

Guía de referencia rápida - Eventos

Eventos

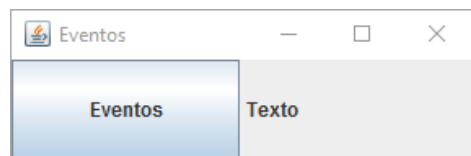
Es una acción que realiza el usuario en la aplicación, en Java se utilizan Listener.

Listener	Descripción	Métodos	Eventos
ActionListener	Se produce al hacer click en un componente, también si se pulsa Enter teniendo el foco en el componente.	actionPerformed	JButton: Click o Pulsar Enter con el foco activado en él. JList: Doble click en un elemento de la lista. JMenuItem: selecciona una opción del menú. JTextField: al pulsar Enter con el foco activado.
KeyListener	Se produce al pulsar una tecla. según el método cambiara la forma de pulsar la tecla.	keyTyped keyPressed keyReleased	keyTyped: al pulsar y soltar la tecla. keyPressed: al pulsar la tecla. keyReleased: al soltar la tecla.
FocusListener	Se produce cuando un componente gana o pierde el foco, es decir, que este seleccionado.	focusGained focusLost	Recibir o perder el foco.
MouseListener	Se produce cuando realizamos una acción con el ratón.	mouseClicked mouseEntered mouseExited mousePressed mouseReleased	mouseClicked: pinchar y soltar. mouseEntered: entrar en un componente con el puntero. mouseExited: salir de un componente con el puntero. mousePressed: presionar el botón. mouseReleased: soltar el botón.
MouseMotionListener	Se produce con el movimiento del mouse.	mouseDragged mouseMoved	mouseDragged: click y arrastrar un componente. mouseMoved: al mover el puntero sobre un elemento.

❑ ActionListener



Crear una ventana con dos componentes (Botón y Etiqueta)



`implements ActionListener` ①

`public class Ventana extends JFrame` ↓ {

② 1) Add import for java.awt.event.ActionListener

2) Implement all abstract methods

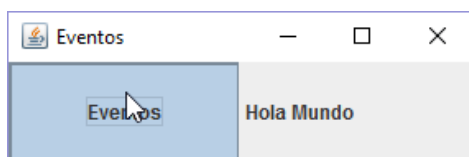
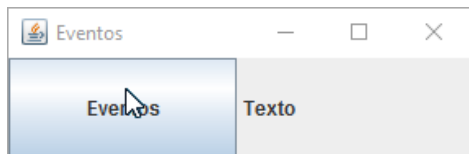
`import java.awt.event.ActionListener;`

@Override

```
public void actionPerformed(ActionEvent e) {  
    throw new UnsupportedOperationException ("Not supported yet.");  
    //To change body of generated methods, choose Tools | Templates.  
}
```

③ `L1.setText("Hola Mundo");`

④ `B1.addActionListener(this);`




① Agregar el Listener que se utilizará (ActionListener).

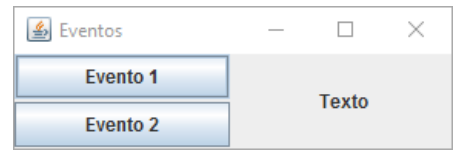
② Se marcan dos advertencias la primera la biblioteca a utilizar, y la segunda la implementación de los métodos abstractos.

③ Se implementa el código de acción requerida, sustituyendo al que esta por omisión

④ Se agrega el Listener en el componente para que se pueda ejecutar la acción.


❑ ActionListener (2 componentes)

 Crear una ventana con 3 componentes (2 Botones y 1 Etiqueta)



implements ActionListener ❶

public class Ventana extends JFrame {

- ❷  1) Add import for java.awt.event.ActionListener
2) Implement all abstract methods

import java.awt.event.ActionListener;

@Override

```
public void actionPerformed(ActionEvent e) {  
    throw new UnsupportedOperationException ("Not supported yet.");  
    //To change body of generated methods, choose Tools | Templates.  
}
```

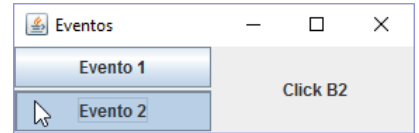
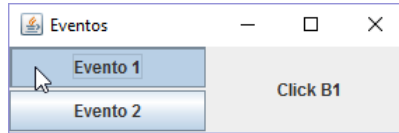
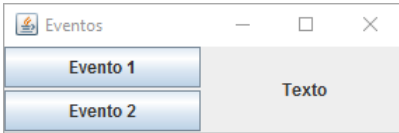
❸

```
if (e.getSource().equals(B1))  
{ L1.setText("Click B1"); }
```


```
B1.addActionListener(this);  
B2.addActionListener(this);
```

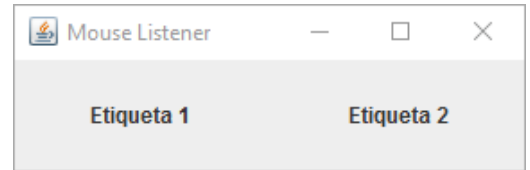
```
if (e.getSource().equals(B2))  
{ L1.setText("Click B2"); }
```

Detectar que componente ejecuta la acción.




❑ MouseListener

 Crear una ventana con dos componentes (2 Etiqueta)



implements MouseListener

public class Ventana extends JFrame {

-  1) Add import for java.awt.event.MouseListener
2) Implement all abstract methods

```
L1.addMouseListener(this);
```

@Override

```
public void mouseClicked(MouseEvent e)  
{ }
```

@Override

```
public void mousePressed(MouseEvent e)  
{ } ← L2.setText("Pressed Button [" + e.getButton() + "]);
```

@Override

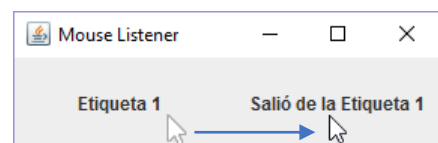
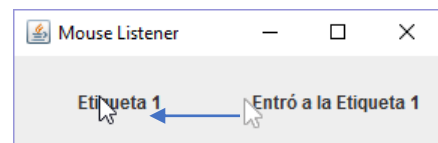
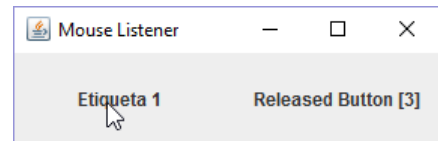
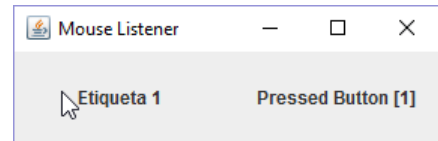
```
public void mouseReleased(MouseEvent e)  
{ } ← L2.setText("Released Button [" + e.getButton() + "]);
```

@Override


```
public void mouseEntered(MouseEvent e)  
{ } ← L2.setText("Entró a la Etiqueta 1");
```

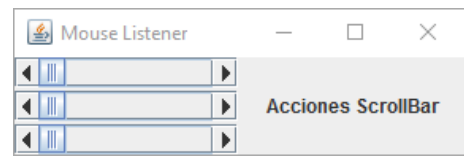
@Override

```
public void mouseExited(MouseEvent e)  
{ } ← L2.setText("Salió de la Etiqueta 1");
```




❑ adjustmentValueChanged

 Crear una ventana con 4 componentes (3 ScrollBar y 1 Etiqueta)



implements AdjustmentListener ①

public class Ventana extends JFrame ↓ {

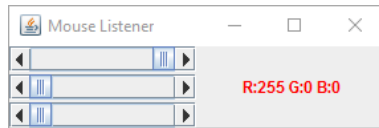
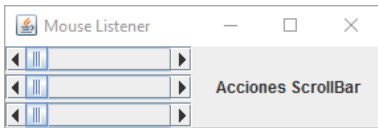
- ②  1) Add import for java.awt.event AdjustmentListener;
2) Implement all abstract methods

import java.awt.event AdjustmentListener;

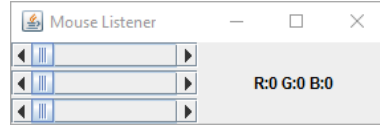
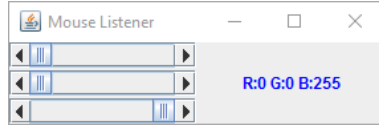
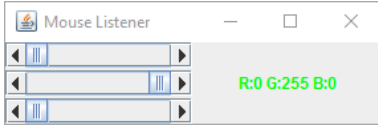
@Override

```
public void adjustmentValueChanged(AdjustmentEvent e) {  
    throw new UnsupportedOperationException ("Not supported yet.");  
    //To change body of generated methods, choose Tools | Templates.  
}
```


③ L1.setText ("R:" + SR.getValue () + " G:" + SG.getValue () + " B:" + SB.getValue ());
L1.setForeground (new Color (SR.getValue (), SG.getValue (), SB.getValue ());

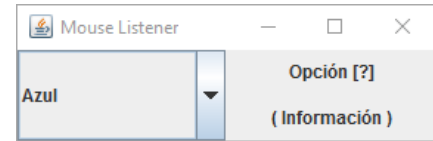


④ SR.addAdjustmentListener (this);
SG.addAdjustmentListener (this);
SB.addAdjustmentListener (this);




❑ ActionListener

 Crear una ventana con 3 componentes (1 ComboBox y 2 Etiquetas)



implements ActionListener ①

public class Ventana extends JFrame ↓ {

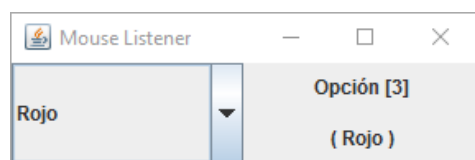
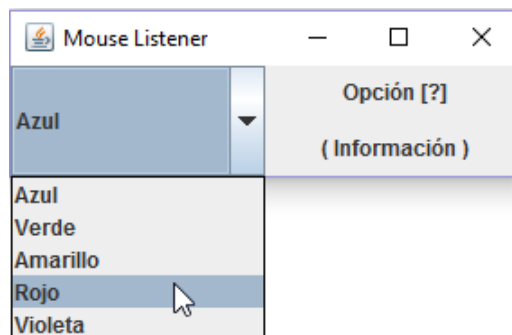
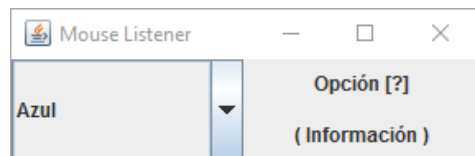
- ②  1) Add import for java.awt.event.ActionListener
2) Implement all abstract methods

③ // Eventos ComboBox
C1.addActionListener (this);


import java.awt.event.ActionListener;

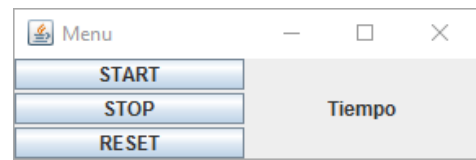
@Override

```
public void actionPerformed(ActionEvent e) {  
    if (e.getSource ().equals (C1))  
    { L1.setText ("( " + C1.getItemAt (C1.getSelectedIndex ()) + " )");  
      L2.setText ("Opción [" + C1.getSelectedIndex () + "]"); }  
}
```




❑ actionPerformed (Timer)

 Crear una ventana con 4 componentes (3 Botones y 1 Etiqueta)



implements ActionListener ①

public class Ventana extends JFrame ↓ {

- ②  1) Add import for java.awt.event.ActionListener
2) Implement all abstract methods

import java.awt.event.ActionListener;

@Override

```
public void actionPerformed(ActionEvent e) {  
    throw new UnsupportedOperationException ("Not supported yet.");  
    //To change body of generated methods, choose Tools | Templates.  
}
```

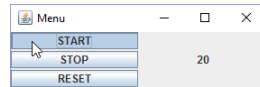
if (e.getSource().equals(B1))
{ Tiempo.star(); }

Detectar que componente ejecuta la acción.

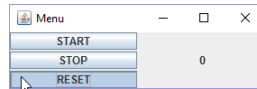
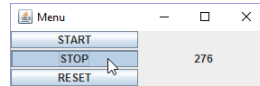
if (e.getSource().equals(B2))
{ Tiempo.stop(); }

El componente **Timer** también ejecuta **actionPerformed**

```
if (e.getSource().equals(B3))  
{  
    cuenta=0;  
    L1.setText (""+cuenta);  
}
```



```
if (e.getSource().equals(TIEMPO))  
{ L1.setText (""+cuenta); }
```



④

Acciones de los botones

```
B1.addActionListener(this);  
B2.addActionListener(this);  
B3.addActionListener(this);
```

Funciones del Timer

star() Inicia temporizador
Stop() Detiene temporizador

Declaración

Timer Tiempo;

Instancia

```
Tiempo = new Timer(ms,this)  
Tiempo = new Timer(100,this)
```