Programación en Java

Tema 6. Interfaces gráficas de usuario (Parte II – Componentes y eventos)

Luis Rodríguez Baena

Universidad Pontificia de Salamanca (campus Madrid)
Facultad de Informática

Etiquetas. JLabel

- Componentes de texto fijo.
  - Su contenido sólo se puede cambiar mediante código.

Constructores	
JLabel()	Crea una etiqueta sin texto
Label(String mensaje)	Crea una etiqueta con un mensaje determinado alineada a la izquierda
Label(String mensaje, int alin)	Crea una etiqueta con un mensaje alineada según el segundo parámetro. alin puede tomar los valores SwingConstants.RIGHT, SwingConstants.LEFT y SwingConstants.CENTER
Algunos métodos	
getText()	Devuelve el texto de la etiqueta
setText(String mensaje)	Establece el texto de la etiqueta

Campos de texto. JTextField

- Componentes de texto modificables por el usuario.
  - Desciende de la clase JTextComponent.

Constructores	
JTextField()	Crea un campo de texto vacío y de un ancho de 0 columnas
JTextField(int cols)	Crea un campo de texto vacío y con un ancho de cols columnas.
JTextField(String texto, int cols)	Crea una etiqueta con un texto inicial y un ancho de col columnas
Algunos métodos	
getText()	Devuelve el texto del campo
setText(String mensaje)	Establece el texto del campo
setEditable(boolean ed)	Establece si el texto puede o no ser editado por el usuario.
setHorizontalAlignment(int al)	Establece la alineación del texto. Puede tomar los valores valores SwingConstants.RIGHT, SwingConstants.LEFT, SwingConstants.CENTER, SwingConstants.LEADING (el texto se rellena de izquierda a derecha) o SwingConstants.TRAILING (el texto se rellena de derecha a izquierda)

Casillas de verificación. JCheckBox

- Casillas que pueden estar o no seleccionadas.
  - Puede haber más de una seleccionada por grupo.

Constructores	
JCheckBox()	Crea un casilla sin texto
JTextBox(String etiqueta)	Crea una casilla de verificación con una etiqueta inicial
JCkeckBox(String etiqueta, boolean selecionado)	Crea una casilla de verificación con una etiqueta. Si seleccionado es true aparece seleccionada.
Algunos métodos	
isSelected()	Devuelve true si está seleccionada.
setSelected(boolean estado)	Establece el estado de la casilla.

Botones de radio. JRadioButton (I)

- Sólo puede haber uno seleccionado por grupo.
  - Para utilizarlo es necesario crear un grupo de botones (clase `ButtonGroup`) y añadir el botón de radio mediante el método `add`.

```
…
ButtonGroup grupo = new ButtonGroup();
JRadioButton opc1 = new JRadioButton("Opción 1");
JRadioButton opc2 = new JRadioButton("Opción 2",true);
grupo.add(opc1);
grupo.add(opc2);
setContentPane().add(opc1);
setContentPane().add(opc2);
…
```

Botones de radio. JRadioButton (II)

Constructores (de JRadioButton)	
JRadioButton()	Crea un botón sin texto
JRadioButton(String etiqueta)	Crea un botón de radio con una etiqueta inicial
JRadioButton(String etiqueta, boolean selecionado)	Crea un botón de radio con una etiqueta. Si seleccionado es true aparece seleccionada.
Algunos métodos	
isSelected()	Devuelve true si está seleccionado.
setSelected(boolean estado)	Establece el estado del botón

Botones. JButton

Constructores	
JButton()	Crea un botón sin texto
JButton(String etiqueta)	Crea un botón con una etiqueta inicial

- Iconos.
  - Los botones, botones de radio, casillas de verificación y etiquetas admiten en sus constructores un argumento para incluir un icono.
  - El icono puede ser cualquier archivo GIF.
  - Es necesario crear un instancia de la clase `ImageIcon` y hacer referencia a ella en el constructor (`new ImageIcon(String ruta)`).

```
ImageIcon zipIcon = new ImageIcon("zipIcon.gif");
ImageIcon folderIcon = new ImageIcon("folder.gif");
ImageIcon pdfIcon = new ImageIcon("pdfIcon.gif");
JButton btn = new JButton(pdfIcon);
JCheckBox chkIcon = new JCheckBox("Zip",zipIcon); //Icono sin seleccionar
chkIcon.setSelectedIcon(folderIcon); //Icono seleccionado
JLabel lbl = new JLabel("Texto",zipIcon,SwingConstants.LEFT);
```

Cuadros combinados. JComboBox

- Requiere la creación de un objeto y rellenar sus elementos.
  - Su constructor más simple es un constructor sin argumentos.

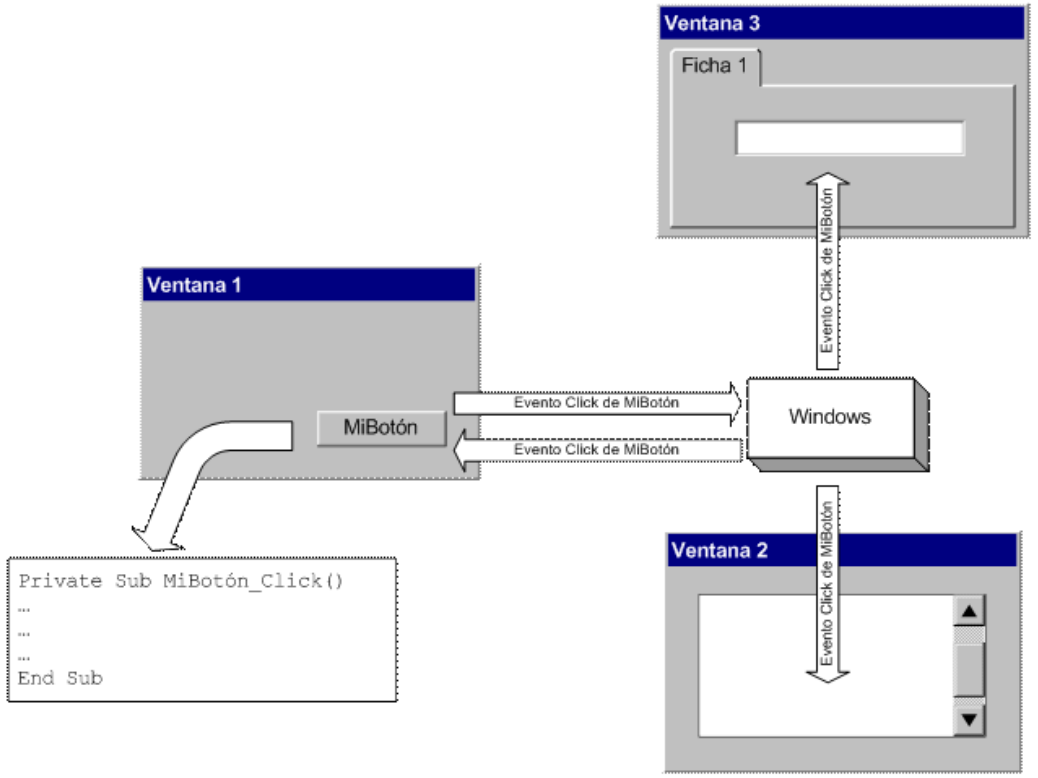
Algunos métodos	
setEditable(boolean opc)	Si opc se establece a true, el usuario puede introducir los valores.
insertItemAt(Object obj, int índice)	Introduce un objeto en la posición indicada que no debe ser mayor que el número de elementos menos 1
removeItem(Object obj)	Elimina el objeto indicado
removeItemAt(int indice)	Elimina el elemento indicado
removeAllItems()	Elimina todos los elementos
getSelectedItem()	Devuelve el elemento seleccionado.
getSelectedIndex()	Devuelve el índice del elemento seleccionado (-1 si el combo es editable y el usuario ha introducido un valor)

Gestión de eventos (I)

- Evento: suceso que ocurre en un sistema.
  - Los entornos GUI siempre están monitorizando los sucesos.
  - Cuando se produce uno el entorno informa a los programas que se están ejecutando.
    - ✓ Los programas deciden si desean hacer algo en respuesta al mensaje.
- En un entorno como Visual Basic cada componente ya tiene asociados una serie de eventos.
- En Java, es necesario registrar que eventos queremos que "escuche" cada componente.

Gestión de eventos (II)

- Un evento en Visual Basic.



Modelo de eventos AWT

- AWT controla la forma en que los eventos se dirigen desde los "orígenes de eventos" hasta los "oyentes de eventos".
- Los orígenes tienen métodos que permiten asociarles "oyentes" para monitorizar los eventos.
  - Los oyentes incluyen las acciones asociadas a los eventos que se registraron.
- Para la gestión de eventos será necesario:
  - Implementar un objeto "oyente"
    - ✓ Será alguna interfaz `Listener`.
  - Registrar ese objeto en el componente (*objetoOrigen*.add*Evento*Listener(*objetoOyenteEvento*).
  - Implementar los métodos de la interfaz `Listener`.

Un ejemplo (I)

```
1.  import javax.swing.*;
2.  import java.awt.*;
3.  import java.awt.event.*;
4.  public class EjemploEventos{
5.      public static void main(String args[]){
6.              FrameBotones frm = new FrameBotones();
7.              frm.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
8.              frm.show();
9.      }
10. }
11. class FrameBotones extends JFrame{
12.     JButton btnAzul = new JButton("Azul");
13.     JButton btnAmarillo = new JButton("Amarillo");
14.     JButton btnRojo = new JButton("Rojo");
15.
16.     FrameBotones(){
17.             setTitle("Ejemplo de eventos");
18.             getContentPane().setLayout(new FlowLayout());
19.             getContentPane().add(btnAzul);
20.             getContentPane().add(btnAmarillo);
21.             getContentPane().add(btnRojo);
22.             pack();
```

Un ejemplo (II)

```
23.             btnAzul.addActionListener(new MiOyente());
24.             btnAmarillo.addActionListener(new MiOyente());
25.             btnRojo.addActionListener(new MiOyente());
26.     }
27.
28.     class MiOyente implements ActionListener{
29.             public void actionPerformed(ActionEvent event){
30.                     if(event.getSource() == btnAzul)
31.                             getContentPane().setBackground(Color.BLUE);
32.                     else if(event.getSource() == btnAmarillo)
33.                             getContentPane().setBackground(Color.YELLOW);
34.                     else if(event.getSource() == btnRojo)
35.                             getContentPane().setBackground(Color.RED);
36.
37.             }
38.     }
39. }
```

Interfaz Listerner y clases *anónimas*

- Generar la clase "oyente" en el propio método add del componente.

```
class HolaMundoFrame extends JFrame{
    JPanel panelBoton = new JPanel();
    JPanel panelEtiqueta = new JPanel();
    JButton btn = new JButton("Pulsa");
    JLabel lbl = new JLabel("¡Hola, mundo!");

    HolaMundoFrame(){
        setTitle("Hola Mundo!");
        panelBoton.add(btn);
        panelEtiqueta.add(lbl);
        setContentPane(panelBoton);
        pack();
        btn.addActionListener(new ActionListener(){
            public void actionPerformed(ActionEvent event){
                setContentPane(panelEtiqueta);
                pack();
            }
        });
    }
}
```

Tipos de eventos y oyentes (I)

Eventos generados y componentes que los soportan		
Eventos generados	**Significado**	**Componentes**
ActionEvent	Hacer clic en el botón	JTextField JComboBox AbstractButton
AdjustementEvent	Cambiar el valor de la barra de desplazamiento	JScrollbar
ItemEvent	Seleccionar o deseleccionar un item	JComboBox AbstractButton
TextEvent	Cambiar el texto	JTextComponent
ComponentEvent	Mover, cambiar tamaño, mostrar u ocultar un componente	Component
ContainerEvent	Añadir o eliminar un componente de un container	Container
FocusEvent	Obtener o perder el focus	Component
KeyEvent	Pulsar o soltar una tecla	Component
MouseEvent	Pulsar o soltar un botón del ratón; entrar o salir de un componente; mover o arrastrar el ratón	Component
MouseWheelEvent	Al girar la rueda del ratón (a partir de SDK 1.4)	Component
WindowEvent	Acciones sobre una ventana: abrir, cerrar, iconizar, restablecer e iniciar el cierre	Window

Tipos de eventos y oyentes (II)

Interfaces de Listener y métodos de cada interface		
Evento	**Interface Listener**	**Métodos de Listener**
ActionEvent	ActionListener	actionPerformed()
AdjustementEvent	AdjustementListener	adjustementValueChanged()
ComponentEvent	ComponentListener	componentHidden() componentMoved() componentResized() componentShown()
ContainerEvent	ContainerListener	componentAdded() componentRemoved()
FocusEvent	FocusListener	focusGained() focusLost()
ItemEvent	ItemListener	itemStateChanged()
KeyEvent	KeyListener	keyPressed() keyReleased() keyTyped()

Tipos de eventos y oyentes (III)

Interfaces de Listener y métodos de cada interface *(continuación)*		
Evento	**Interface Listener**	**Métodos de Listener**
MouseEvent	MouseListener	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
	MouseMotionListener	mouseDragged() mouseMoved()
MouseWheelEvent	MouseWhellListener	mouseWheelMoved()
TextEvent	TextListener	textValueChanged()
WindowEvent	WindowListener	windowActivated() windowDeactivated() windowClosed() windowClosing() windowIconified() windowDeiconified() windowOpened()
	WindowFocusListener	windowGainedFocus() windowLostFocus()
	WindowStateListener	windowStateChanged()

Adaptadores (I)

- Es necesario implementar todos los métodos de las interfaces `Listener`.
  - Por ejemplo, para hacer algo al cerrar una ventana sería necesario implementar los 7 métodos de la interfaz `WindowListener`.
- Algunos interfaces tienen adaptadores,
  - Clases hijas que proporcionan métodos vacíos para cada uno de los métodos de la interfaz

Adaptadores (II)

Interfaces que proporcionan adaptadores	
Interface Listener	**Adaptador**
ComponentListener	ComponentAdapter
ContainerListener	ContainerAdapter
FocusListener	FocusAdapter
KeyListener	KeyAdapter
MouseListener	MouseAdapter
MouseMotionListener	MouseMotionAdapter
WindowListener	WindowAdapter
WindowFocusListener	WindowAdapter
WindowStateListener	WindowAdapter

```
//Se utiliza extends en lugar de implements. Es una clase no una interfaz
class MiWindowListener extends WindowAdapter{
   public void windowClosing(WindowEvent e){
        System.exit(0);
   }
}
```

Ejemplo

- Un GUI para introducir empleados fijos o eventuales (el código del ejemplo se proporciona en un archivo aparte).
  - Utiliza la clase Empresa de la práctica 0.
  - Utiliza la superclase abstracta Empleado y las subclases EmpleadoFijo y EmpleadoEventual.