

Unidad 10

Programación orientada a eventos: Swing

Programación
1º D.A.M.

1

Contenido

1. AWT vs. Swing
2. Elementos de Swing
3. Componentes y contenedores
4. Eventos
5. Partes de un programa Swing

2

1. AWT vs. Swing

1. Características comunes
2. Características de AWT
3. Problemas de AWT
4. Aparición de Swing
5. Aportaciones de Swing
6. Versiones que los soportan

3

1. AWT vs. Swing

■ Características comunes

- Paquetes con clases para crear GUI's
 - Interfaces gráficas de usuario
- Forman parte de la colección de clases JFC
 - *Java Foundation Classes*
 - Paquetes para programar GUI's y multimedia

4

1. AWT vs. Swing

■ Características de AWT

- Primera solución Java para GUI's transportables
- Clases independientes del S.O.
 - Usa clases comunes a todos los S.O. gráficos
 - La máquina virtual traduce esa clase a la forma del S.O. concreto
 - Es el sistema el que genera realmente los componentes
 - Vista coherente al S.O. de que se trate
 - Misma aplicación más elegante o menos según el S.O.

5

1. AWT vs. Swing

■ Problemas de AWT

- Compatibilidad en varios sistemas
- Carencia de componentes avanzados
 - Árboles, tablas, ...
- Gran consumo de recursos del sistema

6

1. AWT vs. Swing

- Aparición de Swing
 - Versión 1.2 de Java (Java2)
 - Se incorpora a las JFC
- Aportaciones de Swing
 - Muchas clases nuevas
 - Menor consumo de recursos
 - Mejor construcción de la apariencia
 - Look & Feel
 - Se construye a partir de AWT
- Versiones que los soportan
 - JDK 1.0 y 1.1 → AWT
 - JDK 1.2 (Java2) → AWT + Swing

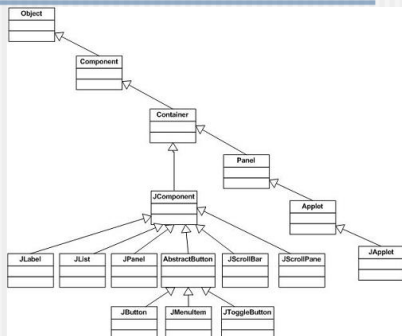
7

2. Elementos de Swing

1. Componentes
2. Contenedores
3. Eventos

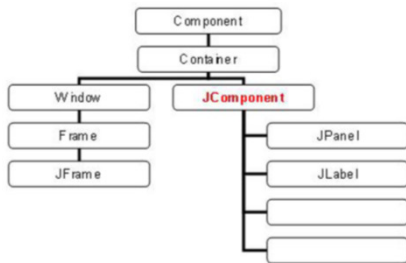
8

2. Elementos de Swing



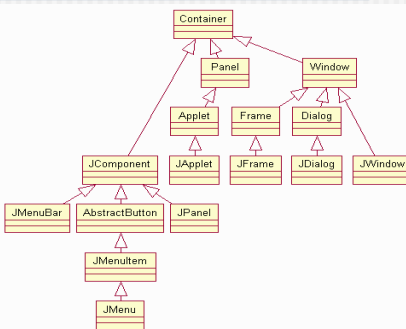
9

2. Elementos de Swing



10

2. Elementos de Swing



11

3. Componentes y contenedores

12

3. Componentes y contenedores

- Componentes
 - Elementos básicos de programación Swing
 - Todo lo que aparece en un GUI de Java
 - Ubicados en contenedores
- Contenedores
 - Agrupan componentes
- Administrador de diseño
 - Dispone la presentación de los componentes

13

3. Componentes y contenedores

- La clase JComponent
 - javax.swing.JComponent
 - Abstracta
 - Padre de todos los componentes
 - Métodos
 - Para controlar la apariencia del objeto
 - Visibilidad, tamaño, posición, tipo de letra, color, ...
 - Al dibujarlo se le asigna un dispositivo de presentación
 - Para controlar el comportamiento del componente
 - Acción del usuario sobre el componente → Evento
 - Objeto de control de eventos → Listener

14

3. Componentes y contenedores

- Contenedores
 - Componentes para almacenar y manejar otros componentes
 - Métodos de java.awt.Container
 - add
 - Añadir un componente a un contenedor
 - remove
 - Eliminar un componente de un contenedor

15

3. Componentes y contenedores

■ Contenedores especiales de Swing

- JWindow
 - Panel de ventana sin bordes ni elementos visibles
- JFrame
 - Ventana típica, con bordes, botones de cierre, ...
- JPanel
 - Contenedor genérico para agrupar componentes
- JDialog
 - Cuadro de diálogo
- JApplet
 - Agrupa componentes a mostrar en un navegador

16

4. Eventos

1. Descripción
2. Escuchadores de eventos (*Listeners*)
3. Adaptadores (*Adapters*)
4. Gestión de eventos
5. Principales objetos de evento

17

4.1. Descripción

- Son un objeto
 - Se lanza por un objeto
 - Cuando ocurre una determinada situación
 - Clic del ratón, pulsación de una tecla, ...
 - Enviado a otro objeto
 - Listener (escuchador)
- Programación basada en eventos
 - Flujo del programa desviado según evento producido
 - Usuario desencadena mensajes con sus acciones

18

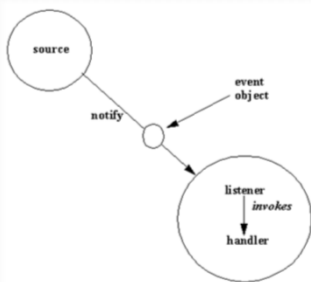
4.1. Descripción

■ Objetos implicados

- Objeto fuente
 - Lanza los eventos, dependiendo del tipo que sea
 - JLabel → de ratón, pero no de teclado
- Objeto escuchador u oyente (*listener*)
 - Recibe el evento producido
 - Lo captura y ejecuta el código asociado
 - Obligado a implementar interfaz asociada al evento
- Objeto de evento
 - Envío del objeto fuente al escuchador

19

4.1. Descripción



20

4.2. Escuchadores de eventos (*Listeners*)

■ Descripción

- Interfaces asociadas a cada tipo de evento
 - Métodos a la espera de que se produzca el evento
 - Método manejador del evento
 - Disparado automáticamente ante el evento
- Ejemplo
 - Interfaz ActionListener
 - Métodos escuchadores de eventos de acción
 - Clic del ratón, etc
 - Ej.: Método `actionPerformed`
 - Gestiona eventos de tipo `ActionEvent`

21

4.2. Escuchadores de eventos (*Listeners*)

- Si una clase quiere capturar eventos
 - Implementar la interfaz (o interfaces) de captura de ese tipo de evento
 - Implementar sus métodos, indicando qué hacer en caso de producirse el evento
- Ejemplo
 - Objeto que quiere escuchar eventos `ActionEvent`
 - Implementar interfaz `ActionListener`
 - En consecuencia, definir método `actionPerformed`
 - Código invocado automáticamente
 - Invocado a producir evento de acción el objeto fuente

22

4.2. Escuchadores de eventos (*Listeners*)

- Actores en el escuchador de eventos
 - **Objeto** de evento
 - Se dispara cuando ocurre un suceso
 - Ej. : Para capturar el ratón
 - `MouseEvent`
 - **Interfaz** de escucha
 - Ha de estar implementada en la clase que captura el evento
 - Ej. : Para clase escuchadora de eventos de ratón
 - `MouseListener`
 - **Método** o métodos de captura del evento
 - Lanzado cuando el evento se produce
 - Ej. : Para capturar eventos `MouseEvent` → interfaz `MouseListener`
 - `mouseReleased` → invocado al liberar un botón del ratón
 - `mousePressed` → invocado al pulsar un botón del ratón
 - `mouseEntered` → invocado cuando el cursor entra en el objeto
 - `mouseExited` → invocado cuando el cursor sale del objeto
 - `mouseClicked` → invocado cuando se hace click en el ratón

23

4.3. Adaptadores (*Adapters*)

- Problema de los *Listeners*
 - Son interfaces, en ocasiones con varios métodos
 - Cada uno asociado a un evento distinto
 - Clase que los implementa obligada a implementar todos sus métodos
 - Aun cuando no requiera tratar algún evento
 - Los métodos no usados tendrán sólo el esqueleto
- Solución : uso de *Adapters*
 - Clase asociada a una interfaz *Listener* concreta
 - Evita tener que poner esqueleto de métodos no usados
 - No todos los *Listener* tienen adaptador
 - Pero se podría crear

24

4.3. Adaptadores (*Adapters*)

- Parejas Listener / Adapter
 - Package java.awt.event
 - ComputerListener / ComputerAdapter
 - ContainerListener / ContainerAdapter
 - FocusListener / FocusAdapter
 - HierarchyBoundsListener / HierarchyBoundsAdapter
 - KeyListener / KeyAdapter
 - MouseListener / MouseAdapter
 - MouseMotionListener / MouseMotionAdapter
 - WindowListener / WindowAdapter
 - Package java.awt.dnd
 - DragSourceListener / DragSourceAdapter
 - DragTargetListener / DragTargetAdapter
 - Package javax.swing.event
 - InternalFrameListener / InternalFrameAdapter
 - MouseInputListener / MouseInputAdapter

25

4.4. Gestión de eventos

- Disparo, captura y tratamiento
- Eliminación de oyentes

26

4.4. Gestión de eventos

- Disparo, captura y tratamiento
 - Objeto fuente
 - Añadir capacidad de enviar eventos
 - Método `add<interfaz_captura_tipo_evento>`
 - Parámetro : objeto escuchador de eventos
 - Ej. : `miBoton.addActionListener(this)`
 - Objeto escuchador de eventos
 - Implementar interfaz asociada al evento
 - Ej. : `... implements ActionListener`
 - Implementar el método de gestión del evento
 - Ej. : `actionPerformed`

27

4.4. Gestión de eventos

■ Eliminación de oyentes

■ Método `remove`

- Para que oyente del objeto deje de escuchar eventos
- Ej. :
`miBoton.removeActionListener(this)`

28

4.4. Gestión de eventos

■ Ejemplo

```
public class Ventana extends JFrame implements ActionListener{
    JButton miBoton = new JButton ("Aceptar");

    public Ventana() {
        ...
        miBoton.addActionListener(this);
        ...
    }
    ...
    public void actionPerformed(ActionEvent e)
    {
        // Código de manejo del evento
    }
}
```

29

4.5. Principales objetos de evento

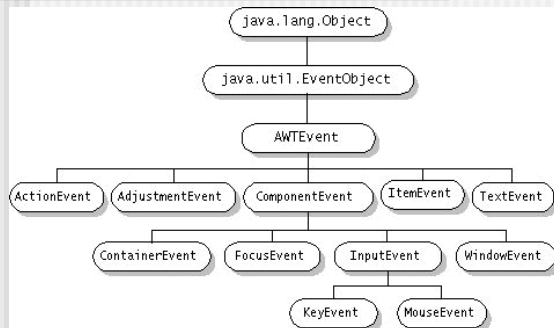
■ EventObject

■ AWTEvent

- ActionEvent
- AdjustmentEvent
- ComponentEvent
 - ContainerEvent
 - FocusEvent
 - InputEvent
 - KeyEvent
 - MouseEvent
 - MouseWheelEvent
 - PaintEvent
 - WindowEvent
- ItemEvent
- TextEvent

30

4.5. Principales objetos de evento



31

4.5. Principales objetos de evento

- **EventObject**
 - Superclase de todos los objetos de evento
 - Métodos
 - Object getSource()
 - String toString()
- **AWTEvent**
 - Padre de los eventos de componentes de Swing y AWT
 - Paquete java.awt.event
 - Métodos
 - void getID()
 - String paramString()
 - void setSource(Object o)
 - protected void consume()
 - protected boolean isConsumed()

32

4.5. Principales objetos de evento

- **InputEvent**
 - Padre de eventos asociados a teclado y ratón
 - KeyEvent
 - MouseEvent
 - Indicadores para saber qué teclas o botones estaban pulsados
 - Método getModifiers() y operación AND (&) con constantes de máscara
 - SHIFT_MASK, ALT_MASK, ALT_GRAPH_MASK, ...
 - Métodos específicos para saberlo
 - isShiftDown(), isAltDown(), isControlDown(), ...
 - Métodos
 - void consume()
 - boolean isConsumed()
 - int getModifiers()
 - int getModifiersExt()
 - long getWhen()
 - boolean isAltDown()
 - boolean isAltGraphDown()
 - boolean isControlDown()
 - boolean isShiftDown()

33

4.5. Principales objetos de evento

- `java.awt.event.ComponentEvent`
 - Un componente se mueve, cambia de tamaño o de visibilidad
- `java.awt.event.FocusEvent`
 - Cambia el foco de un componente
- `java.awt.event.KeyEvent`
 - Se pulsa una tecla
- `java.awt.event.MouseEvent`
 - Se pulsa el ratón sobre un componente
- `java.awt.event.MouseWheelEvent`
 - Movimiento de rueda del ratón
- `java.awt.event.WindowEvent`
 - Captura de eventos de ventana

34

4.5. Principales objetos de evento

- `java.awt.event.ContainerEvent`
 - Se añaden o quitan controles a un contenedor
- `java.awt.event.ActionEvent`
 - Se ejecuta una orden de acción
- `java.awt.event.AdjustmentEvent`
 - Se mueve el scroll de la pantalla
- `javax.swing.event.HyperlinkEvent`
 - Sucede algo a un texto de enlace
- `java.awt.event.ItemEvent`
 - Cambia el estado de un control de tipo ítem
- `javax.swing.event.ListSelectionEvent`
 - Cambio en la selección de un control de lista

35

4.5. Principales objetos de evento

- `javax.swing.event.MenuEvent`
 - Cambio en la selección de un menú
- `javax.swing.event.PopupMenuEvent`
 - Cambio en la selección de un menú de tipo popup
- `javax.swing.event.MenuKeyEvent`
 - Se pulsaron teclas hacia el menú
- `javax.swing.event.MenuDragMouseEvent`
 - Se arrastró el ratón sobre un elemento de menú
- `javax.swing.event.ChangeEvent`
 - Cambio en la selección de un control de lista
- `java.awt.event.InternalFrameEvent`
 - Cambios en los objetos de ventana interna

36

5. Partes de un programa Swing

37

5. Partes de un programa Swing

- Importación de paquetes Swing
- Configuración de aspecto y comportamiento
- Configuración del contenedor de alto nivel
- Configuración e inclusión de componentes
- Manejo de eventos

38

Unidad 10 Programación orientada a eventos: Swing

Programación
1º D.A.M.

39
