Tema 7: Interfaces gráficas de usuario III

Manejo de eventos

1. Eventos

Swing maneja la interacción del usuario con las interfaces gráficas mediante una serie de interfaces (en el otro sentido) llamadas *Event Listeners*.

Se produce un evento cada vez que el usuario interacciona con una ventana (pulsa un botón del ratón, pasa el puntero del ratón por encima de una imagen, pulsa una tecla, etc.).

Cada tipo de evento lleva asociado un manejador de eventos distinto. Por ejemplo:

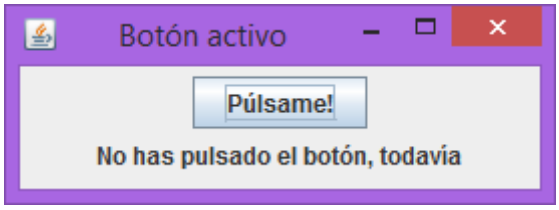
- **ActionListener**: Se ha producido la acción por defecto sobre un componente. Por ejemplo, en el caso de un botón sería pulsarlo.
- **KeyListener**: Se ha pulsado una tecla.
- **MouseListener**: Se ha hecho clic con el ratón.

2. Eventos de acción: ActionListener

Un evento de acción corresponde a la acción predeterminada sobre un componente. Vamos a ver dos ejemplos con botones, cuya acción predeterminada es "pulsar".

2.1 Ejemplo: Pulsar un botón

Vamos a crear una aplicación con una ventana con este estilo:



El código hasta ahora es:

```
public class BotonActivo extends JFrame implements ActionListener{

	private JButton btnBoton;
	private JLabel lblLetrero;

	public BotonActivo(){
		super("Botón activo");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(275,100);

		setLayout(new FlowLayout());

		btnBoton = new JButton("Púlsame!");
		lblLetrero = new JLabel("No has pulsado el botón, todavía");

		add(btnBoton);
		add(lblLetrero);

		setVisible(true);
	}

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

Siempre que queramos manejar eventos tendremos que importar el siguiente paquete:

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

Queremos que al pulsar el botón cambie el letrero que tiene debajo. Pulsar es el evento por defecto de un botón. Para manejar este evento usaremos una interfaz *ActionListener*.

En primer lugar hay que decidir qué clase va a escuchar este evento (es decir, la clase que contiene el código que se ejecutará al pulsar el botón). En este caso usaremos nuestra propia clase *BotonActivo*. Por tanto indicamos que se añada un manejador de eventos al botón y que éste sea la propia clase:

```
btnBoton.addActionListener(this);
```

Ahora tenemos que hacer que nuestra clase esté preparada para gestionar este evento: haremos que implemente la interfaz ActionListener.

```
public class BotonActivo extends JFrame implements ActionListener{
```

Ya sabemos que implementar una interface consiste en implementar todos sus métodos. La interface ActionListener sólo tiene un método, llamado *actionPerformed( )*. Por tanto, añadimos este método a nuestra clase:

```
@Override
public void actionPerformed(ActionEvent e) {
	// Código que se ejecutará al pulsarse el botón

}
```

El objeto *ActionEvent e* contiene información sobre el objeto que ha desencadenado el evento, aunque en este caso no la usaremos.

Dentro haremos que la etiqueta cambie de texto:

```
@Override
public void actionPerformed(ActionEvent e) {
	lblLetrero.setText("Has pulsado el botón, por fin!");
}
```

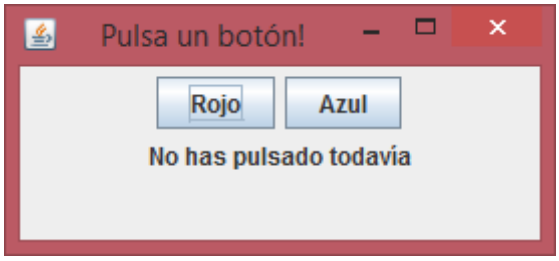
El programa completo:

```
public class BotonActivo extends JFrame implements ActionListener {

	private JButton btnBoton;
	private JLabel lblLetrero;

	public BotonActivo() {
		super("Botón activo");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(275, 100);

		setLayout(new FlowLayout());

		btnBoton = new JButton("Púlsame!");
		lblLetrero = new JLabel("No has pulsado el botón, todavía");

		btnBoton.addActionListener(this);

		add(btnBoton);
		add(lblLetrero);

		setVisible(true);
	}

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

	@Override
	public void actionPerformed(ActionEvent e) {
		// Código que se ejecutará al pulsarse el botón
```

```
            lblLetrero.setText("Has pulsado el botón, por fin!");
        }
}
```

2.2. Ejemplo: ¿Qué botón se ha pulsado?

En este caso haremos una ventana con dos botones y una etiqueta. La etiqueta mostrará un mensaje diferente en función del botón que se haya pulsado. Esto lo conseguiremos con el objeto ActionEvent que recibe el manejador del evento.



El código hasta ahora es:

```
public class DosBotones extends JFrame {

	private JButton btnRojo;
	private JButton btnAzul;
	private JLabel lblTexto;

	public DosBotones() {
		super("Pulsa un botón!");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(275, 125);

		setLayout(new FlowLayout());

		btnRojo = new JButton("Rojo");
		btnAzul = new JButton("Azul");
		lblTexto = new JLabel("No has pulsado todavía");

		add(btnRojo);
		add(btnAzul);
		add(lblTexto);

		setVisible(true);
	}

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

Queremos capturar los eventos producidos por los dos botones, así que añadimos las siguientes líneas:

```
btnRojo.addActionListener(this);
btnAzul.addActionListener(this);
```

Y, para que nuestra clase pueda encargarse de estos eventos, hacemos que implemente la interfaz ActionListener:

```
public class DosBotones extends JFrame implements ActionListener{
```

Y le añadimos el único método de la interfaz ActionListener:

```
@Override
public void actionPerformed(ActionEvent arg0) {
	Object boton = arg0.getSource();
	if (boton == btnRojo)
		lblTexto.setText("Has pulsado el botón rojo");
	else if (boton == btnAzul)
		lblTexto.setText("Has pulsado el botón azul");
}
```

Hemos usado el método *getSource( )* de ActionEvent para obtener el objeto (botón) que generó el evento (pulsación).

El código completo queda:

```
public class DosBotones extends JFrame implements ActionListener {

	private JButton btnRojo;
	private JButton btnAzul;
	private JLabel lblTexto;

	public DosBotones() {
		super("Pulsa un botón!");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(275, 125);

		setLayout(new FlowLayout());

		btnRojo = new JButton("Rojo");
		btnAzul = new JButton("Azul");
		lblTexto = new JLabel("No has pulsado todavía");

		btnRojo.addActionListener(this);
		btnAzul.addActionListener(this);

		add(btnRojo);
		add(btnAzul);
		add(lblTexto);
```

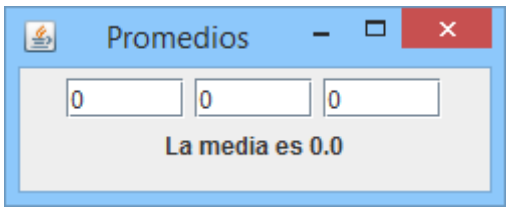
```
		setVisible(true);
	}

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

	@Override
	public void actionPerformed(ActionEvent arg0) {
		Object boton = arg0.getSource();
		if (boton == btnRojo)
			lblTexto.setText("Has pulsado el botón rojo");
		else if (boton == btnAzul)
			lblTexto.setText("Has pulsado el botón azul");
	}
}
```

3. Eventos de foco: FocusListener

Se dice que un componente tiene el foco cuando está seleccionado. La interfaz FocusListener tiene dos métodos:

- Foco recibido: focusGained( )
- Foco perdido: focusLost( )

Ejemplo: Programa que muestra una serie de casillas de texto para que introduzcamos números. Cada vez que cambiamos el cursor de una casilla a otra, se muestra la media aritmética de todos los valores.




```
public class MediaAritmetica extends JFrame implements FocusListener {

	private JTextField txtNum1 = new JTextField("0",5);
	private JTextField txtNum2 = new JTextField("0",5);
	private JTextField txtNum3 = new JTextField("0",5);
	private JLabel lblMedia = new JLabel("La media es 0");

	public MediaAritmetica() {
		super("Promedios");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(250, 100);

		setLayout(new FlowLayout());
```

```
		txtNum1.addFocusListener(this);
		txtNum2.addFocusListener(this);
		txtNum3.addFocusListener(this);

		add(txtNum1);
		add(txtNum2);
		add(txtNum3);
		add(lblMedia);

		setVisible(true);
	}

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

	@Override
	public void focusGained(FocusEvent arg0) {
		try {
			double promedio = Double.parseDouble(txtNum1.getText())
					+ Double.parseDouble(txtNum2.getText())
					+ Double.parseDouble(txtNum3.getText());
			promedio /=3;
			lblMedia.setText("La media es " + promedio);
		} catch (NumberFormatException nfe) {
			lblMedia.setText("Error de entrada");
		}
	}

	@Override
	public void focusLost(FocusEvent arg0) {
		// Dejado en blanco a propósito. No hace falta utilizar ambos métodos de
		// la interfaz.
	}
}
```

4. Eventos de selección: ItemListener

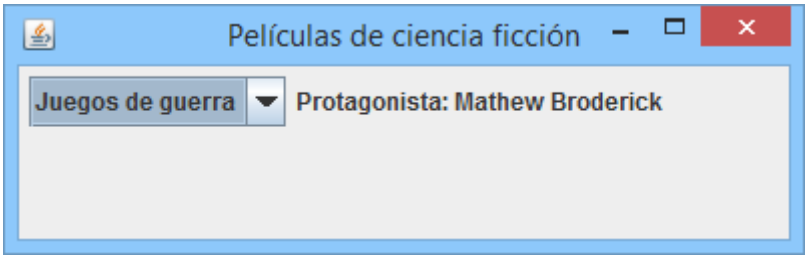
Estos eventos ocurren al cambiar la opción seleccionada en un JComboBox. Son parecidos a los ActionEvent sólo que estos últimos se generan cada vez que se hace clic en el JComboBox, aunque no cambie la selección.

El único método de la interfaz ItemListener es:

- `public void itemStateChanged(ItemEvent arg0)`

Una manera de averiguar qué opción se ha seleccionado del JComboBox, por ejemplo, es utilizar el método *getSelectedIndex( )* del mismo.

Ejemplo: Mediante un combobox elegimos una película y se nos muestra el nombre del protagonista.




```
public class Peliculas extends JFrame implements ItemListener {

	private JComboBox cmbPeliculas;
	private JLabel lblActores;
	private String[] actores = { "Michael J. Fox", "Mathew Broderick", "Jeff
Bridges",
			"Peter Weller", "Arnold Schwarzenneger" };

	public Peliculas() {
		super("Películas de ciencia ficción");
		setSize(400, 125);
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

		setLayout(new FlowLayout(FlowLayout.LEFT));

		String[] peliculas = { "Regreso al futuro", "Juegos de guerra", "Tron",
				"Robocop", "Terminator" };

		cmbPeliculas = new JComboBox(peliculas);
		cmbPeliculas.addItemListener(this);

		lblActores = new JLabel("Selecciona una película.");

		add(cmbPeliculas);
		add(lblActores);

		setVisible(true);
	}

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

	@Override
	public void itemStateChanged(ItemEvent arg0) {
		lblActores.setText("Protagonista: "
				+ actores[cmbPeliculas.getSelectedIndex()]);
	}
}
```

5. Eventos de teclado: KeyListener

Estos eventos ocurren cuando se pulsa una tecla. La interfaz KeyListener dispone de tres métodos:

- `void keyTyped(KeyEvent arg0): Ocurre al pulsar una tecla con representación UNICODE. No ocurre si se trata de una tecla de acción (F1, F2, ...) o de modificación (SHIFT, CTRL, ...).`
- `void keyPressed(KeyEvent arg0): Ocurre al pulsar una tecla.`
- `void keyReleased(KeyEvent arg0): Ocurre al soltar una tecla que estaba pulsada.`

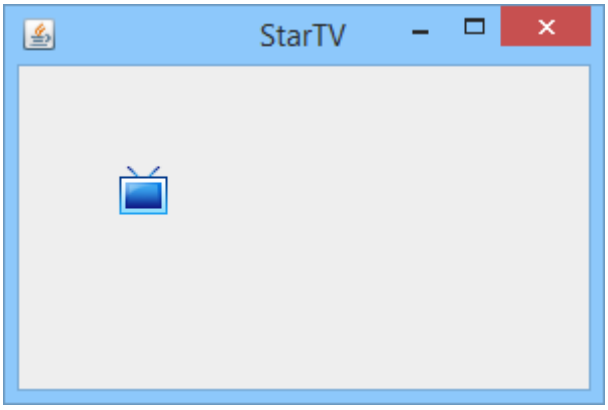
Cuando se produzca un evento de teclado nos pueden ser útiles los siguientes métodos para saber qué tecla se ha pulsado:

- `arg0.getKeyChar(): Devuelve un carácter.`
- `arg0.getKeyCode(): Devuelve un entero que representa la tecla pulsada.`

Por ejemplo, para comprobar si se ha pulsado la tecla 'w' podemos usar estas dos construcciones:

```
if (arg0.getKeyChar() == 'w') {

}

if (arg0.getKeyCode() == KeyEvent.VK_W){

}
```

Ejemplo: Vamos a dibujar en un JFrame una etiqueta con un icono, y haremos que se mueva al pulsar las teclas a, w, s, d.



Para el movimiento usaremos el método:

- `lblMovil.setBounds(x, y, 24, 24);`

Que coloca la etiqueta lblMovil en las coordenadas (x, y) con un tamaño 24px x 24px (elegido porque es el del icono). Además, para que el método funcione, deberemos establecer *null* como layout de nuestra ventana.

Código completo:

```
public class Nave extends JFrame implements KeyListener {

	private int x = 50, y = 50; // Coordenadas de la imagen.
	private JLabel lblMovil; // Etiqueta que contiene la imagen.

	public Nave() {
		super("StarTV");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 200);

		setLayout(null);

		addKeyListener(this);

		ImageIcon imgMovil = new ImageIcon("iconos/toolbar/67.png");
		JButton btnMovil = new JButton(imgMovil);
		lblMovil = new JLabel(imgMovil);

		lblMovil.setBounds(x, y, 24, 24);
		add(lblMovil);

		setVisible(true);
	}

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

	@Override
	public void keyPressed(KeyEvent arg0) {

		// Cuando se pulse una de las teclas de dirección moveremos 5 px la
		// imagen en la dirección que toque.
		if (arg0.getKeyChar() == 'w') {
			y -= 5;
		}

		if (arg0.getKeyChar() == 's') {
			y += 5;
		}

		if (arg0.getKeyChar() == 'a') {
			x -= 5;
		}

		if (arg0.getKeyChar() == 'd') {
			x += 5;
```

```
			}

			lblMovil.setBounds(x, y, 24, 24);
		}

		@Override
		public void keyReleased(KeyEvent arg0) {
			// No utilizado.
		}

		@Override
		public void keyTyped(KeyEvent arg0) {
			// No utilizado.
		}
}
```

6. Eventos de ratón: MouseListener

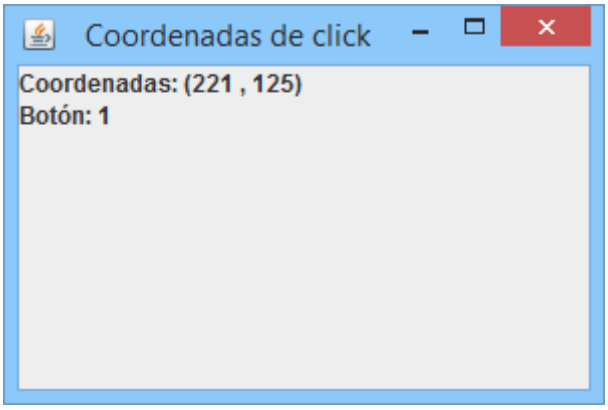
Estos eventos ocurren cuando se pulsa un botón del ratón o se desplaza el puntero por encima de un componente. La interfaz MouseListener dispone de cinco métodos:

- `public void mouseClicked(MouseEvent arg0)`: Invocado cuando el botón del ratón ha sido clicado (pulsado y soltado).
- `public void mouseEntered(MouseEvent arg0)`: Invocado cuando el puntero del ratón entra en un componente.
- `public void mouseExited(MouseEvent arg0)`: Invocado cuando el ratón sale de un componente.
- `public void mousePressed(MouseEvent arg0)`: Invocado cuando el botón del ratón se pulsa sobre un componente.
- `public void mouseReleased(MouseEvent arg0)`: Invocado cuando el botón del ratón ha sido soltado en un componente.

Cuando se produzca un evento de ratón nos pueden ser útiles los siguientes métodos:

- `arg0.getX()`: Devuelve la coordenada X del punto donde se ha hecho clic.
- `arg0.getY()`: Devuelve la coordenada Y del punto donde se ha hecho clic.
- `arg0.getButton()`: Devuelve un número entero que indica cuál de los botones se ha pulsado.

Ejemplo: Ventana que mediante dos etiquetas indica dónde se ha hecho clic y con qué botón.




```
public class Click extends JFrame implements MouseListener {

	private JLabel lblCoordenadas;
	private JLabel lblBotones;

	public Click() {
		super("Coordenadas de click");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 200);
		setVisible(true);

		setLayout(new BoxLayout(this.getContentPane(), BoxLayout.Y_AXIS));

		lblCoordenadas = new JLabel("Coordenadas:");
		lblBotones = new JLabel("Botón:");

		add(lblCoordenadas);
		add(lblBotones);

		addMouseListener(this);
	}

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

	@Override
	public void mouseClicked(MouseEvent arg0) {
		int x, y;
		x = arg0.getX();
		y = arg0.getY();

		lblCoordenadas.setText("Coordenadas: (" + x + " , " + y + ")");
		lblBotones.setText("Botón: " + arg0.getButton());
	}

	@Override
	public void mouseEntered(MouseEvent arg0) {
		// No utilizado
	}
```

```
	@Override
	public void mouseExited(MouseEvent arg0) {
		// No utilizado
	}

	@Override
	public void mousePressed(MouseEvent arg0) {
		// No utilizado
	}

	@Override
	public void mouseReleased(MouseEvent arg0) {
		// No utilizado
	}
}
```

7. Adaptadores de clase

Los adaptadores de clase nos permiten manejar eventos sin tener que implementar todos los métodos de las interfaces. Son clases que ya implementan todos los métodos de una interface (vacíos) y nosotros sólo tenemos que sobreescribir los que nos interesen y hacer que nuestra clase herede de una de estas clases.

- FocusAdapter
- KeyAdapter
- MouseAdapter
- Etc.

Ejemplo 1: Programa que muestra en una etiqueta las teclas que se van pulsando. Utiliza como manejador de eventos una clase interna anónima.

```
public class AdaptadorClase extends JFrame {

	private JLabel lblTexto;

	public AdaptadorClase() {
		super("Letras");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 200);

		setLayout(new FlowLayout());

		lblTexto = new JLabel("Ve pulsando teclas");
		add(lblTexto);

		addKeyListener(new KeyAdapter() {
			@Override
			public void keyTyped(KeyEvent evento) {
				lblTexto.setText("" + evento.getKeyChar());
			}
```

```
		});

		setVisible(true);
	}

	public static void main(String[] args) {
		new AdaptadorClase2();
	}
}
```

Ejemplo 2: Programa que muestra en una etiqueta las teclas que se van pulsando. Utiliza como manejador de eventos una clase interna (no anónima). La clase MonitorTeclas también se podría haber creado de la manera habitual, en otro archivo .java.

```
public class AdaptadorClase2 extends JFrame {

	private JLabel lblTexto;

	public AdaptadorClase2() {
		super("Letras");
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		setSize(300, 200);

		setLayout(new FlowLayout());

		lblTexto = new JLabel("Ve pulsando teclas");
		add(lblTexto);

		MonitorTeclas monitor = new MonitorTeclas(this);
		addKeyListener(monitor);

		setVisible(true);
	}

	public static void main(String[] args) {
		new AdaptadorClase2();
	}
}

class MonitorTeclas extends KeyAdapter {

	private AdaptadorClase2 ventana;

	MonitorTeclas(AdaptadorClase2 ventana) {
		this.ventana = ventana;
	}

	@Override
	public void keyTyped(KeyEvent evento) {
		// String texto = ventana.lblTexto.getText();
		ventana.lblTexto.setText("" + evento.getKeyChar());
	}
}
```