Gestión de eventos

Cuando se actúa sobre algún componente de una interfaz gráfica, se genera un evento.

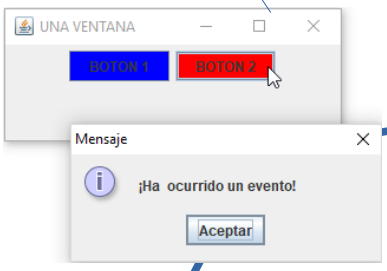
Java crea un objeto de esa clase de evento, y este evento se transmite a un determinado método para que lo gestione.

La responsabilidad de gestionar un evento que ocurre en una interfaz gráfica la tiene otra clase oyente que habrá que crear.

Habrá que registra un objeto de la clase Oyente en el componente fuente para que todo funcione..

FUENTE DE EVENTOS

Cuando se pulsa un botón,
Java crea un objeto ActionEvent que
se transmite al método ActionPerformed



CONECTAR OYENTE CON FUENTE

Creamos un objeto de la clase Oyente y lo registramos como oyente del componente origen del ActionEvent mediante addActionListener

```
Oyente pulsarBoton = new Oyente();
boton1.addActionListener(pulsarBoton);
boton2.addActionListener(pulsarBoton);
```

```
import java.awt.event.*;    // Eventos
import javax.swing.*;       // Componentes

public class Oyente implements ActionListener {

   @Override
   public void actionPerformed(ActionEvent event) {
      JOptionPane.showMessageDialog(null, "¡Ha  ocurrido un evento!");
   }
}
```

OYENTE DE EVENTOS

Creamos una Clase Oyente que
implemente la interfaz ActionListener y
desarrolle el método ActionPerformed

Plantilla de interfaz gráfica con eventos

```
import java.awt.*;         // Componentes de dibujo
import javax.swing.*;      // Componentes de una interfaz gráfica
import java.awt.event.*;   // Eventos

// La clase implementa la interfaz ActionListener
public class ClaseGuiEventos implements ActionListener {

   // Programa principal
   public static void main(String[] args) {
      ClaseGuiEventos gui = new ClaseGuiEventos();
   }

   // Atributos
   private JFrame ventana;
   private JButton boton1;
   private JButton boton2;

   // Constructor
   public ClaseGuiEventos() {

      // Crea y configura la ventana principal
      ventana = new JFrame();
      ventana.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      ventana.setLocation(100, 50);
      ventana.setSize(300, 120);
      ventana.setTitle("UNA VENTANA");
      ventana.setLayout(new FlowLayout());

      // Crea y configura los componentes y los asigna a la ventana principal
      boton1 = new JButton();
      boton1.setText("BOTON 1");
      boton1.setBackground(Color.BLUE);
      ventana.add(boton1);

      boton2 = new JButton();
      boton2.setText("BOTON 2");
      boton2.setBackground(Color.RED);
      ventana.add(boton2);

      // Asignamos el Oyente al boton1
      boton1.addActionListener(this);

      // Asignamos el Oyente al boton2
      boton2.addActionListener(this);

      ventana.setVisible(true);
   }

   // Si ocurre el evento ActionEvent muestra un mensaje
   // Mediante el método getSource() identificamos que botón se ha pulsado
   // Estamos aprovechando el polimorfismo y deberemos usar el casting
   @Override
   public void actionPerformed(ActionEvent evento) {
      JButton boton = (JButton) evento.getSource();
      if (boton.getText().equals("BOTON 1")) {
         JOptionPane.showMessageDialog(null, "Se ejecutan las tareas del botón 1");
      } else {
         JOptionPane.showMessageDialog(null, "Se ejecutan las tareas del botón 2");
      }
   }

   // Creamos una clase interna si necesitamos otro ActionListener
   private class OtraAcciondiferente implements ActionListener {
   @Override
   public void actionPerformed(ActionEvent event) {
      JOptionPane.showMessageDialog(null, "Se ejecutan otras tareas");
   }
}
```