Transparencias

de

Java

Tema 8:
Gestión de eventos

w TEMA 8: GESTIÓN DE EVENTOS CON AWT

◆ LA GESTIÓN DE EVENTOS

- ✶ Programación Tradicional:
  - ❑ Generalmente, no existen modelos de gestión de eventos definidos.
  - ❑ El programa debe, mediante un bucle, preocuparse de monitorizar constantemente las acciones que realiza el usuario.
  - ❑ Ejemplo: Si el usuario pulsa ENTER, el programa no lo detecta a no ser que explícitamente lo pregunte mediante un getChar()
- ✶ Programación en Java (y en otros lenguajes actuales)
  - ❑ Existen modelos de eventos definidos.
  - ❑ El programa es avisado automáticamente de todas aquellas acciones (eventos) en los que esté interesado.
  - ❑ Ejemplo: Si el usuario pulsa un botón, el entorno avisa al programa invocando un método especial que éste tiene reservado para la notificación de ese evento (“pulsar botón”).

◆ MODELO DE EVENTOS EN JAVA

- ✶ El modelo de eventos de Java distingue dos elementos:
  - ❑ Fuentes de Eventos
    - ■ Elementos sobre los que se producen los eventos.
    - ■ Ejemplos: un botón, una lista, un panel,...
  - ❑ Escuchadores de Eventos
    - ■ Elementos que reciben las notificaciones de los eventos
- ✶ ¿Quién puede ser una fuente de eventos?

- Cualquier componente de AWT (o SWING) sobre el que se puedan producir eventos (prácticamente todos)

¿Quién puede ser un escuchador de eventos?

- Cualquier objeto (perteneciente a una clase) que implemente alguno de los interfaces definidos en Java para la notificación de los eventos.

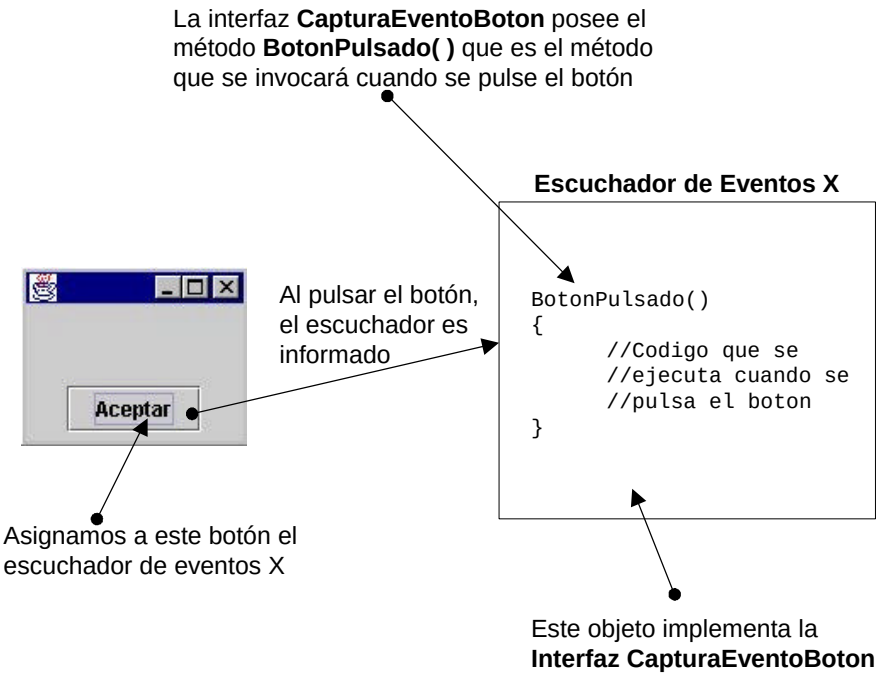
Funcionamiento:

- Toda “Fuente de Eventos” debe tener asignado un “Escuchador de Eventos” que reciba las notificaciones de sus eventos.
- Cuando se produce un evento sobre la fuente de eventos, su escuchador es informado.
- Para ello, se invoca el método que el escuchador tenga definido para la notificación de ese tipo de evento.
- El escuchador, dentro de ese método, tendrá el código necesario para tratar ese evento.

¿Cómo se asigna un escuchador a una fuente de eventos?

```
<FuenteEvento>.add<EventoListener>(<Escuchador>)
```

Ejemplo Resumen:



◆ INTERFACES PARA LA NOTIFICACIÓN DE EVENTOS

- ✶ Interfaces que contienen métodos que serán invocados cuando se produzca un determinado tipo de evento.
- ✶ Cada interfaz está especializado en capturar un tipo de eventos determinado.
- ✶ Lista de algunas Interfaces y sus métodos

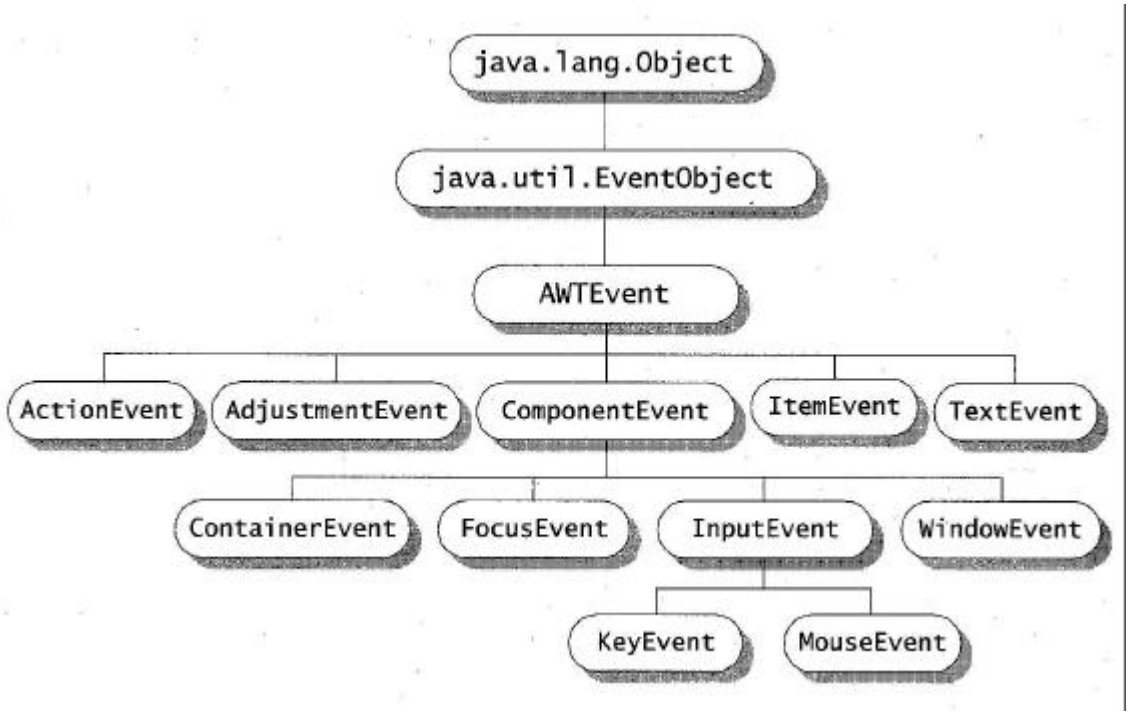
INTERFAZ	MÉTODOS
ActionListener	actionPerformed(ActionEvent)
FocusListener	focusGained(FocusEvent) focusLost(FocusEvent)
TextListener	textValueChanged(TextEvent)
KeyListener	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)
MouseListener	mouseClicked(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent)
MouseMotionListener	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener	windowActivated(WindowEvent) windowClosed(WindowEvent) windowClosing(WindowEvent) windowDeactivated(WindowEvent) windowDeiconified(WindowEvent) windowIconified(WindowEvent) windowOpened(WindowEvent)

  - ❑ Más interfaces en los apuntes y en el API

- ✶ Todos estos métodos reciben como parámetro un objeto “Evento” que posee información sobre el evento ocurrido.

◆ TIPOS DE EVENTOS

✶ Los objetos de tipo “Evento” están organizados en jerarquías de clases de eventos, de la siguiente forma:



- La clase padre de todos los eventos es EventObject y la de los eventos de naturaleza gráfica es AWTEvent
- Cuando ocurre un evento, los objetos de tipo “Evento” son pasados como parámetro en los métodos que posee el escuchador.
- Cada objeto de tipo “Evento” posee información sobre el evento concreto que se ha producido.

✶ Obtención de la información de un evento:

- Cada tipo de evento posee una serie de métodos que permiten consultar la información del evento ocurrido.
- Métodos de la clase EventObject (más en el API)
  - getSource()
    - ✓ Devuelve el objeto (fuente de eventos) sobre el que se ha producido el evento.
- Métodos de la clase ActionEvent (más en el API)

- getModifiers()
  - ✓ Devuelve un código que aporta información variada sobre el evento (por ejemplo, indica las combinaciones de teclas pulsadas: SHIFT+ALT)
- getActionCommand()
  - ✓ Devuelve un string que ayuda a determinar el tipo de acción a ejecutar ante el evento. Tiene múltiples usos.

❑ Métodos de la clase MouseEvent (más en el API)

- getModifiers()
  - ✓ Devuelve un código que aporta información variada sobre el evento (por ejemplo, indica el botón del ratón pulsado)
- getX() y getY()
  - ✓ Indica la coordenada X e Y donde se ha pulsado el ratón
- GetClickCount()
  - ✓ Indica el número de clicks seguidos que se han efectuado con el ratón

✶ Tipos de eventos generados por cada fuente de eventos

❑ Cada componente (fuente de eventos), en función de la acción que se realice sobre él, generará distintos tipos de eventos

❑ Ejemplo:

Componente	Evento	Hecho que lo genera
Button	ActionEvent	El usuario hace un clic sobre el botón.
Checkbox	ItemEvent	El usuario selecciona o deselecciona el Checkbox
List	ActionEvent	El usuario hace doble click sobre un elementos de la lista
	ItemEvent	El usuario selecciona o deselecciona un elemento de la lista

Component	MouseEvent	El usuario pulsa o suelta un botón del ratón, el cursor del ratón entra o sale o el usuario mueve o arrastra el ratón
	FocusEvent	El componente gana o pierde el foco
	KeyEvent	El usuario pulsa o suelta una tecla
TextField	ActionEvent	El usuario termina de editar el texto (hace un intro)
Window	WindowEvent	La ventana se abre, se cierra, se minimiza, se reestablece o se cierra.

◆ EJEMPLOS

✶ Ejemplo Sencillo

- Ventana con un botón y una etiqueta. Cuando se pulse el botón queremos que en la etiqueta se visualice la fecha y hora actual.

- Pasos:
  - Crear la ventana y todos sus componentes.
  - Crear un escuchador que sea capaz de atender a los eventos generados al pulsar un botón.
    - ✓ Evento generado: ActionEvent
    - ✓ Interfaz que posee los métodos para atender ese evento: ActionListener
  - Asignar ese escuchador al botón.
  - Incorporar el código a ejecutar cuando se pulse el botón.
- Código:

```
import javax.swing.JFrame;
import javax.swing.JButton;
```

```
import javax.swing.JLabel;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Date;

class EscuchadorBoton implements ActionListener
{
   JLabel etiqueta;

   public EscuchadorBoton(JLabel etiq)
   {
      this.etiqueta = etiq;
   }

   public void actionPerformed(ActionEvent e)
   {
      etiqueta.setText("Botón Pulsado: " + new Date());

   }
}

public class VentanaBoton extends JFrame
{
   JPanel panelBoton;
   JLabel etiqueta;
   JButton boton;

   public VentanaBoton()
   {
      etiqueta = new JLabel();
      panelBoton = new JPanel();
      boton = new JButton("Pulsa Aquí");

      panelBoton.add(boton);

      this.getContentPane().setLayout(new
                  BorderLayout());
      this.getContentPane().add(etiqueta,"North");
      this.getContentPane().add(panelBoton,"South");
```

```
      EscuchadorBoton escuchador = new
                  EscuchadorBoton(etiqueta);
      boton.addActionListener(escuchador);

      this.setSize(300,100);
      this.setTitle("Ejemplo Sencillo");
      this.show();
   }

   public static void main(String[] args)
   {
      new VentanaBoton();
   }
}
```

- Ejemplo Sencillo siendo el escuchador la propia ventana
  - A menudo, y puesto que cualquier objeto puede convertirse en un escuchador, en lugar de crear un objeto aparte para el escuchador se pone como escuchador de los eventos generados por los elementos de la ventana a la propia ventana.
  - En la asignatura usaremos este segundo enfoque.
  - Código:

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Date;

public class VentanaBotonEscuchador extends JFrame
                  implements ActionListener
{
   JPanel panelBoton;
   JLabel etiqueta;
   JButton boton;

   public VentanaBotonEscuchador()
```

```
    {
       etiqueta = new JLabel();
       panelBoton = new JPanel();
       boton = new JButton("Pulsa Aquí");

       panelBoton.add(boton);

       this.getContentPane().setLayout(new
                   BorderLayout());
       this.getContentPane().add(etiqueta,"North");
       this.getContentPane().add(panelBoton,"South");

       boton.addActionListener(this);

       this.setSize(300,100);
       this.setTitle("Ejemplo Sencillo");
       this.show();
    }

    public void actionPerformed(ActionEvent e)
    {
       etiqueta.setText("Botón Pulsado: " + new Date());
    }

    public static void main(String[] args)
    {
       new VentanaBoton();
    }
 }
```

- Ejemplo con varias fuentes de eventos (del mismo tipo)
  - Tenemos una ventana con tres botones y una etiqueta.

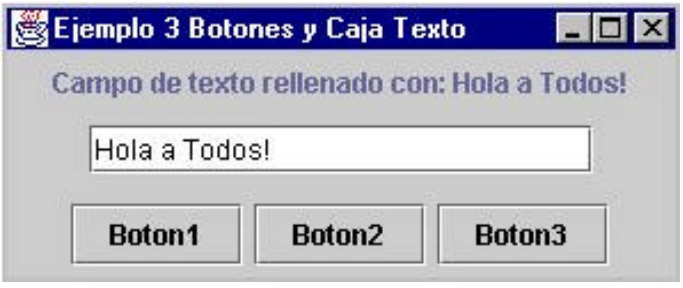
  - Si asignamos a los tres botones un mismo escuchador (la propia ventana), ¿Cómo podemos averiguar dentro del método actionPerformed( ) el botón que se ha pulsado?

- Solución: consultado la información que viene en el evento

```
import javax.swing.JFrame;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.util.Date;

public class VentanaBotones extends JFrame implements
                ActionListener
{
   JPanel panelBotones;
   JPanel panelEtiqueta;
   JLabel etiqueta;
   JButton boton1;
   JButton boton2;
   JButton boton3;

   public VentanaBotones()
   {
      etiqueta = new JLabel();
      panelBotones = new JPanel();
      panelEtiqueta = new JPanel();
      boton1 = new JButton("Boton1");
      boton2 = new JButton("Boton2");
      boton3 = new JButton("Boton3");

      panelEtiqueta.add(etiqueta);
      panelBotones.add(boton1);
      panelBotones.add(boton2);
      panelBotones.add(boton3);

      this.getContentPane().setLayout(new
                  BorderLayout());
      this.getContentPane().add(panelEtiqueta,"North");
      this.getContentPane().add(panelBotones,"South");

      boton1.addActionListener(this);
      boton2.addActionListener(this);
      boton3.addActionListener(this);
```

```
        this.setSize(300,100);
        this.setTitle("Ejemplo 3 Botones");
        this.show();
    }

    public void actionPerformed(ActionEvent e)
    {
        JButton botonPulsado = (JButton)e.getSource();
        if (botonPulsado == boton1)
            etiqueta.setText("Se ha pulsado el botón 1");

        if (botonPulsado == boton2)
            etiqueta.setText("Se ha pulsado el botón 2");

        if (botonPulsado == boton3)
            etiqueta.setText("Se ha pulsado el botón 3")
    }

    public static void main(String[] args)
    {
        new VentanaBotones();
    }
}
```

- Ejemplo con varias fuentes de eventos (de distinto tipo)
  - Tenemos una ventana con tres botones, una caja de texto y una etiqueta.



  - En este caso, ¿cómo averiguamos si el elemento sobre el que se produce el evento es un botón o la caja de texto?
  - Solución: consultado la información del evento (ídem anterior)

```
public class VentanaBotonesTexto extends JFrame
                    implements ActionListener
```

```
{
   ...
   JButton boton1;
   JButton boton2;
   JButton boton3;
   JTextField texto;

   public VentanaBotonesTexto()
   {
      ...
      boton1.addActionListener(this);
      boton2.addActionListener(this);
      boton3.addActionListener(this);
      texto.addActionListener(this);
      ...
   }

   public void actionPerformed(ActionEvent e)
   {
      Object elemento = e.getSource();
      if (elemento == boton1)
         etiqueta.setText("Se ha pulsado el botón 1");
      if (elemento == boton2)
         etiqueta.setText("Se ha pulsado el botón 2");
      if (elemento == boton3)
         etiqueta.setText("Se ha pulsado el botón 3");
      if (elemento == texto)
         etiqueta.setText("Campo de texto rellenado con:
                  " + texto.getText());
   }
   ...
}
```

- Si en algún momento necesitásemos conocer el tipo al que pertenece alguno de los objetos, podríamos hacer:

```
Object elemento = e.getSource();
if (elemento instanceof JButton)
{
   System.out.println("El elemento es un botón");
   JButton boton = (JButton)elemento;
}
if (elemento instanceof JTextField)
```

```
{
  System.out.println("El elemento es de texto");
  JTextField cajaTexto = (JTextField)elemento;
}
```

✶ Ejemplo Completo

- Aspecto de la ventana:

- En este ejemplo se capturan distintos tipos de eventos:
  - Eventos generados al pulsar botones
    - ✓ Evento: ActionEvent
    - ✓ Interfaz: ActionListener
    - ✓ Acción a realizar: cambiar el color del panel superior
  - Eventos generados al mover y pulsar el ratón sobre un panel
    - ✓ Evento: MouseEvent
    - ✓ Interfaz: MouseListener
    - ✓ Acción a realizar: Visualizar en una lista el tipo de evento que se ha producido

- Eventos generados al cerrar, minimizar, maximizar,… la ventana
  - ✓ Evento: WindowEvent
  - ✓ Interfaz: WindowListener
  - ✓ Acción a realizar: cerrar la ventana y terminar el programa.

- Código:

```
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JList;
import javax.swing.JButton;
import javax.swing.JLabel;
import javax.swing.JScrollPane;
import javax.swing.DefaultListModel;
import java.awt.Color;
import java.awt.GridLayout;
import java.awt.BorderLayout;
import java.awt.event.ActionEvent;
import java.awt.event.MouseEvent;
import java.awt.event.WindowEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseListener;
import java.awt.event.WindowListener;

public class VentanaEventos extends JFrame implements
                ActionListener, MouseListener,
                WindowListener
{
   JPanel panelSuperior;
   JPanel panelInferior;
   JPanel panelBotones;
   JButton botonAzul;
   JButton botonVerde;
   JButton botonRojo;
   JButton botonLimpiar;
   JButton botonSalir;
   JLabel etiqueta;

   JList listaEventos;
   DefaultListModel datosLista;
   JScrollPane scrollLista;
```

```
public VentanaEventos()
{
   panelSuperior = new JPanel();
   panelInferior = new JPanel();
   panelBotones = new JPanel();
   botonAzul = new JButton("Azul");
   botonVerde = new JButton("Verde");
   botonRojo = new JButton("Rojo");
   botonLimpiar = new JButton("Limpiar");
   botonSalir = new JButton("Salir");
   etiqueta = new JLabel("Panel que captura los
                  eventos del ratón");
   datosLista = new DefaultListModel();
   listaEventos = new JList(datosLista);
   scrollLista = new JScrollPane(listaEventos);

   panelBotones.add(botonAzul);
   panelBotones.add(botonVerde);
   panelBotones.add(botonRojo);
   panelBotones.add(botonLimpiar);
   panelBotones.add(botonSalir);

   panelSuperior.add(etiqueta);

   panelInferior.setLayout(new BorderLayout());
   panelInferior.add(scrollLista, "Center");
   panelInferior.add(panelBotones, "South");

   this.getContentPane().setLayout(new
               GridLayout(2,1));
   this.getContentPane().add(panelSuperior);
   this.getContentPane().add(panelInferior);

   // Asignamos escuchadores a todos los elementos
   // que puedan recibir eventos
   // Ponemos como escuchador de todos ellos a la
   // propia ventana
   botonAzul.addActionListener(this);
   botonVerde.addActionListener(this);
   botonRojo.addActionListener(this);
   botonLimpiar.addActionListener(this);
   botonSalir.addActionListener(this);
```

```
    panelSuperior.addMouseListener(this);
    this.addWindowListener(this);

    this.setTitle("Ventana con Eventos");
    this.setSize(500,400);
    this.show();
  }

  // Método del interfaz ActionListener
  public void actionPerformed(ActionEvent e)
  {
    JButton botonPulsado = (JButton)e.getSource();
    if (botonPulsado == botonAzul)
    {
      panelSuperior.setBackground(Color.blue);
    }
    if (botonPulsado == botonVerde)
    {
      panelSuperior.setBackground(Color.green);
    }
    if (botonPulsado == botonRojo)
    {
      panelSuperior.setBackground(Color.red);
    }
    if (botonPulsado == botonLimpiar)
    {
      datosLista.clear();
      panelSuperior.setBackground(Color.lightGray);
    }
    if (botonPulsado == botonSalir)
    {
      System.exit(0);
    }
  }

  // Los 5 métodos del interfaz MouseListener
  public void mouseClicked(MouseEvent e)
  {
    String linea = "Evento CLICK, ";
    if (e.getModifiers() == MouseEvent.BUTTON1_MASK)
    {
      linea = linea + "con BOTON IZQUIERDO ";
    }
```

```
    if (e.getModifiers() == MouseEvent.BUTTON2_MASK)
    {
      linea = linea + "con BOTON DEL CENTRO ";
    }
    if (e.getModifiers() == MouseEvent.BUTTON3_MASK)
    {
      linea = linea + "con BOTON DERECHO ";
    }
    linea = linea + "en coordenadas (" + e.getX() +
                "," + e.getY() + ")";
    datosLista.addElement(linea);
  }

  public void mouseEntered(MouseEvent e)
  {
    String linea = "Evento ENTRAR AL PANEL en
                coordenadas (" + e.getX() + "," +
                e.getY() + ")";
    datosLista.addElement(linea);
  }

  public void mouseExited(MouseEvent e)
  {
    String linea = "Evento SALIR DEL PANEL en
                coordenadas (" + e.getX() + "," +
                e.getY() + ")";
    datosLista.addElement(linea);
  }

  public void mousePressed(MouseEvent e)
  {
    // Este método se invoca cuando se pulsa el botón
    // del ratón
  }

  public void mouseReleased(MouseEvent e)
  {
    // Este método se invoca cuando se libera el
    // botón del ratón
  }

  //Los 7 métodos del interfaz WindowListener
  public void windowClosing(WindowEvent e)
```

```
    {
      System.exit(0);
    }

    public void windowClosed(WindowEvent e){ }
    public void windowIconified(WindowEvent e){ }
    public void windowOpened(WindowEvent e){ }
    public void windowActivated(WindowEvent e){ }
    public void windowDeactivated(WindowEvent e){ }
    public void windowDeiconified(WindowEvent e){ }

    public static void main(String[] args)
    {
      new VentanaEventos();
    }
}
```