PROGRAMACIÓN EN JAVA

TEMA 3: Swing

Bibliografía

- "JAVA 2 Interfaces gráficas y aplicaciones para Internet – 2ª Edición"
  Fco. Javier Ceballos. Ed. Ra-Ma, 2006
  Capítulo: 2

Contenido

- Adaptadores
- Ejemplo – sin usar adaptador
- Ejemplo – utilizando adaptador
- Eventos
- Manejadores de eventos de swing
- Resumen de manejadores de eventos
- Interfaces, adaptadores y métodos
- Solución – Evento

Adaptadores

- Para entender el uso de adaptadores debemos recordar ¿Cómo se asignan manejadores de eventos a un componente?

  R= 1. Crear la clase que implemente la interfaz
  2. Construir un objeto de esa clase
  3. Vincular el ME con el componente

- Ahora debemos recordar la filosofía de implementación de las interfaces (obligatoriamente tenemos que definir todos y c/u de los métodos)

- La idea de los adaptadores es que son clases especiales que implementan interfaces que contienen más de un método, estas clases definirán de forma predeterminada cada uno de los métodos declarados en la interfaz que están implementando, luego nosotros nos debemos crear una clase derivada del adaptador y redefinir únicamente aquellos métodos que vallamos a utilizar (La idea es evitar el trabajo de tener que redefinir métodos innecesarios)

- ¿Qué debemos hacer si no utilizamos los adaptadores?

  R= Tendríamos que redefinir todos los métodos proporcionados por la interfaz, dejando vacíos todos aquellos que no necesiten realizar alguna operación fuera de lo normal

Ejemplo – sin usar adaptador

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

class frame extends JFrame {
        public frame() {
                setTitle("Hola!!!");
                setSize(400,400);
                addWindowListener(new WindowListener() {
                        public void windowOpened(WindowEvent e) {}
                        public void windowClosing(WindowEvent e) {
                                frameWindowClosing(e);
                        }
                        public void windowClosed(WindowEvent e) {}
                        public void windowActivated(WindowEvent e) {}
                        public void windowDeactivated(WindowEvent e) {}
                        public void windowIconified(WindowEvent e) {}
                        public void windowDeiconified(WindowEvent e) {}
                });
        }
        public void frameWindowClosing(WindowEvent e) {
                System.out.println("Se genero el evento windowClosing");
                System.exit(0);
        }
        public static void main(String[] args) {
                new frame().setVisible(true);
        }
}
```

Ejemplo – utilizando adaptador

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

class frame extends JFrame {
        public frame() {
                setTitle("Hola!!!");
                setSize(400,400);
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                frameWindowClosing(e);
                        }
                });
        }
        public void frameWindowClosing(WindowEvent e) {
                System.out.println("Se genero el evento windowClosing");
                System.exit(0);
        }
        public static void main(String[] args) {
                new frame().setVisible(true);
        }
}
```

Eventos

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

class frame extends JFrame {
        public frame() {
                initComponents();
        }
        private void initComponents() {
                //Etiqueta lblSaludo
                lblSaludo = new JLabel("Hola Mundo");
                lblSaludo.setBounds(90,50,80,70);
                //JFrame frame
                setTitle("PRUEBA!!!");
                setSize(300,200);
                getContentPane().setLayout(null);
                //Añadir componentes
                getContentPane().add(lblSaludo);
                //Añadir manejadores de eventos
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
        }
        public static void main(String[] args) {
                frame fr = new frame();
                fr.setVisible(true);
                //new frame().setVisible(true);
        }
        private JLabel lblSaludo = null;
}
```

Eventos (cont.)

- Dado el ejemplo anterior ¿Cómo hacemos para que al dar clic sobre la etiqueta esta cambie de color y cambie el mensaje que se muestra inicialmente?

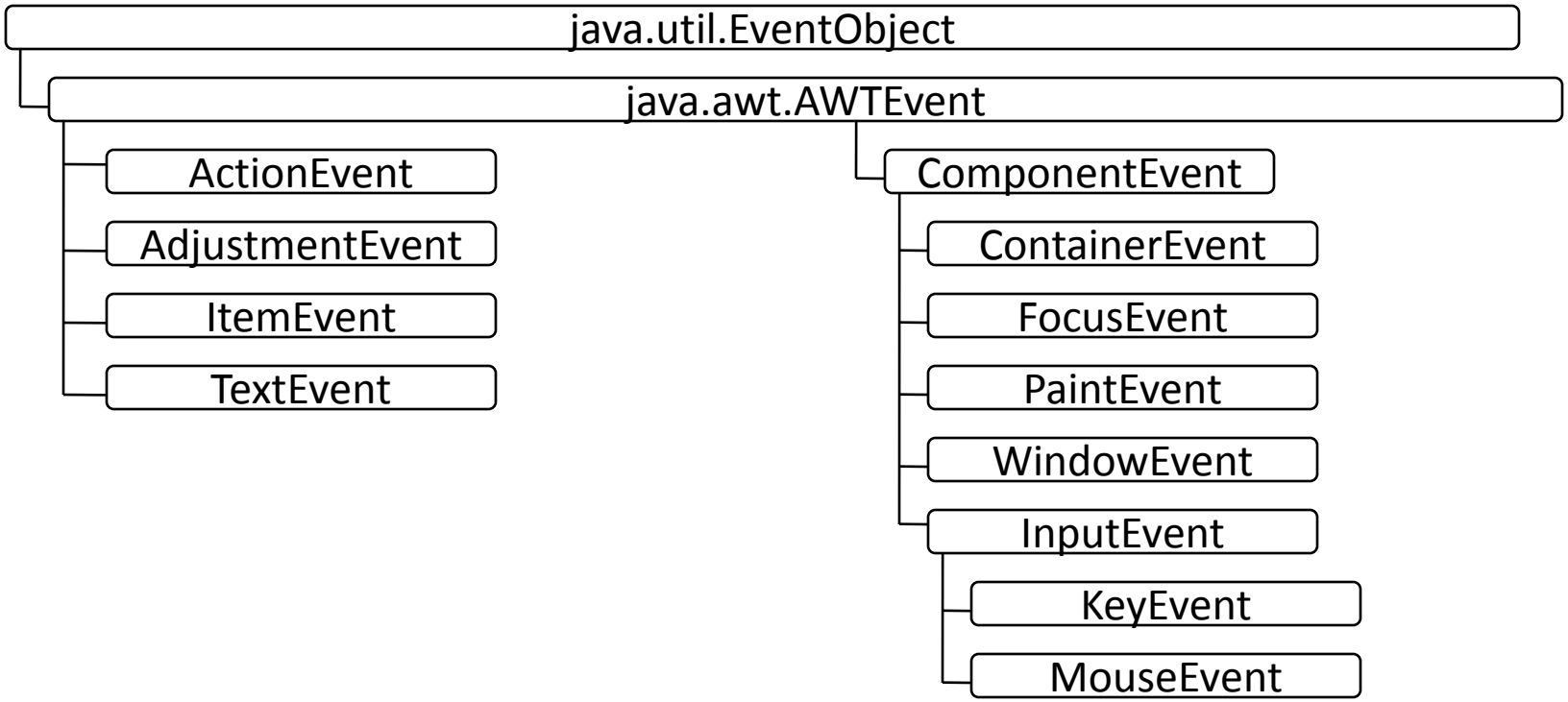
  R= Debemos escribir el código necesario que escuche el evento de darle clic a la etiqueta con el mouse

- Aunque los desarrolladores de JAVA considerasen que para mejorar el lenguaje se necesitaba dejar a un lado las librerías AWT e introducir las Swing no sintieron lo mismo del sistema de gestión de eventos, pues consideraron que era lo suficientemente bueno

- Por lo tanto, se usará el mismo modelo de delegación de eventos (es decir, objetos fuentes de eventos y objetos receptores de eventos que implementan las interfaces necesarias)

Manejadores de eventos de swing

- JAVA 2 proporciona varios manejadores de eventos, cada uno de los cuales maneja un tipo particular de eventos
- Los tipos de eventos más comunes son:



Manejadores de eventos de swing (cont.)

- Los manejadores de eventos que manipulan los tipos de eventos de la fig. anterior son objetos que serán implementados a partir de las interfaces siguientes:

Manejador de evento	Evento manejado	Descripción
ComponentListener	ComponentEvent	Generados por los componentes cuando cambian su tamaño, posición o visibilidad
FocusListener	FocusEvent	Generados por los componentes cuando ganan o pierden el foco (ganar el foco conlleva recibir entradas desde el teclado)
KeyListener	KeyEvent	Generados por el componente que tiene el foco, cuando recibe entradas desde el teclado
MouseListener	MouseEvent	Generados por los componentes cuando el cursor del ratón entra en su área o sale de ella, y cuando el usuario pulsa o suelta un botón del ratón
MouseMotionListener	MouseEvent	Generados por un componente cuando el ratón se mueve sobre él (El adaptador MouseInputAdapter implementa tanto MouseListener como MouseMotionListener)

Manejadores de eventos de swing (cont.)

- Otros manejadores de eventos soportados por sólo algunos componentes son:

Manejador de evento	Evento manejado	Descripción
ActionListener	ActionEvent	Permite manejar los eventos de acción en los componentes: JButton, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem, JToggleButton, JCheckBox, JRadioButton, JComboBox, JFileChooser y JPasswordFIeld. Un evento de acción ocurre cuando se hace clic sobre un botón en general, cuando se elige un componente de un menú o cuando se pulsa la tecla Enter en una caja de texto
AdjustmentListener	AdjustmentEvent	Permite a una barra de desplazamiento (JScrollBar) responder a los movimientos de su cuadrado de desplazamiento
ContainerListener	ContainerEvent	Permite a un contenedor notificar que se le ha añadido o quitado un componente
ChangeListener	ChangeEvent	Permite manejar los eventos relacionados con el cambio de estado de los componentes: JButton, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem, JToggleButton, JCheckBox, JRadioButton, JColoChooser, JProgressBar, JSlider, JTabbedPane y JViewport. Un evento ChangeEvent ocurre cuando un componente cambia de estado

Manejadores de eventos de swing (cont.)

Manejador de evento	Evento manejado	Descripción
ItemListener	ItemEvent	Generados como consecuencia de los cambios de estado en alguno de los componentes siguientes: JButton, JMenuItem, JCheckBoxMenuItem, JRadioButtonMenuItem, JToggleButton, JCheckBox, JRadioButton y JComboBox
ListSelectionListener	ListSelectionEvent	Permite manejar los eventos de selección, los cuales ocurren cuando el elemento seleccionado de un componente JList o JTable cambia o ha cambiado
WindowListener	WindowEvent	Permite manejar los eventos de ventana que son propios de componentes como JFrame y JDialog. Estos eventos se producen cada vez que la ventana se abre, se cierra, se minimiza, se restaura, se activa o se desactiva
CaretListener	CaretEvent	Permite manejar los eventos relacionados con el punto de inserción en los componentes: JTextComponent, JEditorPane, JTextPane, JTextArea, JTextField y JPasswordFIeld. Un evento CaretEvent ocurre cuando se mueve el punto de inserción sobre uno de estos componentes o cuando cambia la selección del texto
DocumentListener	DocumentEvent	Ocurren cuando el documento contenido en alguno de los componentes especificados en la fila anterior cambia
UndoableEditListener	UndoableEditEvent	Ocurren cuando una operación realizada en el documento de alguno de los componentes especificados en la fila anterior puede ser deshecha

Resumen de manejadores de eventos

Interfaz	Evento	Asociación evento-componente	Componentes que generan este tipo de eventos
ActionListener	ActionEvent	addActionListener()	JButton, JList, JTextField, JMenuItem, JCheckBoxMenuItem, JMenu, JPopupMenu
AdjustmentListener	AdjustmentEvent	addAdjustmentListener()	JScrollbar y cualquier objeto que implemente la interfaz Adjustable
ComponentListener	ComponentEvent	addComponentListener()	Component, JButton, JCanvas, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, JTextField
FocusListener	FocusEvent	addFocusListener()	Component, JButton, JCanvas, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, JTextField

Resumen de manejadores de eventos (cont.)

Interfaz	Evento	Asociación evento-componente	Componentes que generan este tipo de eventos
ContainerListener	ContainerEvent	addContainerListener()	Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame
KeyListener	KeyEvent	addKeyListener()	Component, JButton, JCanvas, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, JTextField
MouseListener	MouseEvent	addMouseListener()	Component, JButton, JCanvas, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, JTextField
WindowListener	WindowEvent	addWindowListener()	Window, JDialog, JFileDialog, JFrame
ItemListener	ItemEvent	addItemListener()	JCheckBox, JComboBox, JList, JCheckBoxMenuItem
TextListener	TextEvent	addTextListener()	JTextComponent, JTextArea, JTextField

- La clase que escucha los eventos debe implementar todos los métodos de la interface, aunque no los use todos, porque si no se convierte en abstracta
- La solución a esto es hacer uso de los adaptadores

Interfaces, adaptadores y métodos

Interfaz	Adaptador	Métodos
ActionListener	-	public void actionPerformed(ActionEvent e)
AdjustmentListener	-	public void adjustmentValueChanged(AdjustmentEvent e)
ComponentListener	ComponentAdapter	public void componentHidden(ComponentEvent e) public void componentShown(ComponentEvent e) public void componentMoved(ComponentEvent e) public void componentResized(ComponentEvent e)
ContainerListener	ContainerAdapter	public void componentAdded(ContainerEvent e) public void componentRemoved(ContainerEvent e)
FocusListener	FocusAdapter	public void focusGained(FocusEvent e) public void focusLost(FocusEvent e)
KeyListener	KeyAdapter	public void keyPressed(KeyEvent e) public void keyReleased(KeyEvent e) public void keyTyped(KeyEvent e)
MouseListener	MouseAdapter	public void mouseClicked(MouseEvent e) public void mouseEntered(MouseEvent e)
		public void mouseExited(MouseEvent e) public void mousePressed(MouseEvent e) public void mouseReleased(MouseEvent e)

Interfaces, adaptadores y métodos (cont)

Interfaz	Adaptador	Métodos
MouseMotionListener	MouseMotionAdapter	public void mouseDragged(MouseEvent e) public void mouseMoved(MouseEvent e)
WindowListener	WindowAdapter	public void windowOpened(WindowEvent e) public void windowClosing(WindowEvent e) public void windowClosed(WindowEvent e) public void windowActivated(WindowEvent e) public void windowDeactivated(WindowEvent e) public void windowIconified(WindowEvent e) public void windowdeiconified(WindowEvent e)
ItemListener	-	public void itemStateChanged(ItemEvent e)

Solución – Evento

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

class frame extends JFrame {
        public frame() {
                initComponents();
        }
        private void initComponents() {
                //Etiqueta lblSaludo
                lblSaludo = new JLabel("Hola Mundo");
                lblSaludo.setBounds(90,50,80,70);
                lblSaludo.addMouseListener(new MouseAdapter() {
                        public void mouseClicked(MouseEvent e) {
                                lblSaludoMouseClickled(e);
                        }
                });
                //JFrame frame
                setTitle("PRUEBA!!!");
                setSize(300,200);
                getContentPane().setLayout(null);
                //Añadir componentes
                getContentPane().add(lblSaludo);
                //Añadir manejadores de eventos
                addWindowListener(new WindowAdapter() {
                        public void windowClosing(WindowEvent e) {
                                System.exit(0);
                        }
                });
        }
```

Solución – Evento (cont.)

```
public void lblSaludoMouseClickled(MouseEvent e) {
        int rojo = (int)(Math.random()*256);
        int verde = (int)(Math.random()*256);
        int azul = (int)(Math.random()*256);
        JLabel lblTemporal = (JLabel) e.getSource();
        if(lblTemporal.getText()=="Hola Mundo")
                lblTemporal.setText("Adios Mundo");
        else
                lblTemporal.setText("Hola Mundo");
        lblTemporal.setForeground(new Color(rojo,verde,azul));
}
public static void main(String[] args) {
        frame fr = new frame();
        fr.setVisible(true);
        //new frame().setVisible(true);
}
private JLabel lblSaludo = null;
}
```