GUI

Manejo de eventos de ratón

Para manejar eventos de ratón debe implementarse los métodos de las interfaces de escucha de eventos **MouseListener** y **MouseMotionListener**. Estos eventos pueden atraparse para cualquier componente de la GUI que derive de java.awt.Component. A continuación se muestran los métodos de las interfaces:

Métodos de la interface **MouseListener**

public void mousePressed(MouseEvent evento)
Se llama cuando se oprime un botón del ratón, mientras el cursor del ratón está sobre un componente.

public void mouseClicked(MouseEvent evento)
Se llama cuando se oprime y suelta un botón del ratón, mientras el cursor del ratón permanece estacionario sobre un componente.

public void mouseReleased(MouseEvent evento)
Se llama cuando se suelta un botón del ratón después de ser oprimido. Este evento siempre sigue después de un evento **mousePressed**.

public void mouseEntered(MouseEvent evento)
Se llama cuando el botón del ratón entra a los límites de un componente.

public void mouseExited(MouseEvent evento)
Se llama cuando el cursor del ratón sale de los límites de un componente.

Métodos de la interface **MouseMotionListener**

public void mouseDragged(MouseEvent evento)
Se llama cuando el botón del ratón se oprime mientras el cursor del ratón se encuentra sobre un componente y se mueve mientras el botón sigue oprimido. Este evento siempre sigue después de una llamada a **mousePressed**. Todos los eventos de arrastre del ratón se envían al componente en el cual empezó la acción de arrastre.

public void mouseMoved(MouseEvent evento)
Se llama al moverse el ratón cuando su cursor se encuentra sobre un componente. Todos los eventos de movimiento se envían al componente sobre el cual se encuentra el ratón posicionado en ese momento.

Objeto MouseEvent

Cada uno de los manejadores de eventos de ratón toma un objeto **MouseEvent** como su argumento. Un objeto **MouseEvent** contiene información acerca del evento de ratón que ocurrió, incluyendo las coordenadas **x** e **y** de la ubicación en donde ocurrió el evento.

Ejemplo que demuestra la implementación de las interfaces MouseListener y MouseMotionListener

```
// Fig. 13.17: RastreadorRaton.java
// Demostración de los eventos de ratón.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class RastreadorRaton extends JFrame
   implements MouseListener, MouseMotionListener {

   private JLabel barraEstado;

   // configurar GUI y registrar manejadores de eventos de ratón
   public RastreadorRaton()
   {
      super( "Demostración de los eventos de ratón" );

      barraEstado = new JLabel();
      getContentPane().add( barraEstado, BorderLayout.SOUTH );

      addMouseListener( this );        // escucha sus propios eventos de ratón
      addMouseMotionListener( this );  // y de movimiento de ratón

      setSize( 300, 125 );
      setVisible( true );
   }
   // Manejadores de eventos de MouseListener
   // manejar el evento cuando el botón del ratón se suelta inmediatamente después de
oprimir
   public void mouseClicked( MouseEvent evento )
   {
      barraEstado.setText( "Se hizo clic en [" + evento.getX() +
         ", " + evento.getY() + "]" );
   }
   // manejar evento cuando se oprime el botón del ratón
   public void mousePressed( MouseEvent evento )
   {
      barraEstado.setText( "Se oprimió en [" + evento.getX() +
         ", " + evento.getY() + "]" );
   }
   // manejar evento cuando se suelta el ratón después de arrastrar
   public void mouseReleased( MouseEvent evento )
   {
      barraEstado.setText( "Se soltó en [" + evento.getX() +
         ", " + evento.getY() + "]" );
   }
```

```
   // manejar el evento cuando el ratón entra al área
   public void mouseEntered( MouseEvent evento )
   {
      barraEstado.setText( "Ratón entro en [" + evento.getX() +
         ", " + evento.getY() + "]" );
      getContentPane().setBackground( Color.GREEN );
   }

   // manejar evento cuando el ratón sale del área
   public void mouseExited( MouseEvent evento )
   {
      barraEstado.setText( "Ratón fuera de la ventana" );
      getContentPane().setBackground( Color.WHITE );
   }

   // Manejadores de eventos de MouseMotionListener
   // manejar el evento cuando el usuario arrastra el ratón con el botón oprimido
   public void mouseDragged( MouseEvent evento )
   {
      barraEstado.setText( "Se arrastró en [" + evento.getX() +
         ", " + evento.getY() + "]" );
   }

   // manejar el evento cuando el usuario mueve el ratón
   public void mouseMoved( MouseEvent evento )
   {
      barraEstado.setText( "Se movió en [" + evento.getX() +
         ", " + evento.getY() + "]" );
   }

   public static void main( String args[] )
   {
      RastreadorRaton aplicacion = new RastreadorRaton();
      aplicacion.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }

} // fin de la clase RastreadorRaton
/**************************************************************************
 * (C) Copyright 1992-2003 by Deitel & Associates, Inc. and               *
 * Prentice Hall. All Rights Reserved.                                    *
 **************************************************************************/
```

Uso de clases internas

Ejemplo de uso de una clase interna que maneja los eventos de `ItemListener`

```
// Fig. 13.11: PruebaCasillaVerificacion.java
// Creación de botones JCheckBox.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class PruebaCasillaVerificacion extends JFrame {
   private JTextField campo;
   private JCheckBox negrita, cursiva;

   // configurar GUI
   public PruebaCasillaVerificacion()
   {
      super( "Prueba de JCheckBox" );

      // obtener panel de contenido y establecer su esquema
      Container contenedor = getContentPane();
      contenedor.setLayout( new FlowLayout() );

      // configurar objeto JTextField y establecer su tipo de letra
      campo = new JTextField( "Observe el cambio en el estilo de tipo de letra", 25 );
      campo.setFont( new Font( "Serif", Font.PLAIN, 14 ) );
      contenedor.add( campo );

      // crear objetos casilla de verificación
      negrita = new JCheckBox( "Negrita" );
      contenedor.add( negrita );

      cursiva = new JCheckBox( "Cursiva" );
      contenedor.add( cursiva );

      // registrar componentes de escucha para los objetos JCheckBox
      ManejadorCasillaVerificacion manejador = new ManejadorCasillaVerificacion();
      negrita.addItemListener( manejador );
      cursiva.addItemListener( manejador );

      setSize( 300, 100 );
      setVisible( true );

   } // fin del constructor de PruebaCasillaVerificacion

   public static void main( String args[] )
   {
      PruebaCasillaVerificacion aplicacion = new PruebaCasillaVerificacion();
      aplicacion.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }
```

```
   // clase interna privada para manejo de eventos de ItemListener
   private class ManejadorCasillaVerificacion implements ItemListener {
      private int valNegrita = Font.PLAIN;
      private int valCursiva = Font.PLAIN;

      // responder a eventos de casilla de verificación
      public void itemStateChanged( ItemEvent evento )
      {
         // procesar eventos de casilla de verificación negrita
         if ( evento.getSource() == negrita )
            valNegrita = negrita.isSelected() ? Font.BOLD : Font.PLAIN;

         // procesar eventos de casilla de verificación cursiva
         if ( evento.getSource() == cursiva )
            valCursiva = cursiva.isSelected() ? Font.ITALIC : Font.PLAIN;

         // establecer tipo de letra del campo de texto
         campo.setFont( new Font( "Serif", valNegrita + valCursiva, 14 ) );

      } // fin del método itemStateChanged

   } // fin de la clase interna privada ManejadorCasillaVerificacion

} // fin de la clase PruebaCasillaVerificacion
```


Clases Adaptadoras para implementar manejadores de eventos

Muchas de las interfaces de escucha de eventos proporcionan varios métodos, **`MouseListener`** y **`MouseMotionListener`** son dos ejemplos. No siempre es deseable declarar todos los métodos de una interfaz de escucha de eventos. Por ejemplo un programa podría necesitar solamente el manejador **`mouseClicked`** de la interfaz **`MouseListener.`** Para muchas de las interfaces de escucha de eventos que contienen varios métodos, el paquete **`java.awt.event`** y el paquete **`java.swing.event`** proporciona clases adaptadoras de escucha de eventos. Una clase adaptadora implementa una interfaz y proporciona una implementación predeterminada(con un cuerpo vacío para los métodos) de todos los métodos en la interfaz. El programador puede extender la clase adaptadora para heredar la implementación predeterminada de cada método y, por ende sobrescribir el (los) métodos necesarios para el manejo de eventos.

Ejemplo de uso de una clase adaptadora que maneja eventos de `MouseListener`

```
// Fig. 13.20: DetallesRaton.java
```

```
// Demostración de los clics de ratón y cómo diferenciar entre los botones del ratón.
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class DetallesRaton extends JFrame {
   private int xPos, yPos;

   // establecer cadena barra título; registrar escucha ratón; ajustar tamaño y
mostrar ventana
   public DetallesRaton()
   {
      super( "Clics y botones del ratón" );

      addMouseListener( new ManejadorClicsRaton() );

      setSize( 350, 150 );
      setVisible( true );
   }

   // dibujar objeto String en la ubicación donde se hizo clic con el ratón
   public void paint( Graphics g )
   {
      // llamar al método paint de la superclase
      super.paint( g );

      g.drawString( "Se hizo clic en: [" + xPos + ", " + yPos + "]",
         xPos, yPos );
   }

   public static void main( String args[] )
   {
      DetallesRaton aplicacion = new DetallesRaton();
      aplicacion.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
   }

   // clase interna para manejar eventos de ratón
   private class ManejadorClicsRaton extends MouseAdapter {

      // manejar evento de clic del ratón y determinar cuál botón se oprimió
      public void mouseClicked( MouseEvent evento )
      {
         xPos = evento.getX();
         yPos = evento.getY();

         String titulo = "Se hizo clic " + evento.getClickCount() + " Veces";
```

```
         if ( evento.isMetaDown() )  // botón derecho del ratón
            titulo += " con el botón derecho del ratón";

         else if ( evento.isAltDown() )  // botón de en medio del ratón
            titulo += " con el botón central del ratón";

         else  // botón izquierdo del ratón
            titulo += " con el botón izquierdo del ratón";

         setTitle( titulo );  // establecer barra de título de la ventana
         repaint();

      } // fin del método mouseClicked

   } // fin de la clase interna privada ManejadorClicsRaton

} // fin de la clase DetallesRaton
```


A continuación se muestran varias clases adaptadoras de **`java.awt.event`** y las interfaces que implementan.

Clase adaptadora de eventos	Implementa a la interfaz
`ComponentAdapter`	**`ComponentListener`**
`ContainerAdapter`	**`ContainerListener`**
`FocusAdapter`	**`FocusListener`**
`KeyAdapter`	**`KeyListener`**
`MouseAdapter`	**`MouseListener`**
`MouseMotionAdapter`	**`MouseMotionListener`**
`WindowAdapter`	**`WindowListener`**

Enlaces de interés

1. public abstract class **`MouseAdapter`** extends **`Object`** implements **`MouseListener.`**
   http://java.sun.com/j2se/1.4.2/docs/api/java/awt/event/MouseAdapter.html
2. How to Write a Mouse Listener.
   http://java.sun.com/docs/books/tutorial/uiswing/events/mouselistener.html