel intermedio que nos facilita la colocación de componentes.

Un panel puede contener otros paneles para crear una Jerarquía.

Para añadir componentes a un panel se utiliza el método **add(componente)**.

# Contenedores



## EJERCICIO DE CLASE 2:

Utilizando la plantilla del ejercicio 1. Hacer manualmente:

Obtener el panel de la ventana principal (**getContentPane**) y añadir un botón dentro (**add**).

¡El botón hay que definirlo e iniciarlo antes de poder añadirlo!

Cuando termines, prueba a añadir otro botón.

# Contenedores



## EJERCICIO 2:



Intentad que quede algo tal que así.

¡Mirad la documentación y ejemplos!

# Layouts



## PROBLEMA:

Hasta ahora sabemos añadir componentes dentro de la ventana...  
¿Pero cómo los coloco?

Administradores de diseño (layouts)

Están pensados para mostrar varios componentes a la vez en un orden preestablecido. ¿Qué tipos de layout podemos utilizar?

# Layouts



Layout	¿Qué hace?
Absolute (ó nulo)	Posicionamiento libre
BorderLayout	Divide el contenedor en Norte, Sur, Este, Oeste y centro.
GridLayout	Diseño por rejillas
GridBagLayout	Igual pero un componente puede ocupar varias filas o columnas.
FlowLayout	Se colocan de izquierda a derecha, como las palabras en un párrafo
BoxLayout	Parecido a FlowLayout. Más control sobre los elementos. Horizontal y vertical. <a href="#"><u>INFO</u></a>



# Layouts



## ABSOLUTE LAYOUT (Layout nulo):



Crear un contenedor con layout nulo involucra los siguientes pasos:

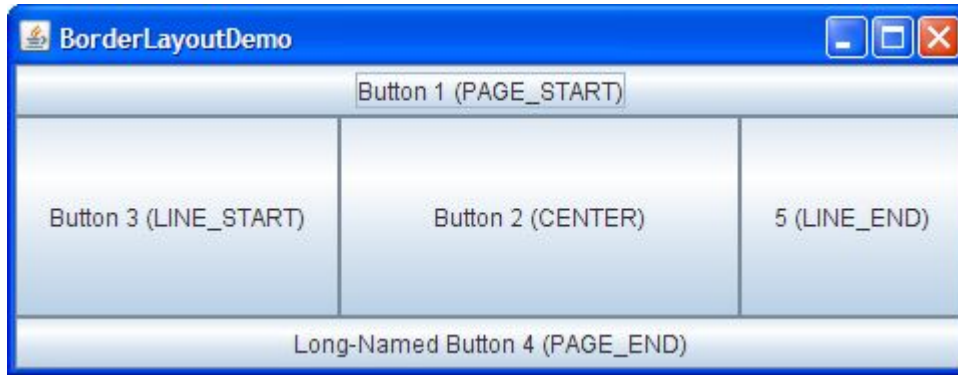
1. Definir el layout como nulo para el contenedor. *setLayout(null)*.
2. Para cada uno de los componentes que introduzcamos en el contenedor, hay que definir su posición y tamaño mediante *setBounds()*;

Utilizar contenedores con este layout tiene inconvenientes cuando redimensionamos.

# Layouts



## BORDER LAYOUT:



El layout por defecto los JRootPane. Hay cinco posiciones prefijadas en este layout.

Todo el espacio sobrante de los componentes es utilizado por el componente del medio.

# Layouts



## GRID LAYOUT:



Este layout simplemente hace que los componentes que añadamos sean igual en tamaño y los muestra en un número de filas y columnas con el que se inicializa el GridLayout.

# Layouts



## GRID BAG LAYOUT:

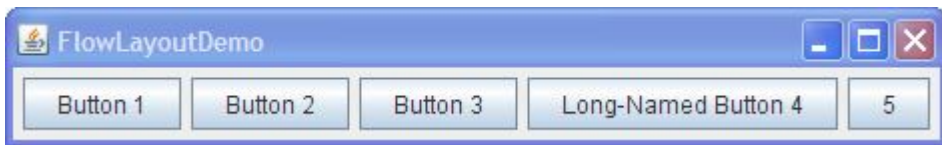


Este layout es más sofisticado y preciso que los anteriores. Alinea a los componentes mediante celdas, permitiendo que los componentes ocupen más de una celda. Tanto filas como columnas pueden tener diferentes anchos y altos.

# Layouts



## FLOW LAYOUT:

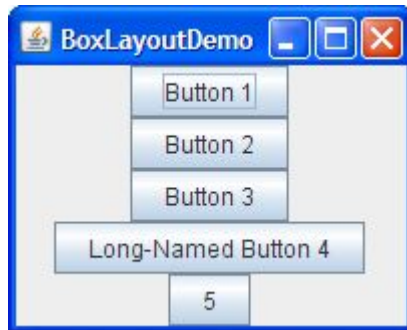


Coloca los elementos en la misma fila. Una vez no queda espacio para añadir más componentes, los pasa a la siguiente fila.

# Layouts

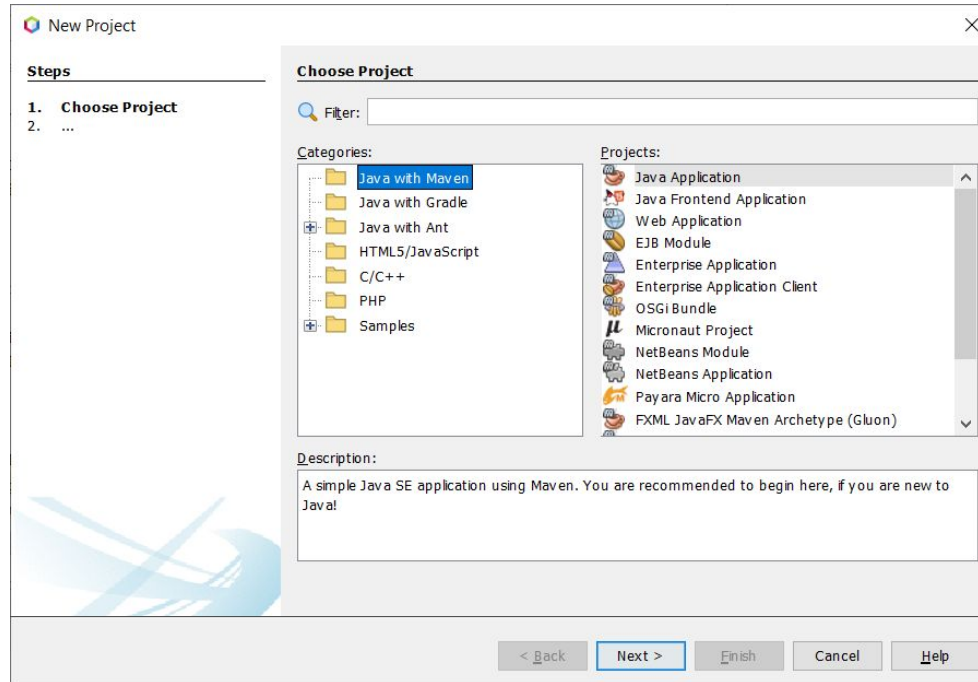


## BOX LAYOUT:

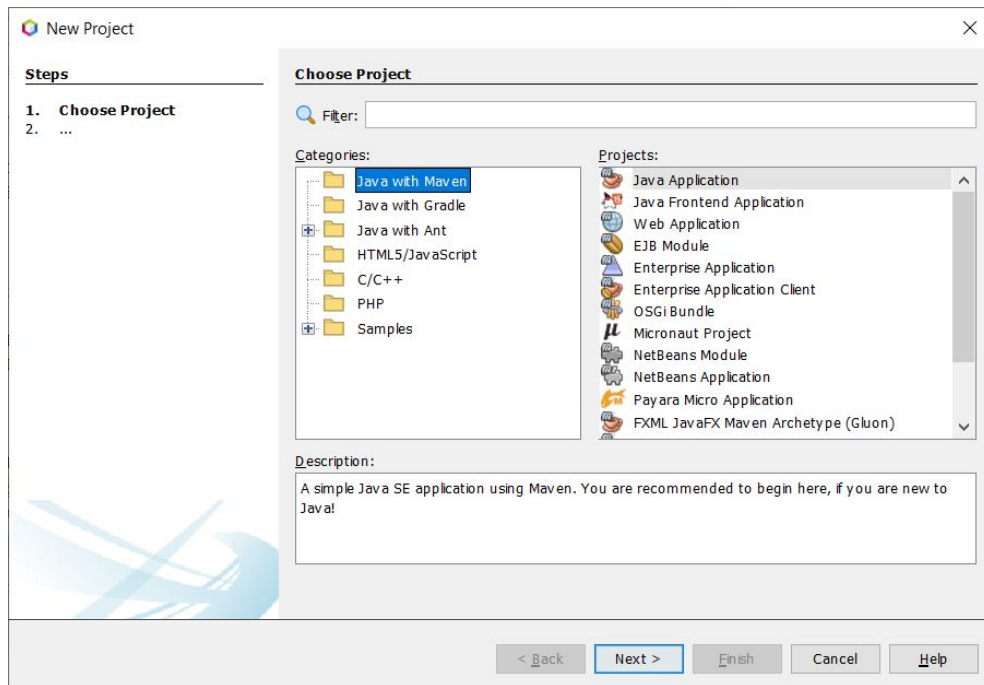


Este layout pone los componentes en la misma columna, respetando el tamaño de los componentes. Permite alinear los componentes, por ejemplo, en la imagen en el centro.

# Interfaces CON NETBEANS: Introducción

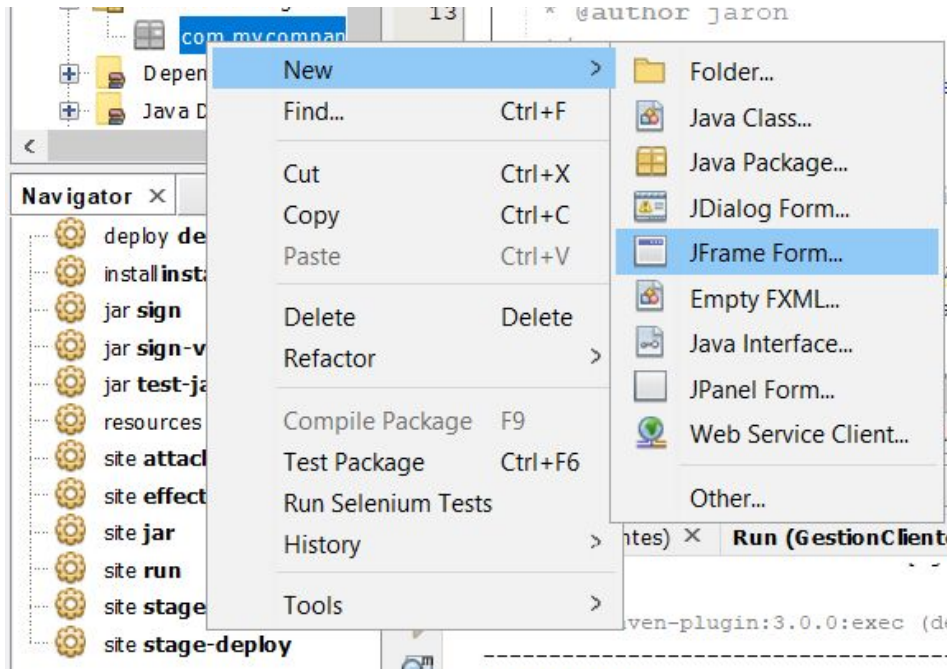


# Crear un nuevo proyecto Java con Netbeans





# En el paquete, crear un nuevo JFrame



# Añadir Botón

ans IDE 12.4

File Edit View Source Refactor Run Debug Profile Team Tools Window Help

The screenshot shows the NetBeans IDE 12.4 interface. The main editor displays a Java Swing window titled "Mi primer Botón" with a single button. The "Palette" window on the right lists various Swing controls under "Swing Controls" and "Swing Menus". The "Other Components - Properties" window is empty, showing "<No Properties>". The "Output" window at the bottom shows three tabs for running the application.

Start Page x PantallaPrincipal.java x DialogoAlta.java x Cliente.java x Principal.java x

Source Design History

The Preview Design button (in the toolbar) enables you to test the design of the form.

Mi primer Botón

Palette x

Swing Controls

- Label
- Button
- Toggle Button
- Check Box
- Radio Button
- Button Group
- Combo Box
- List
- Text Field
- Text Area
- Scroll Bar
- Slider
- Progress Bar
- Formatted Field
- Password Field
- Spinner
- Separator
- Text Pane
- Editor Pane
- Tree
- Table

Swing Menus

- Menu Bar
- Menu
- Menu Item
- Menu Item / CheckBox
- Menu Item / RadioButton
- PopUp Menu

Other Components - Properties x

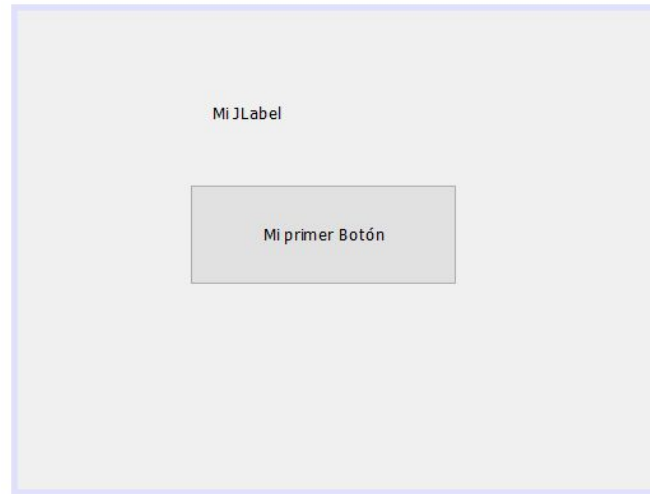
<No Properties>

Other Components

Output x

Run (GestionClientes) x Run (GestionClientes) x Run (GestionClientes) x

# Añadir JLabel



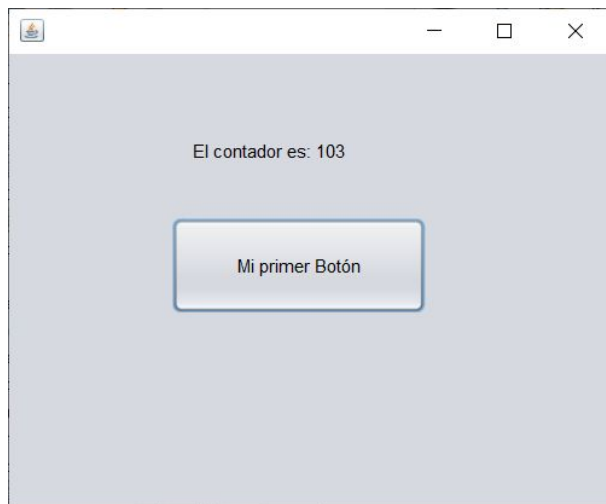
# Cambiar texto al pulsar el botón

Al hacer doble clic sobre el botón, nos lleva al código

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {  
    //Cambiar texto al pulsar boton  
  
    this.jLabel1.setText("Nuevo Texto");  
}
```

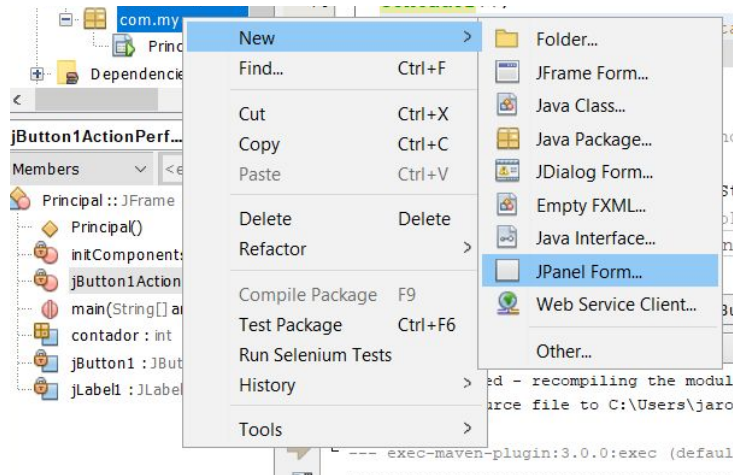
# Ejercicio 1

Crea una aplicación que cuente el nº de clic que va dando el usuario.



# Crear un nuevo JDialog Form

Para mostrar el nuevo JDialog desde un botón del principal:

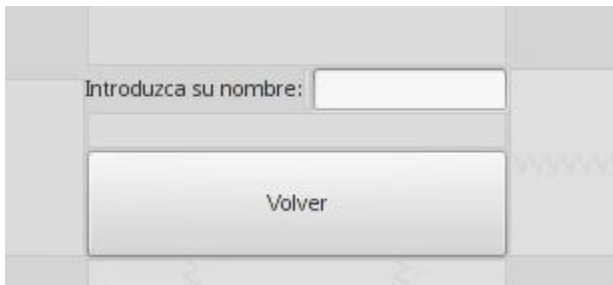


```
private void jButtonDialogoActionPerformed(java.awt.event.ActionEvent evt) {  
    PantallaSecundaria pantallaSecundaria = new PantallaSecundaria(this,true);  
    pantallaSecundaria.setVisible(true);  
}
```

Para cerrar el secundario (desde un botón)

```
private void jButtonVolverActionPerformed(java.awt.event.ActionEvent evt) {  
    setVisible(false);  
}
```

# Enviar datos de Secundario a Principal



Hago referencia a la pantalla principal desde el constructor del JDialog

```
private PantallaPrincipal pantallaPrincipal;  
/**  
 * Creates new form PantallaSecundaria  
 */  
public PantallaSecundaria(java.awt.Frame parent, boolean modal) {  
    super(parent, modal);  
    pantallaPrincipal = (PantallaPrincipal)parent;  
    initComponents();  
}
```

En principal creo un método para cambiar el JLabel

```
public void establecerNombre(String nombre)  
{  
    jLabelSaludo.setText("Bienvenido a la aplicación "+nombre);  
}
```

## En el JDialog

Paso el texto del campo de texto al principal a través del método público que hemos creado.

```
private void jButtonVolverActionPerformed(java.awt.event.ActionEvent evt) {  
    String nombre = jTextFieldNombre.getText();  
    pantallaPrincipal.establecerNombre(nombre);  
    setVisible(false);  
}
```



## Ejercicio 2

Crea una aplicación con 3 botones(Abrir JDialog 1, Abrir JDialog 2, Abrir JDialog 3)

Cada botón abrirá un nuevo JDialog.

Al cerrar el JDialog, en el principal aparecerá el nº de JDialog que se ha cerrado.

Ampliación 1:

Al iniciar la aplicación uno de los botones (de forma aleatorio cada vez) tendrá premio. Se deberá mostrar en el JDialog correspondiente el mensaje: ¡Premio! . Si no se acierta se mostrará en el JDialog !Fallaste! Vuelve a intentarlo!.

Ampliación 2:

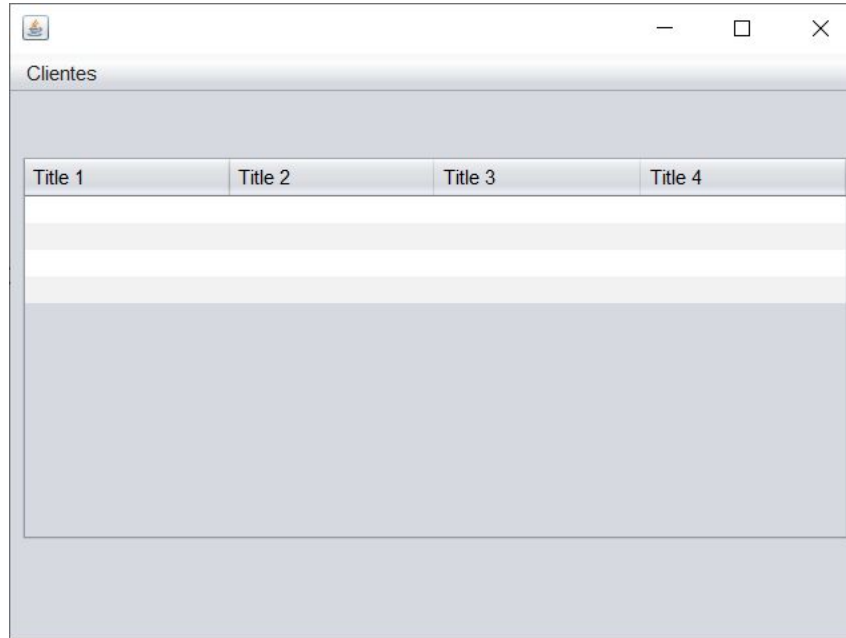
Crea un botón “Nuevo Juego” que vuelva a barajar los botones.



# Proyecto Guiado: Gestión Clientes

# Java Application - Gestión Clientes

- Añadir Menú - Añadir item al menú: Alta..
- Añadir tabla




# Añadir Alta

ComboBox

y

Spinner (cambiar de  
default a date)



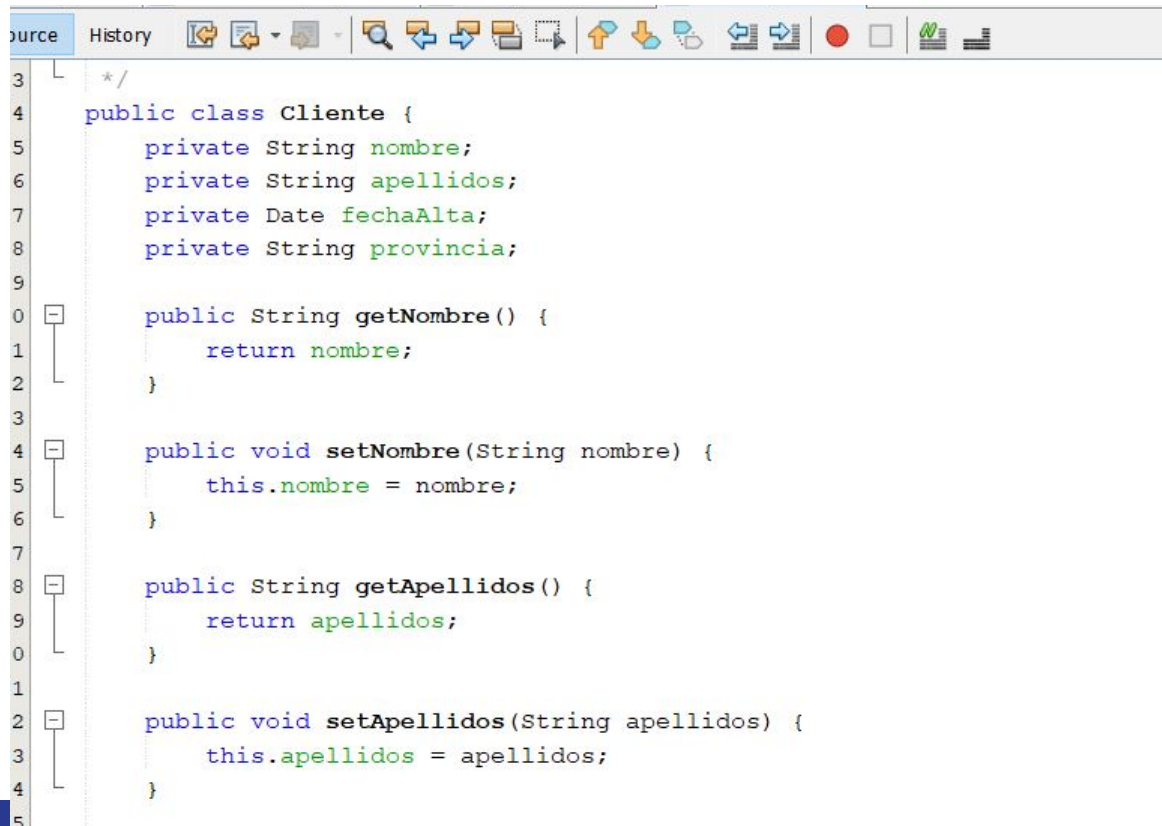
Nombre:

Apellidos:

Provincia:

Fecha de Alta:

# Crear Clase Cliente



```
3  */
4  public class Cliente {
5      private String nombre;
6      private String apellidos;
7      private Date fechaAlta;
8      private String provincia;
9
10     public String getNombre() {
11         return nombre;
12     }
13
14     public void setNombre(String nombre) {
15         this.nombre = nombre;
16     }
17
18     public String getApellidos() {
19         return apellidos;
20     }
21
22     public void setApellidos(String apellidos) {
23         this.apellidos = apellidos;
24     }
25 }
```

Pulsando alt + insert en netbeans podemos crear los getter y setter automáticamente.

Pulsando alt + enter sobre date lo importa solo.

# Añadir a la clase cliente

```
public String[] toArrayString() {  
    String[] s= new String[4];  
    s[0]= nombre;  
    s[1]= apellidos;  
    s[2]= provincia;  
    s[3]= fechaAlta.toString();  
    return s;  
}
```

Para pasar el  
cliente a array  
de String para  
rellenar la tabla.

```
public Cliente(String nombre, String apellidos, Date fechaAlta, String provincia) {  
    this.nombre = nombre;  
    this.apellidos = apellidos;  
    this.fechaAlta = fechaAlta;  
    this.provincia = provincia;  
}
```

Constructor!

# En pantallaPrincipal Inicializar tabla

```
private void inicializarTabla() {  
  
    DefaultTableModel dtm = new DefaultTableModel();  
    dtm.setColumnIdentifiers(new String[]{"Nombre", "Apellidos", "Provincia", "Fecha de Alta"});  
    this.jTableClientes.setModel(dtm);  
}
```

Para añadir un cliente a la tabla utilizamos el método previamente creado para convertir a array de strings

```
public void aniadirCliente(Cliente cliente) {  
  
    DefaultTableModel dts= (DefaultTableModel) jTableClientes.getModel();  
    dts.addRow(cliente.toArrayString());  
}
```

## En dialogo alta..

No olvides que hay que rescatar PantallaPrincipal en el constructor de la clase.

```
public class DialogoAlta extends javax.swing.JDialog {  
    private PantallaPrincipal principal;  
    /**  
     * Creates new form DialogoAlta  
     */  
    public DialogoAlta(java.awt.Frame parent, boolean modal) {  
        super(parent, modal);  
        initComponents();  
        principal = (PantallaPrincipal) parent;  
    }  
}
```



# En dialogo alta en la acción del botón:

Pasar el cliente pulsando el botón.

```
private void jButtonAltaActionPerformed(java.awt.event.ActionEvent evt) {  
    String nombre= this.jTextFieldNombre.getText();  
    String apellidos= this.jTextFieldApellidos.getText();  
    Date fechaAlta=(Date)this.jSpinnerFechaAlta.getValue();  
    String provincia= this.jComboBox1.getSelectedItem().toString();  
    Cliente cliente=new Cliente(nombre,apellidos,fechaAlta,provincia);  
    principal.aniadirCliente(cliente);  
  
    dispose();          // TODO add your handling code here:  
}
```

## Ejercicio 3

Crea una interfaz gráfica similar al ejercicio visto en clase, cogiendo de referencia el juego pokemon que hemos realizado, y que te permita:

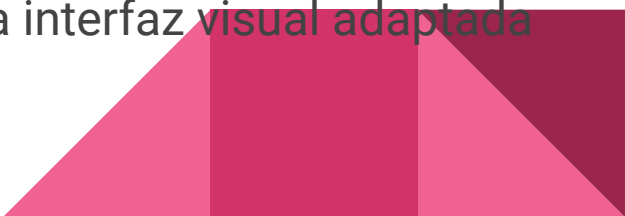
Ver en una tabla los Pokemon con sus datos correspondientes.

Se leerán del archivo correspondiente.

Añadir un nuevo Pokemon a la tabla.

Guardar el archivo con los nuevos datos.

El programa se llamará Creador de Pokemon y tendrá una interfaz visual adaptada al juego pokemon.



## Ejercicio 4

Crea una aplicación:

- En el principal se mostrará el botón Nuevo Juego y una tabla con los records.
  - Al iniciar un Jdialog modal, aparecerá un botón en el que tendremos que pulsar un botón, y el programa contará cuantos segundos lo hemos pulsado (ver <https://www.jc-mouse.net/java/contar-segundos-problema-resuelto> )
  - Al finalizar, aparecerá un nuevo Jdialog pidiendo el nombre.
  - Se guardará nuestro nombre, la fecha y el nº de segundos que hemos conseguido, y se añadirá a la tabla de la ventana principal.
- 