Manejo de eventos AWT

En las aplicaciones en modo consola, nuestro método `main` determina el orden en el que se ejecutan las operaciones de nuestro programa.

En las aplicaciones con interfaces gráficas de usuario, el orden en el que se ejecutan las operaciones dependerá de las acciones del usuario. Nosotros sólo hemos de preocuparnos de qué acciones ha de realizar nuestra aplicación, cuándo han de realizarse y definir los correspondientes manejadores de eventos, que serán invocados automáticamente cuando sus eventos asociados se produzcan.

El modelo de eventos de Java

Cada lenguaje de programación tiene su propio modelo de eventos:

- En Visual Basic, cada componente genera eventos específicos y nosotros redefinimos la respuesta del componente a cada evento (modelo simple pero bastante inflexible).
- En C, todos los eventos se sitúan en una cola: hemos de ir leyendo los eventos de esta cola y decidir qué acción realizar en función del evento extraído de la cola (tedioso de programar).
- En Java, se definen clases auxiliares (*event listeners*) que pueden recibir eventos de tipos específicos (p.ej. el clic del botón del ratón). Estas clases se asocian luego a componentes específicos.
  - A diferencia de VB, el manejador no viene predeterminado (mayor flexibilidad).
  - A diferencia de C, cada tipo de evento se puede tratar por separado.

En términos de objetos y métodos,
el manejo de eventos en Java funciona de la siguiente forma:

- Los manejadores de eventos (*event listeners*) se registran en las distintas fuente de eventos (*event source*).
- Una fuente de eventos (el ratón, un botón, una ventana…) envía objetos de tipo `EventObject` a todos los manejadores registrados cuando se produce un nuevo evento.
- Cada manejador de eventos utiliza la información recibida a través del objeto de tipo `EventObject` para realizar las acciones que estime adecuadas.

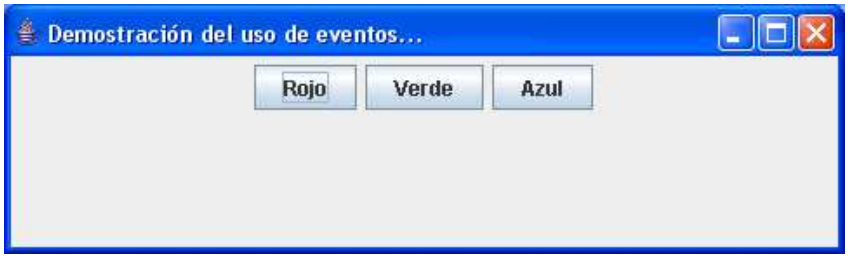
En Java, los eventos están organizados en una jerarquía de clases dentro del paquete java.awt.event

- La clase `java.util.EventObject` es la clase base de todos los eventos en Java.
- La subclase `java.awt.AWTEvent` es la clase base de todos los eventos que se utilizan en la construcción de GUIs.
- Cada tipo de evento `XxxEvent` tiene asociada una interfaz `XxxListener` que es la que nos permite definir manejadores de eventos.
- Para simplificar la implementación de algunos manejadores de eventos, el paquete `java.awt.event` incluye clases base `XxxAdapter` que implementan las interfaces `XxxListener`

Por ejemplo, para gestionar los eventos del ratón usaremos `MouseAdapter`, para controlar los eventos de una ventana recurriremos a `WindowAdapter` y para especificar lo que hará nuestra aplicación cuando se pulse un botón emplearemos `ActionListener`.

Ejemplo: Botones & ActionListener

Una aplicación que nos permita cambiar el color del fondo




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

class EventFrame extends JFrame
{
  public EventFrame()
  {
    setTitle("Demostración del uso de eventos...");
    setSize(600,440);
    addWindowListener(new MainWindowListener());

    Container contenido = getContentPane();
    contenido.add(new ButtonPanel());
  }
}

class MainWindowListener extends WindowAdapter
{
  public void windowClosing(WindowEvent e)
  {
    System.exit(0);
  }
}

public class EventTest
{
  public static void main(String[] args)
  {
    JFrame frame = new EventFrame();
    frame.setVisible(true);
  }
}
```

```
class ButtonPanel extends JPanel
                  implements ActionListener
{
  private JButton redButton;
  private JButton greenButton;
  private JButton blueButton;

  public ButtonPanel()
  {
    redButton = new JButton("Rojo");
    greenButton = new JButton("Verde");
    blueButton = new JButton("Azul");

    this.add(redButton);
    this.add(greenButton);
    this.add(blueButton);

    redButton.addActionListener(this);
    greenButton.addActionListener(this);
    blueButton.addActionListener(this);
  }

  public void actionPerformed (ActionEvent event)
  {
    Object source = event.getSource();
    Color  color = getBackground();

    if (source == redButton)
       color = Color.RED;
    else if (source == greenButton)
       color = Color.GREEN;
    else if (source == blueButton)
       color = Color.BLUE;

    setBackground(color);
    repaint();
  }
}
```

Ejemplo: El ratón & MouseAdapter

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

public class MouseTest
{
  public static void main(String[] args)
  {
    JFrame frame = new MouseFrame();
    frame.setVisible(true);
  }
}

class MouseFrame extends JFrame
{
  public JLabel info;

  public MouseFrame()
  {
    setTitle("Demostración del uso de eventos...");
    setSize(600,440);
    addWindowListener(new MainWindowListener());

    info = new JLabel("Juegue con el ratón");

    Container contenido = getContentPane();
    contenido.add(info, BorderLayout.SOUTH);

    this.addMouseListener
        (new MiMouseAdapter(this));
    this.addMouseMotionListener
        (new MiMouseMotionAdapter(this));
  }
}

class MainWindowListener extends WindowAdapter
{
  public void windowClosing(WindowEvent e)
  {
    System.exit(0);
  }
}
```


```
class MiMouseAdapter extends MouseAdapter
{
  private MouseFrame frame;

  public MiMouseAdapter (MouseFrame frame)
    { this.frame = frame; }

  public void mouseClicked (MouseEvent event)
  {
    frame.info.setText ("Ratón pulsado en ("
         + event.getX() + "," + event.getY() + ")");
  }

  public void mouseEntered (MouseEvent event)
  {
    frame.info.setText ("El ratón entró en la ventana");
  }

  public void mouseExited (MouseEvent event)
  {
    frame.info.setText ("El ratón salió de la ventana");
  }
}

class MiMouseMotionAdapter extends MouseMotionAdapter
{
  private MouseFrame frame;

  public MiMouseMotionAdapter (MouseFrame frame)
    { this.frame = frame; }

  public void mouseMoved (MouseEvent event)
  {
    frame.info.setText ("Ratón en ("
         + event.getX() + "," + event.getY() + ")");
  }
}
```

Ejemplo: Uso de otros componentes Swing


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

class ListFrame extends JFrame
                implements ActionListener
{
  JPanel    tools;
  JComboBox font;
  JComboBox fontsize;
  JCheckBox bold;
  JCheckBox italic;
  FontPanel panel;

  public ListFrame()
  {
    setTitle("Demostración del uso de eventos...");
    setSize(600,150);
    addWindowListener(new MainWindowListener());

    // Barra de herramientas

    tools = new JPanel();

    tools.add(new JLabel("Fuente "));

    font = new JComboBox();
    font.setEditable(false);
    font.addItem("Serif");
    font.addItem("SansSerif");
    font.addItem("Monospaced");
    font.addActionListener(this);
    tools.add(font);
```

```
    tools.add(new JLabel("Tamaño "));

    fontsize = new JComboBox();
    fontsize.setEditable(true);
    fontsize.addItem("12");
    fontsize.addItem("16");
    fontsize.addItem("24");
    fontsize.addActionListener(this);
    tools.add(fontsize);

    bold = new JCheckBox("Negrita");
    bold.addActionListener(this);
    tools.add(bold);

    italic= new JCheckBox("Cursiva");
    italic.addActionListener(this);
    tools.add(italic);

    getContentPane().add(tools, "South");

    panel = new FontPanel();
    getContentPane().add(panel, "Center");
  }

  public void actionPerformed(ActionEvent event)
  {
    String tipo   = (String)font.getSelectedItem();
    String str    = (String)fontsize.getSelectedItem();
    int    dim    = Integer.valueOf(str).intValue();
    int    estilo = (bold.isSelected()? Font.BOLD: 0)
                  + (italic.isSelected()? Font.ITALIC: 0);

    panel.setFont(tipo, estilo, dim);
    repaint();
  }
}

class MainWindowListener extends WindowAdapter
{
  public void windowClosing (WindowEvent e)
  {
    System.exit(0);
  }
}
```

```
class FontPanel extends JPanel
{
  private Font font;

  public void setFont (String font, int style, int size)
  {
    this.font = new Font ( font, style, size);
  }

  public void paintComponent (Graphics g)
  {
    if (font!=null)
       g.setFont(font);

    g.drawString("Texto del panel", 50,50);
  }
}


public class JListTest
{
  public static void main(String[] args)
  {
    JFrame frame = new ListFrame();
    frame.setVisible(true);
  }
}
```