Intefaces gráficas. Eventos

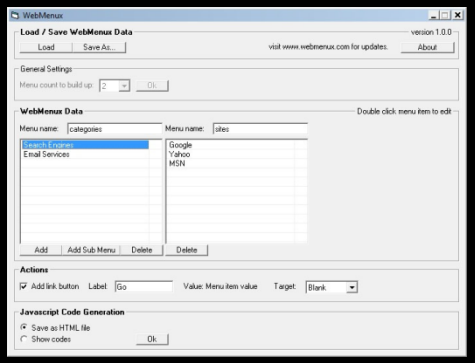
Jose Jesus García Rueda

Planteamiento de objetivos

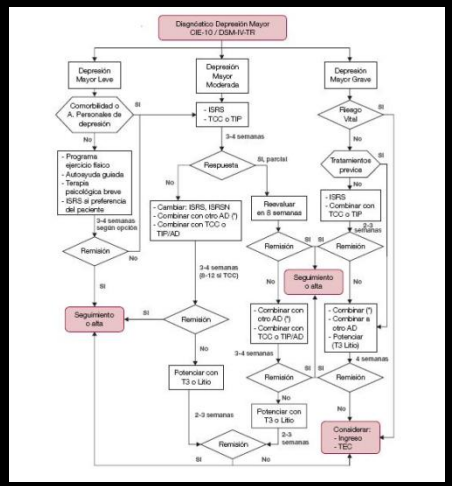
- Ser capaces de añadirle funcionalidad a los elementos gráficos de la interfaz...
- ...modificándolos también como resultado de las acciones sobre ellos.
- En otras palabras, realizar el ciclo completo:
  1. Recibir los eventos que tengan lugar sobre los elementos gráficos.
  2. Procesarlos.
  3. Presentar una realimentación en la pantalla.

Arquitectura de una aplicación gráfica

Interfaz



Procesamiento

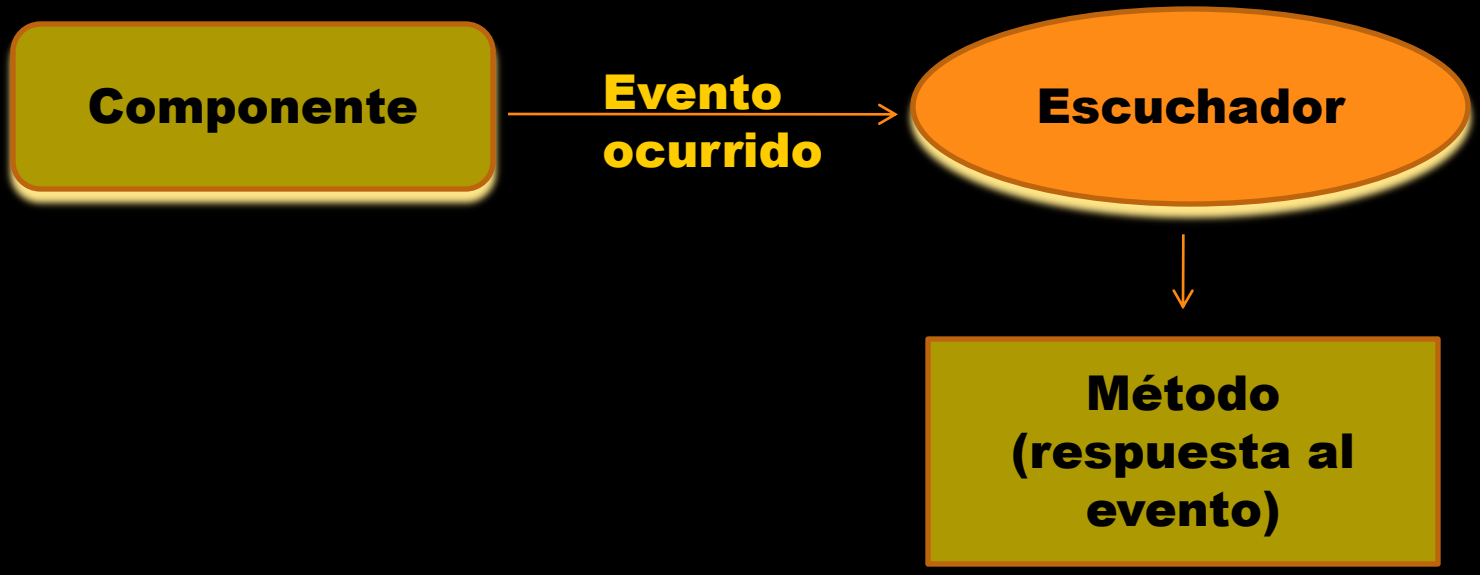


Persistencia

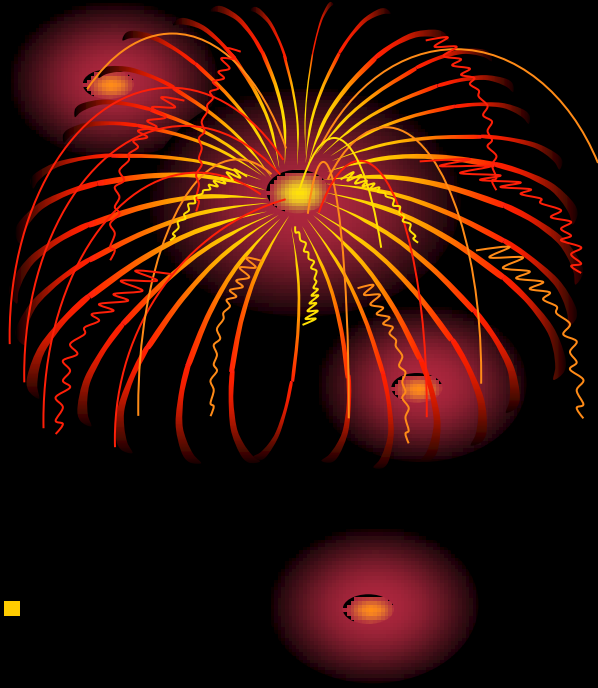
¿Cómo se crea este enlace?

¿Alguien me escucha?

- Cuando el usuario opere sobre la interfaz, algo deberá suceder.
- Para ello deberé programar manejadores de eventos (escuchadores)



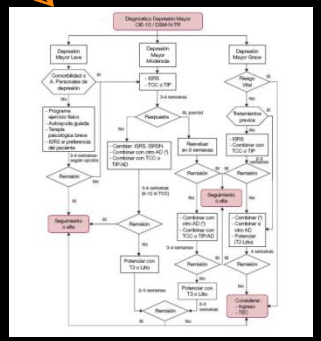
Ejemplos de escuchadores



- **WindowListener**
  - **Escuchador de ventanas.**
- **ActionListener**
  - **Escuchador de botones y otros componentes simples.**
- **¡Consultar continuamente la API!**

La espera activa

- **Una vez que la interfaz gráfica se ha “pintado” en la pantalla…**
- **¡…la aplicación queda a la espera, sin ejecutar ningún código!**



Cuando se acciona sobre la interfaz, el escuchador correspondiente se despierta

¿Y traducido a código?

Paquete que contiene los escuchadores

Los escuchadores suelen ser interfaces

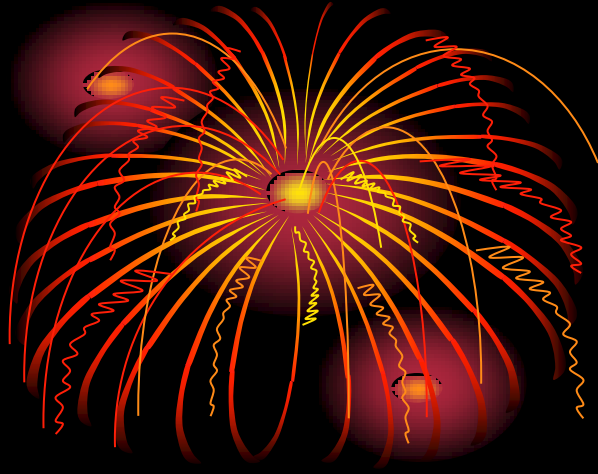
```
import java.awt.event.*;

public class Escuchador implements ActionListener {

   public void actionPerformed (ActionEvent e) {

      System.out.println("Dentro del escuchador");

   }

}
```

Método que se despierta automáticamente

¿Quién escucha a quién?

- Si tengo múltiples componentes gráficos...
- ...y puedo crear tantos escuchadores como quiera...
- ¿Quién escucha a quién?
- Habremos de asociar, explícitamente, los escuchadores a los componentes.
- Las posibles combinaciones son múltiples:
  - Varios escuchadores asociados a un mismo componente.
  - Un escuchador asociado a varios componentes.

¿Cómo realizar la asociación?



```
import javax.swing.*;

public class Ejemplo2 extends JFrame {

   JButton miBoton = new JButton ("Púlsame");
   Escuchador miEscuchador = new Escuchador();

   public Ejemplo2 () {
        getContentPane().add(miBoton);
        miBoton.addActionListener(miEscuchador);
   }

   public static void main (String[] arg) {
        Ejemplo2 ventana = new Ejemplo2();
        ventana.setSize(200, 200);
        ventana.setVisible(true);
     }
}
```

Creamos una instancia del escuchador que corresponda

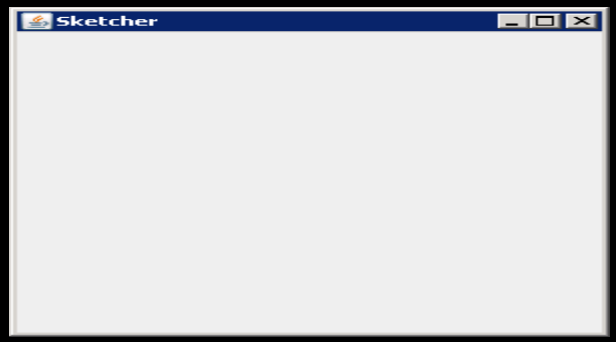
Asociamos el escuchador al componente

¿Qué parte del escuchador se despierta?

- Los escuchadores tienen diferentes métodos para escuchar los distintos eventos.
- Java invoca automáticamente el método oportuno, dependiendo del evento.
- El cuerpo de dichos métodos lo programaremos nosotros, pudiendo invocar desde él a otros métodos.
- Cuando el método termina su ejecución el programa vuelve a quedarse a la espera de nuevos eventos.
- Estos métodos reciben un objeto evento como argumento.

Ejemplo: WindowListener

- Entre los métodos que ofrece están:
  - void windowClosing (WindowEvent evt)
  - void windowOpened (WindowEvent evt)
  - void windowClosed (WindowEvent evt)
  - void windowIconified (WindowEvent evt)
  - void windowDeiconified (WindowEvent evt)
  - void windowActivated (WindowEvent evt)
  - void windowDeactivated (WindowEvent evt)



¿Puedo saber más cosas sobre los eventos?

- El evento que reciben como argumento los métodos de los escuchadores lo pasa Java automáticamente.
- “Preguntando” a ese objeto evento puedo averiguar más cosas sobre lo que realmente ocurrió.
- Preguntar, como siempre, supone invocar los distintos métodos del objeto evento.

Ejemplo

Argumento que pasa Java automáticamente

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

public class Escuchador implements ActionListener {

   public void actionPerformed (ActionEvent e) {

      String fuente = e.getActionCommand();
      System.out.println("Botón que produjo el evento: " + fuente);

   }

}
```

Devuelve la etiqueta del componente sobre el que se produjo el evento

Programación orientada a eventos

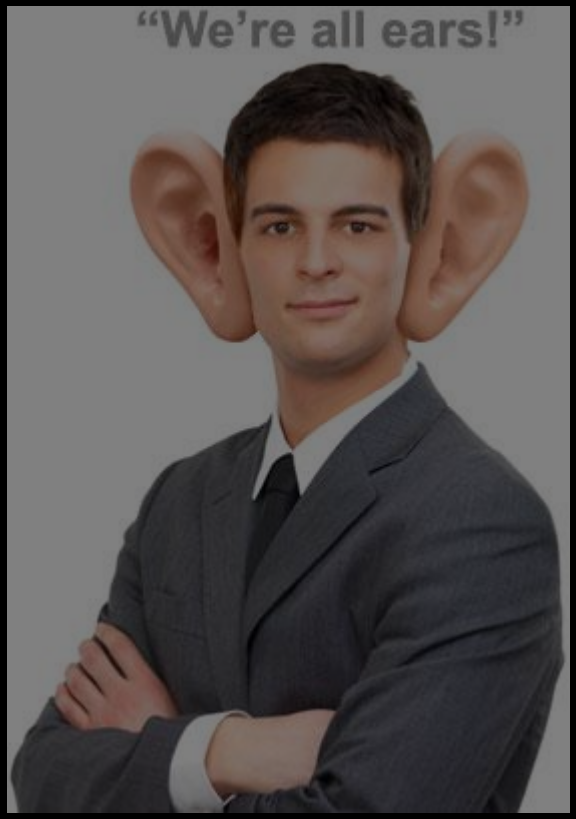
- Todo lo visto no es más que un caso particular de una técnica de programación muy importante y extendida: la Programación Orientada a Eventos.
- En un programa todo está bien planeado: se sabe a priori cuándo va a ocurrir...
- ...¿Cómo tener en cuenta entonces aquellos sucesos del mundo exterior que no sabemos con certeza cuándo ocurrirán?
  - ¿Cuándo se abrirá esa puerta?
  - ¿Cuándo llegará a hervir el agua?
  - ¿Cuándo pulsará el usuario tal botón?
- Los programas tienen mecanismos que les permiten reaccionar (“despertar”) cuando ocurren determinados eventos en el mundo exterior.

Organización del código

- Todo lo visto se rige por los principios de la OO ya conocidos...
- ...por lo tanto todo lo que conocemos hasta la fecha es perfectamente válido aquí.
- Lo único que hemos visto son nuevas piezas del mecano...
  - ...que pueden mezclarse entre sí y con el resto de piezas de la forma en que nos parezca oportuno.
- Ejemplos:
  - Poner los escuchadores en clases independientes.
  - Poner los escuchadores como clases internas.
  - Que los propios componentes gráficos actúen como escuchadores.
  - Escuchadores asociados a más de un elemento gráfico.

Adaptadores

- Algunas interfaces de escuchadores tienen muchos métodos...
- ...que habremos de implementar en su totalidad (son interfaces).
- Los adaptadores son clases que implementan de serie todos los métodos de un escuchador determinado.
- Como son clases, basta con heredarlas y reescribir los métodos que necesitemos.
- Para cada interfaz Listener, hay una clase Adapter:
  - WindowListener → WindowAdapter
  - KeyListener → KeyAdapter
  - MouseListener → MouseAdapter


“We’re all ears!”