élogos

GESTIÓN DE EVENTOS

élogos

ÍNDICE

GESTIÓN DE EVENTOS

1. Eventos e interfaces de escucha

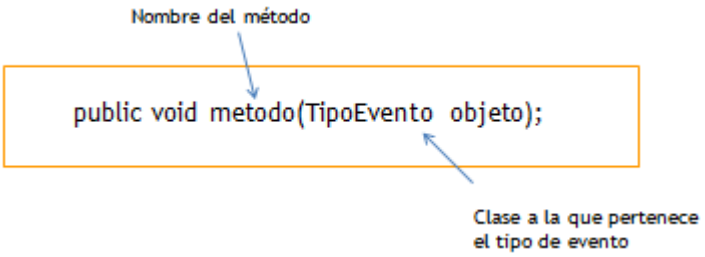
Definición

Un **evento** es un suceso que puede producirse sobre un elemento gráfico, como la pulsación de un botón, la selección de un elemento de una lista o la elección de una opción de menú.

Para responder a eventos en aplicaciones Java, tenemos las llamadas **interfaces de escucha**. Una interfaz de escucha es una interfaz Java en la que se definen los formatos de los métodos de respuesta a determinados eventos.

Las acciones de respuesta a los eventos deberán ser definidas en clases que implementen estas interfaces.

Formatos de métodos de respuesta a eventos



Cuando se produce un evento, se crea un objeto del tipo de evento producido y se pasa como parámetro al método de respuesta.

Como ejemplo, se muestra el siguiente método de respuesta al evento llegada del foco, definido en la interfaz de escucha FocusListener:

```
public void focusGained(FocusEvent objeto);
```

Principales clases de evento

En el paquete java.awt.event se encuentran la mayoría de las clases que representan a los principales eventos de una interfaz gráfica, tanto AWT como swing. Entre los más importantes están:

ActionEvent. Evento que representa una acción sobre un control, como la pulsación de un botón y presionar la tecla Enter sobre una caja de texto.

WindowEvent. Evento que representa un suceso en la ventana, como la pulsación del botón de cerrar, maximización o minimización

FocusEvent. Evento asociado al foco en un control, como la ganancia del foco o la pérdida del mismo

KeyEvent. Evento asociado a alguna acción sobre el teclado, como presionar o soltar una tecla

MouseEvent. Evento asociado a alguna acción sobre el ratón, como pulsación de uno de los botones o el movimiento del ratón

Ejemplo de una interfaz de escucha

Por regla general, por cada clase de evento existe una interfaz de escucha que contiene los métodos para responder a todas las posibles acciones asociadas al tipo de evento.

```
public class EscuchadorVentana implements WindowListener{
    public void windowOpened(WindowEvent e) {
       //apertura de la ventana
    }
    public void windowClosing(WindowEvent e) {
        //pulsación de botón de cierre de la ventana
    }
    public void windowClosed(WindowEvent e) {
        //la ventana se va a cerrar
    }
    public void windowIconified(WindowEvent e) {
        //minimización
    }
    public void windowDeiconified(WindowEvent e) {
        //restauración
    }
    public void windowActivated(WindowEvent e) {
        //la ventana se vuelve activa
    }
    public void windowDeactivated(WindowEvent e) {
        //la ventana se vuelve inactiva
    }
}
```

Principales interfaces de escucha

En el paquete java.awt.event se encuentran también la mayor parte de las interfaces de escucha que tendremos que implementar. Los nombres de estas interfaces se corresponden con los de las clases de evento, sustituyendo “Event” por “Listener”. Entre las más importantes están:

ActionListener. Contiene el método actionPerformed.

WindowListener. Contiene los métodos windowClosing, windowClosed, windowActivated, windowDeactivated, windowOpend, windowIconified, windowDeiconified

FocusListener. Contiene los métodos focusGained y focusLost

KeyListener. Contiene los métodos keyPressed, keyReleased y keyTyped

MouseListener. Contiene los métodos mouseClicked, mouseEntered, mouseExited, mousePressed y mouseReleased

2. Capturar un evento

Procedimiento

1. Implementación de la clase escuchadora
2. Creación del objeto escuchador
3. Asociar objeto escuchador a objeto gráfico

Implementación de la clase escuchadora

Lo primero será crear una clase que implemente la interfaz de escucha donde se encuentra el método o métodos asociados al evento o eventos que nos interesa capturar.

Esta clase puede ser una clase anónima, creada dentro de la propia clase de ventana, o una clase independiente.

La siguiente clase implementa el método de respuesta a la pulsación del botón de cierre de la ventana. En dicho método se procede a finalizar la aplicación:

```
public class EscuchadorVentana implements WindowListener{
   public void windowClosing(WindowEvent e) {
      //finaliza la aplicación
      System.exit(0);
   }
  : //resto de métodos de la interfaz
}
```

Nota

"Aunque solo estemos interesados en responder a un único evento, nuestra clase debe implementar todos los métodos definidos en la interfaz. Si no queremos programar ninguna acción, los dejaremos vacíos."

Creación del objeto escuchador

Para poder asociar el objeto gráfico origen del evento con el código de respuesta, será necesario crear un objeto de la clase escuchadora:

EscuchadorVentana listener = new EscuchadorVentana ();

Esto se suele realizar en el constructor de la clase ventana, que es donde normalmente se crean los objetos gráficos de la interfaz.

Asociar objeto escuchador con objeto gráfico

En el mismo constructor de la ventana, se realiza la asociación entre el objeto escuchador y origen del evento. Para ello, se utilizará uno de los métodos add*Interfaz(Interfaz escuchador)* proporcionados por la clase gráfica

La palabra *Interfaz* representa el nombre de la interfaz de escucha. Por ejemplo, para asociar el objeto gráfico con un escuchador de eventos ActionListener, sería addActionListener, mientras que para los del ratón sería addMouseListener.

En nuestro ejemplo la instrucción de asociación quedaría:

this.addWindowListener(listener);

Ejemplo

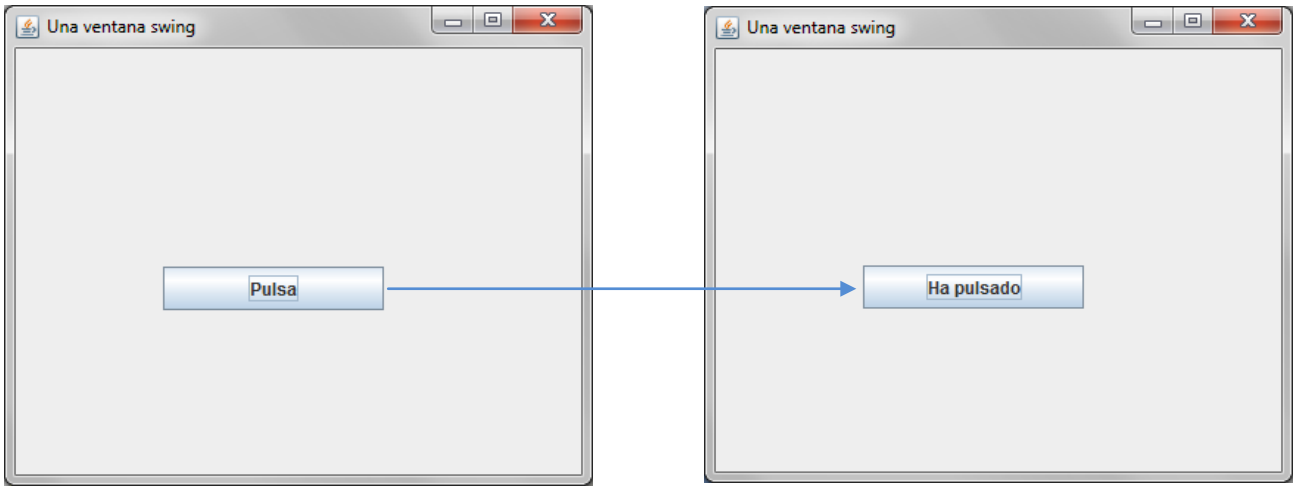
```
public class Ventana extends Frame{
   public Ventana(String titulo, int x, int y, int w, int h){
      super(titulo);
      //colocación ventana
      this.setBounds(x, y, h, h);

      EscuchadorVentana listener=new EscuchadorVentana();
      this.addWindowListener(listener);
      //visualizar ventana
      this.setVisible(true);
   }
   public Ventana(){
      //valores de creación por defecto
      this("ventana por defecto",20,20,400,300);
   }
   public static void main(String[] args){
      //crea el objeto Ventana
      new Ventana("Ventana segunda", 100, 50, 600,400);
   }
}
class EscuchadorVentana implements WindowListener{
    public void windowClosing(WindowEvent e) {
      System.exit(0);
   }
   :
}
```

Origen del evento

Todas las clases de evento heredan EventObject. Esta clase, que se encuentra en el paquete java.util, proporciona un método llamado *getSource()* que nos permite obtener, desde cualquier método escuchador, una referencia al objeto en el que se ha producido el evento.

Para ver su utilidad, veamos el siguiente ejemplo, en el que tenemos una ventana con un botón que, al ser pulsado, cambiará el texto mostrado en el mismo.




```
import javax.swing.*;
public class JVentana extends JFrame{
   public JVentana(String titulo, int x, int y, int w, int h){
      super(titulo);
      this.setBounds(x, y, h, h);
      this.setLayout(null);
      //controles
      JButton b=new JButton("Pulsa");
      b.setBounds(100, 150, 150, 30);
      b.addActionListener(new EscuchadorBoton());
      this.add(b);
      this.setVisible(true);
   }
   public JVentana(){
      //valores de creación por defecto
      this("ventana por defecto",20,20,400,300);
   }
   public static void main(String[] args){
      //crea el objeto JVentana
      new JVentana("Una ventana swing", 100, 50, 600,400);
   }
}
```

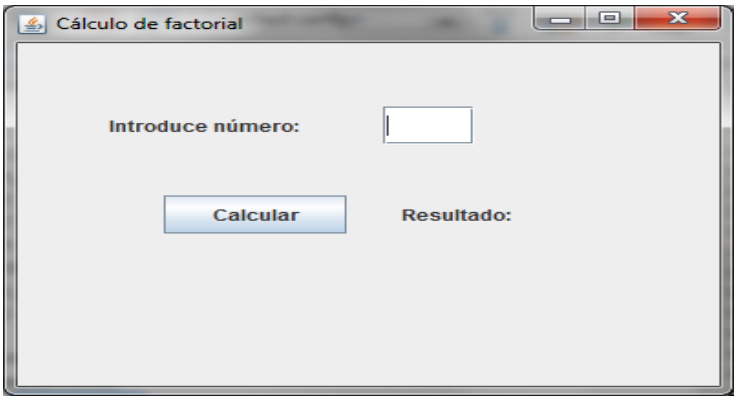
```
import javax.swing.*;
import java.awt.event.*;
public class EscuchadorBoton implements ActionListener{
   public void actionPerformed(ActionEvent e) {
      JButton boton=(JButton)e.getSource();
      boton.setText("Ha pulsado");
   }

}
```

Referencia a objetos de la interfaz

Es muy común que, a la hora de codificar las instrucciones de respuesta a un evento, necesitemos una referencia a un objeto de la interfaz gráfica.

Si el escuchador se codifica como una clase anónima, se podrá acceder directamente a los controles si estos se definen como atributos de la clase ventana. Si el escuchador es una clase independiente, deberá definir un constructor que reciba como parámetro la clase ventana para poder acceder a sus controles.



Versión 1

```
package ejemploeventos;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.*;
public class JVentana extends JFrame{
   JLabel jl,jlr;
   JTextField jt;
   JButton b;
   public JVentana(String titulo, int x, int y, int w, int h){
      super(titulo);
      this.setBounds(x, y, h, h);
      this.setLayout(null);
      //controles
      jl=new JLabel("Introduce número:");
      jl.setBounds(50, 50, 150, 30);
      jt=new JTextField();
      jt.setBounds(200, 50, 50, 30);
      jlr=new JLabel("Resultado:");
      jlr.setBounds(210, 120, 150, 30);
      b=new JButton("Calcular");
      b.setBounds(80, 120, 100, 30);
      //clase anónima para gestión de evento
      ActionListener EscuchadorBoton=new ActionListener() {

         @Override
         public void actionPerformed(ActionEvent e) {
            long num=Long.parseLong(jt.getText());
            long r=1;
            for(int i=1;i<=num;i++){
```

Versión 2

```
import javax.swing.*;
public class JVentana extends JFrame{
   JLabel jl,jlr;
   JTextField jt;
   JButton b;
   public JVentana(String titulo, int x, int y, int w, int h){
      super(titulo);
      this.setBounds(x, y, h, h);
      this.setLayout(null);
      //controles
      jl=new JLabel("Introduce número:");
      jl.setBounds(50, 50, 150, 30);
      jt=new JTextField();
      jt.setBounds(200, 50, 50, 30);
      jlr=new JLabel("Resultado:");
      jlr.setBounds(210, 120, 150, 30);
      b=new JButton("Calcular");
      b.setBounds(80, 120, 100, 30);
      //clase anónima para gestión de evento
      EscuchadorBoton escuchador=new
EscuchadorBoton(this);
      b.addActionListener(escuchador);
      this.add(jl);
      this.add(jt);
      this.add(jlr);
      this.add(b);
      this.setVisible(true);
   }
```

```
                r*=i;
            }
            jlr.setText("Resultado: "+r);
        }
    };
    b.addActionListener(EscuchadorBoton);
    this.add(jl);
    this.add(jt);
    this.add(jlr);
    this.add(b);
    this.setVisible(true);
  }
  public JVentana(){
    //valores de creación por defecto
    this("ventana por defecto",20,20,400,300);
  }
  public static void main(String[] args){
    //crea el objeto JVentana
    new JVentana("Cálculo de factorial", 100, 50, 600,400);
  }
}
```

```
  public JVentana(){
    //valores de creación por defecto
    this("ventana por defecto",20,20,400,300);
  }
  public static void main(String[] args){
    //crea el objeto JVentana
    new JVentana("Cálculo de factorial", 100, 50, 600,400);
  }
}
//////////////////////////
import java.awt.event.*;
public class EscuchadorBoton implements ActionListener{
  private JVentana v;
  public EscuchadorBoton(JVentana jv){
    //recibe el objeto JVentana
    v=jv;
  }
  public void actionPerformed(ActionEvent e) {
        long num=Long.parseLong(v.jt.getText());
        long r=1;
        for(int i=1;i<=num;i++){
          r*=i;
        }
        v.jlr.setText("Resultado: "+r);
      }
}
```

Eventos del ratón

Los métodos para respuesta a los eventos relacionados con el ratón se encuentran repartidos en dos interfaces: MouseListener (eventos sin movimiento) y MouseMotionListener(eventos con movimiento).

En todos los casos, los métodos reciben un objeto MouseEvent, que entre sus principales métodos para obtener información sobre el evento se encuentran:

getX. Coordenada x del punto donde se ha producido el evento, tomando como referencia el borde izquierdo del control

getY. Coordenada y del punto donde se ha producido el evento, tomando como referencia el borde superior del control

isMetaDown. Indica si la tecla meta (normalmente, botón derecho del ratón) estaba activada

getModifiers(). Devuelve una máscara que indica la tecla o teclas que han sido activadas. Para controlar la máscara, se pueden utilizar las constantes definidas en InputEvent.

Eventos del teclado

Los métodos para respuesta a los eventos relacionados con el ratón se encuentran definidos en la interfaz KeyListener.

En todos los casos, los métodos reciben un objeto KeyEvent con el que poder obtener información sobre la tecla que ha generado el evento. Entre sus principales métodos tenemos:

getKeyChar. Devuelve, como un tipo char, el carácter que ha provocado el evento.

getKeyCode. Devuelve el código de la tecla que ha provocado el evento. Los códigos de tecla están definidos como constantes en la propia clase KeyEvent.

isActionKey. Indica si alguna de las teclas de acción (alt, control y mayúsculas) ha sido activada.