# Ejemplo de GUI con Swing

*Programación Orientada a Objetos*
*Facultad de Informática*

**Juan Pavón Mestras**
**Dep. Ingeniería del Software e Inteligencia Artificial**
**Universidad Complutense Madrid**

## Una aplicación Swing sencilla

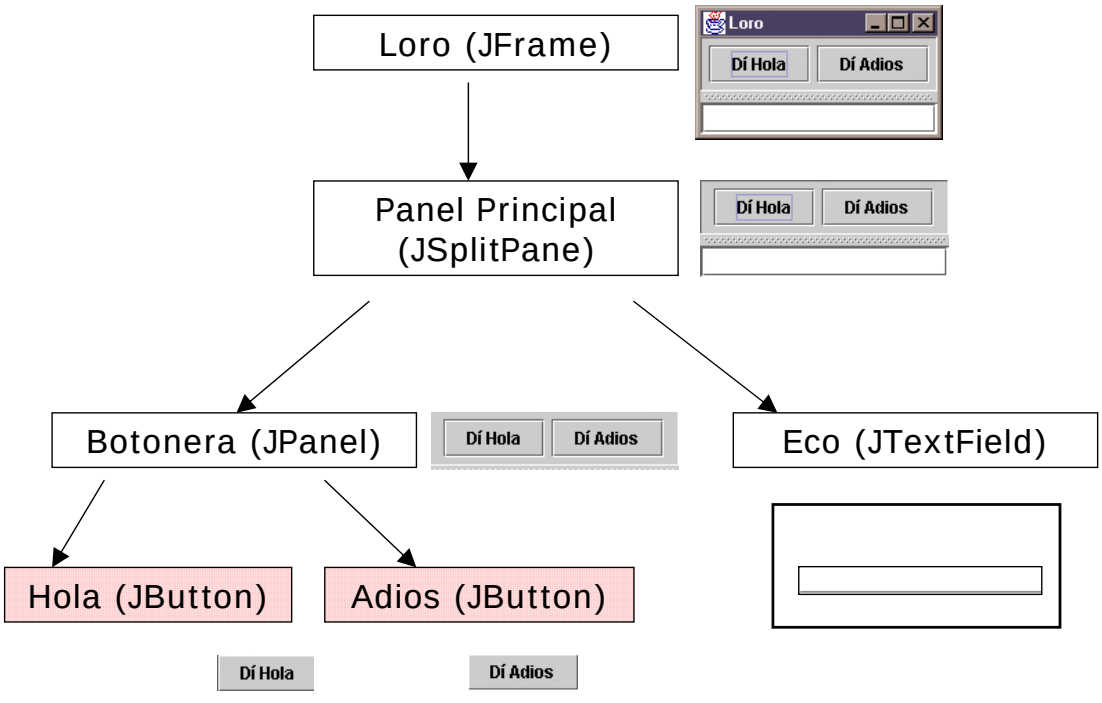
- El comportamiento deseado para esta aplicación es:
  - Cuando el usuario pulsa en el botón Dí Hola, en el campo de texto la aplicación ha de poner 'Hola'
  - Cuando el usuario pulsa en el botón Dí Adios, en el campo de texto la aplicación ha de poner 'Adios'
  - Cuando el usuario cierra la ventana, la aplicación ha de terminar

## Una aplicación Swing sencilla

- El diseño de toda interfaz conlleva, a grandes rasgos, los siguientes pasos:
  - Decidir la estructura de la interfaz
    - Qué componentes gráficos se van a utilizar, y cómo se van a relacionar estos componentes)
  - Decidir la disposición (*layout*) de los componentes
    - Existen dos tipos de componentes: contenedores y componentes atómicos
    - Los contenedores sirven para organizar los componentes contenidos en los mismos. Esta organización se denomina disposición (o *layout*)
  - Decidir el comportamiento de la interfaz: gestión de eventos
    - Algunos componentes son controles: permiten reaccionar ante eventos del usuario. El comportamiento se especifica programando las respuestas a dichos eventos. Normalmente, dichas respuestas supondrán invocar funcionalidades de la lógica de la aplicación
    - Conviene mantener la interfaz y la lógica lo más independientes posibles (veremos patrones que permiten lograr esto)



## Una aplicación Swing sencilla: estructura



## Una aplicación Swing sencilla: estructura

```
import javax.swing.*;

public class Loro extends JFrame {
        private JTextField eco;

        public Loro() {
                setTitle("Loro");

                JComponent botonera = creaBotonera();
                JComponent eco = creaEco();
                // Crea panel con botonera y eco
                JSplitPane panelPrincipal =
                    new JSplitPane(JSplitPane.VERTICAL_SPLIT,botonera,eco);
                // Añade el panel a la ventana principal
                getContentPane().add(panelPrincipal);
                // Se 'redimensiona' toda la interfaz gráfica en la ventana
                pack();
                // Y hace visible la ventana, con sus componentes
                setVisible(true);
        }
// ...
```



## Una aplicación Swing sencilla: estructura

```
        private JComponent creaBotonera() {
                JPanel botonera = new JPanel();
                // Se crean los botones ...
                JButton hola = new JButton("Dí Hola");
                JButton adios = new JButton("Dí Adios");
                // .. y se añaden al panel
                botonera.add(hola);
                botonera.add(adios);
                return botonera;
        }

        private JComponent creaEco() {
         // Se crea el campo de texto donde poner el eco
                eco = new JTextField("Pulsa un botón");
                return eco;
        }
// ...
```

## Una aplicación Swing sencilla: estructura

```
        public static void main(String[] args) {
                Loro loro = new Loro();
                //... aquí termina la ejecución del 'hilo principal',
                // ... pero queda pendiente la ejecución del hilo de
                // ... tratamiento de eventos... A partir de ahora toda la
                // ... ejecución es gobernada por la interacción con el usuario.
        }
}
```

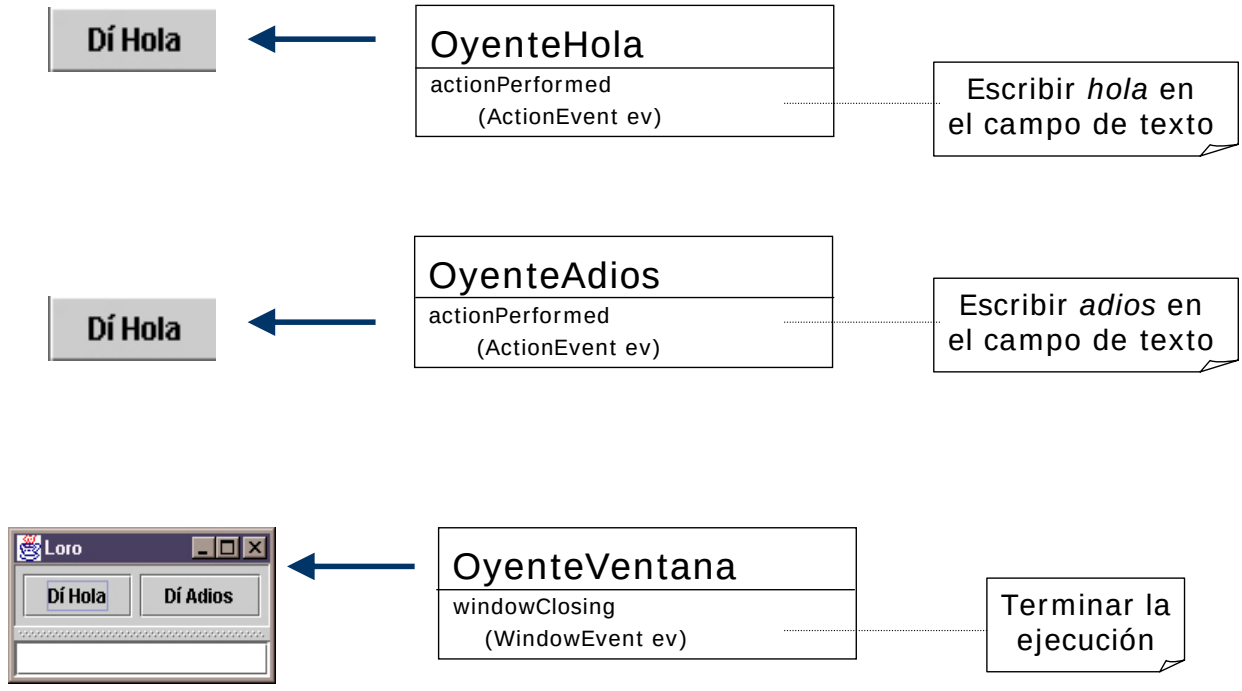


## Una aplicación Swing sencilla: comportamiento

- Los controles señalizan eventos
- Diferentes tipos de eventos, dependiendo de los controles
- La forma de tratar eventos en Swing (y en AWT, a partir de JDK 1.1) es mediante un mecanismo denominado delegación:
  - Por cada tipo de evento notificado por un control, el control acepta un oyente de dicho evento (métodos *addXXXListener*)
  - Dicho oyente ha de implementar una interfaz adecuada (*XXXListener*)
  - Cuando se produce un evento, el control invoca un método apropiado del oyente. Es en este método donde se trata el evento
  - Estas clases están declaradas en el paquete java.awt.event

```
import java.awt.event.*;
```

# Una aplicación Swing sencilla: comportamiento






# Una aplicación Swing sencilla: comportamiento asociado a los botones

```
private JComponent creaBotonera() {
        JPanel botonera = new JPanel(); // Panel para contener los botones
        // Se crean los botones ...
        JButton hola = new JButton("Dí Hola");
        hola.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent ev) {
                            eco.setText("Hola!");
                  }});

        JButton adios = new JButton("Dí Adios");
        adios.addActionListener(
                new ActionListener() {
                    public void actionPerformed(ActionEvent ev) {
                            eco.setText("Adios!");
                  }});

        // .. y se añaden al panel
        botonera.add(hola);
        botonera.add(adios);
        return botonera;
}
```

## Una aplicación Swing sencilla: comportamiento asociado a la ventana principal

```
public Loro() {
        setTitle("Loro");

        addWindowListener(
                new WindowAdapter() {
                        public void windowClosing(WindowEvent ev) {
                        // Se termina la ejecución de la aplicación
                                System.exit(0);
                         }
                });
        // ...
```



## Funcionamiento de las aplicaciones con GUI en Java

- Java es, por diseño, un lenguaje multiproceso: en un programa Java pueden existir (y de hecho existen) simultáneamente múltiples hilos de ejecución (*threads*) concurrentes
- Uno de estos hilos es el hilo de tratamiento de eventos
- En las aplicaciones con GUI, el hilo principal se limita a construir la estructura de la GUI, a asociar los oyentes adecuados con los controles y, hecho esto, termina …
- … pero la aplicación en sí no termina, puesto que todavía queda, al menos, un hilo con vida: el de **tratamiento de eventos**
  - Este hilo se encarga de tratar automáticamente eventos rutinarios (p.ej. redibujar una ventana cuando ésta pasa a primer plano, o cuando se quita una ventana que la ocultaba parcialmente, actualizar la presentación como resultado de cambios ordenados por la aplicación, etc.)
  - … y también se encarga de tratar los eventos de usuario, invocando a los oyentes previamente registrados