INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA DE INGENIERÍA Y CIENCIAS SOCIALES Y ADMINISTRATIVAS

Material para el curso de Programación Orientada a Objetos

Contenido

4.2 Diseño de la interfaz gráfica

4.2.1 Contenedores y componentes de la interfaz gráfica (GUI)

4.2.2 Recepción y procesamiento de eventos a través del enfoque Modelo-Vista-Controlador (MVC)

Interface Grafica

Una Interface Grafica de Usuario (GUI) se compone de los siguientes elementos:

- Contenedores. Están en el tope de la jerarquía de contención. Todos los componentes se agregan visualmente dentro de ellos, por ejemplo, JFrame, JWindow, etc. También se consideran componentes, por lo cual es válido anidar un contenedor dentro de otro.
- Componentes. Botones, Cajas de texto, Listas, etc.
- Layout Managers. Sirven para acomodar dentro de un contenedor a los componentes y otros contenedores, por ejemplo, BorderLayout y GridLayout.

Java Foundation Classes es un conjunto de paquetes para soporte de GUI que comprende:

- Swing: Conjunto de componentes diseñado para reemplazar la mayoría de los contenidos en el AWT original (y algunos más avanzados).
- Otras características como graficas elementales en 2D, internacionalización, “Look and Feel” (Apariencia y Funcionalidad) portable, etc.

Sin embargo hay que mencionar que el modelo de eventos de Java que veremos a continuación, sigue teniendo AWT (Abstract Window Toolkit) como base, ya sea que se utilicen componentes de uno, otro o ambos paquetes.

Principales Paquetes

NOMBRE DEL PACKAGE	FUNCIÓN
javax.swing	Componentes ligeros como Botón, Lista, etc.
javax.swing.border	Clases e Interfaces para dibujar bordes especializados
javax.swing.event	Soporte para recibir los eventos disparados por componentes
javax.swing.undo	Soporte para deshacer o rehacer las últimas acciones, como en editores de texto
javax.swing.colorchooser	Clases e Interfaces usadas por el componente JColorChooser
javax.swing.filechooser	Clases e Interfaces usadas por el componente JFileChooser
javax.swing.table	Clases e Interfaces usadas por el componente JTable
javax.swing.tree	Clases e Interfaces usadas por el componente JTree
javax.swing.plaf	Clases e Interfaces usadas para obtener el Pluggable Look And Feel
javax.swing.plaf.basic javax.swing.plaf.metal javax.swing.plaf.multi javax.swing.plaf.synth	Diversos Look And Feel
javax.swing.text	Clases e Interfaces usadas por componentes de texto
javax.swing.text.html	Clase HTMLEditorKit entre otras para crear editores HTML
javax.swing.text.html.parser	Clases para un analizador HTML
java.awt	Componentes ligeros como Button, Label, etc.
java.awt.event	Eventos que sirven como parámetros en los métodos que los manipulan, además de las interfaces necesarias para los Listeners

Creación de la Interfaz

Los pasos para construir una GUI son:

1. Instanciar cada uno de los componentes y ponerles los valores adecuados a sus atributos (visibilidad, texto, comportamiento, etc.)
2. Agregar visualmente cada uno de ellos al contenedor adecuado mediante el método add (). Estos últimos por supuesto deben estar ya instanciados.
   Opcionalmente puede usarse un LayoutManager, que sirve para acomodar y reacomodar de diversas maneras los componentes.
3. Registrar el o los Listeners necesarios para capturar los eventos deseados para cada componente (llamado objeto Source), mediante los métodos add*XXX*Listener (), donde las XXX corresponden a la categoría de evento que veremos a continuación.
   Estos Listeners son objetos que deben implementar las interfaces necesarias para cada grupo de eventos, y como vimos anteriormente, debemos codificar todos los métodos de cada interface, al menos con sus llaves vacías.

Existen muchos IDE que facilitan esta labor, como NetBeans de Sun, Symantec Visual Café, Borland JBuilder, IBM Visual Age for Java, Eclipse, Microsoft Visual J++, etc.

Veamos las principales características de la clase Component de AWT, de la cual heredan todos los componentes y contenedores:

METODOS	PROPOSITO
void setEnabled(boolean) boolean isEnabled()	El atributo *enabled* indica si el componente responde o no a las acciones del usuario
void setVisible(boolean) boolean isVisible()	El atributo *visible* indica si el componente lo es
void setLocation(int x, int y) void setLocation(Point p) Point getLocation()	Posición en pixeles. La clase *Point* tiene los atributos x, y
void setSize(int ancho, int alto) void setSize(Dimension d) Dimension getSize()	Tamaño en pixeles. La clase *Dimension* tiene los atributos ancho, alto
void setBounds(int x, int y, int ancho, int alto)	Combinación de setLocation y setSize
void requestFocus()	Poner el foco en el objeto, que significa que la siguiente acción del usuario se dirigirá al objeto
void setBackground(Color c) Color getBackground()	Color de fondo
void setForeground(Color c) Color getForeground()	Color del primer plano
void setFont(Font f) Font getFont()	Fuente del texto

Eventos

Se trata de una ocurrencia en el Sistema Operativo que es disparada por alguna acción que sucede en el ambiente grafico, como el usuario tecleando una letra.
Cuando esto pasa, se le notifica al objeto u objetos que son el destino de dicho evento. En Java, esta notificación provoca que un método en específico sea ejecutado. Los programadores escribimos código en dichos métodos para manejar (handle) cada evento.

El modelo de eventos de Java hace una clara distinción entre los 2 objetos involucrados:

- El origen o *Source*, que genera el evento, y que suele ser un componente
- El destino o *Listener*, que recibe la notificación del evento

Este modelo también establece categorías de eventos. Algunas categorías solamente tienen un evento, y otras tienen varios. Cada categoría tiene:

- Su propia clase de evento
- Su propia interface, que define un Listener para los eventos en esa categoría
- Un método diferente para cada evento en la interface, como veremos en las siguientes tablas:

CLASE DE EVENTO	COMPONENTE SOURCE	INTERFASE PARA EL LISTENER	METODOS EN LA INTERFASE
ComponentEvent	cualquier Component	ComponentListener	componentMoved() componentResized() componentShown() componentHidden()
FocusEvent		FocusListener	focusGained() focusLost()
KeyEvent		KeyListener	keyPressed() keyReleased() keyTyped()
MouseEvent		MouseListener	mouseClicked() mouseEntered() mouseExited() mousePressed() mouseReleased()
		MouseMotionListener	mouseMoved() mouseDragged()

Todos los métodos contenidos en estas interfaces son *public void*, y reciben como parámetro un objeto de la primera columna, el cual nos puede dar aun mayor detalle sobre el evento ocurrido, como veremos en el ejemplo.

Además hay que notar la nomenclatura estandarizada, la cual nos facilitará la programación. La siguiente tabla tiene la misma estructura, pero aplica a componentes específicos:

CLASE DE EVENTO	COMPONENTE SOURCE	INTERFASE PARA EL LISTENER	METODOS EN LA INTERFASE
ActionEvent	Button List MenuItem TextField	ActionListener	actionPerformed()
AdjustmentEvent	Adjustable ScrollBar	AdjustmentListener	adjustmentValueChanged()
ContainerEvent	Container	ContainerListener	componentAdded() componentRemoved()
ItemEvent	Checkbox CheckboxMenuItem Choice ItemSelectable List	ItemListener	itemStateChanged()
TextEvent	TextComponent	TextListener	textValueChanged()
WindowEvent	Window	WindowListener	windowActivated() windowClosed() windowClosing() windowDeactivated() windowDeiconified() windowIconified() windowOpened()

Para mostrar su “interés” en un evento determinado, cada Listener debe estar registrado en el o los objetos Source antes de que ocurran dichos eventos. Es en el objeto Source donde se encuentran los métodos necesarios para hacer este registro, y su sintaxis es:
public synchronized void addXXXListener (XXXListener obj);

Los Listeners se definen mediante interfaces, las cuales deben ser implementadas para que un objeto pueda ser aceptable como un Listener.

Analiza el ejemplo Saludador.java
El botón es origen de un ActionEvent. Su Listener es el mismo Applet (este enfoque es simple para efectos didácticos; otros enfoques más sofisticados que requieren conocer adaptadores y patrones de diseño, serán vistos en otro curso).

Por último, en las clases de evento existen algunos métodos que nos brindan mayor información sobre el evento ocurrido. He aquí algunos de ellos:

CLASE DE EVENTO	METODOS	DESCRIPCIÓN
KeyEvent	char getKeyChar() int getKeyCode()	Tecla presionada
MouseEvent	int getX() int getY() Point getPoint()	Posicion del Mouse
EventObject	Object getSource()	Objeto origen del evento
AdjustmentEvent	int getValue()	Nuevo valor del ScrollBar

Referencias

“Java Programming Language, Java SE 6 Student Guide”
Oracle

“Mastering Java 2.0”
Sun Microsystems