EVENTS

Un event simplement es una acció creada per l'usuari normalment. El usuari fa click sobre un boto, desplega una llista, ... Per detectar si un event s'ha produït necessitem els Event Listeners, que vigilen si es produeix un event per tal de poder respondre a aquest. Per tant sempre que creem un objecte que volem escoltar crearem un objecte listener que ens indiqui quina acció s'ha produït. Hi ha tants listeners com events. Son interfícies per tant haurem de implementar mètodes. Per començar haurem de importar java.awt.event.*

Els grups principals son:

- ActionListener: Events d'acció generats per l'usuari.
- AdjustmentListener: Generats quan un component s'ajusta.
- FocusListener: Un component guanya o per el focus.
- ItemListener: Canvis en ítems de Caixes o llistes.
- KeyListener: Pulsació de tecles.
- MouseListener: Pulsació o entrada de ratolí
- MouseMotionListener: Moviement de ratolí sobre component.
- WindowListener: Canvis en finestres

Una classe pot implementar tants listeners com sigui necessari.

Public class Finestra extends JFrame implements ActionListener,MouseListener

Cada EventListener ha de tenir un mètode listener que indicarà quin component està escoltant i com s'ha de comportar en cas que es produeixi.

Els mètodes son molt similars als events:

- addActionListener()
- addFocusLister()
- addItemListener()
- addKeyListern()
- addMouseListener()
- addMouseMotionListener()
- addTextListener()
- addWindowListener()

Exemple:

```
package Events;

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

public class ActionCommand extends JFrame implements ActionListener {
    JButton b1 = new JButton("Curso de JavaScript");
        b1.addActionListener(this);
}
```

Si tenim més d'un component amb un listener ActionEvent hauríem de identificar aquest amb un argument per diferenciar de quin component s'ha produït l'event.

```
    public void actionPerformed(ActionEvent event) {

       //ens permet determinar el component que ha generat l'event.

        Object font = event.getSource();

             If(font instaceof JTextField){

        //programar acció

       }

        Else if (font instace of JButton) {

        //programar altre acció

        }

    }
```

Exercici ActionListener

Crea una classe que es digui CanviaTitol que extengui de la classe JFrame i implementi la interface ActionListener. Crea dos botons. Afegeix un listener a cada botó. Veurem alguns errors. Que ens diuen aquests errors?

Ara volem que en funció del botó polsat canviar el títol de la finestra. Fes-ho amb el mètode setTitle. Al acabar fes us del mètode repaint.

Els metodes mès habituals:

- addActionListener
- actionPerformed
- getActionCommand(): Per saber més informació sobre el component que genera el event.
- setActionCommand(): Podem modificar el text de una etiqueta.

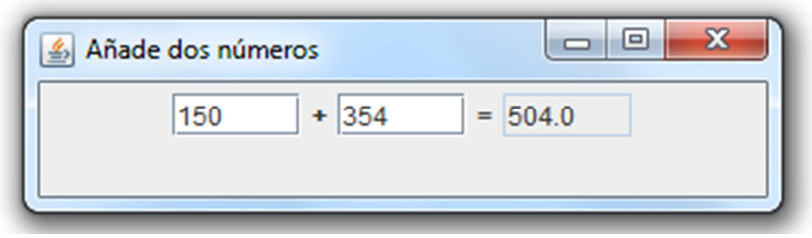
```
public void actionPerformed(ActionEvent evt) {
            b1.setActionCommand("prueba");
            b1.setLabel(b1.getActionCommand());
            repaint();
    }
```

FocusListener

Hem de implementar la interface FocusListener, afegir un listener al objecte amb el mètode addFocusListener i fer servir el focusGained() o focusLost() per saber quin objecte a guanyat o perdut el focus.

Exercici

Implementa la següent aplicació: Les caixes dels números son JTextField i el + i = son JLabels



Un cop estigui feta la finestra, anem a afegir la funcionalitat. Afegeix els FocusListener a les caixes dels operands. Implementa l'event focusGained() o focusLost() Al focusGained hauràs de sumar els valors i fer que la suma aparegui al recuadre resultat. Tingues en compte que recollim strings, per tant hauràs de fer un parser. Al mètode focusLost fes una crida al mètode focusGained.

ItemListener

Es produeix a list,radiobuttons checkboxes la diferència amb el focus es que aquí apuntem a items no a components. El mètode per afegir es el addItemListener i el mètode que haurem d'implementar el itemStateChanged. Per saber a quin item s'ha produit el event tenim el mètode getItem() i el mètode getStateChange() per saber si el item ha estat seleccionat o no. Retornant-nos DESELECTED i SELECTED.

```
package Events;


import java.awt.*;

import java.awt.event.*;

import javax.swing.*;
```

```
public class ElegirFormato extends JFrame implements ItemListener {

    //creamos String para ComboBox

    String[] formatos = { "(elige formato)", "Atom", "RSS 0.92",

        "RSS 1.0", "RSS 2.0" };

    //creamos String para etiquetas

    String[] descripciones = {

        "Formato de sindicación y weblog Atom",

        "Formato de sindicación RSS 0.92 (Netscape)",

        "RSS 1.0 (RDF)",

        "RSS 2.0 (RSS Selección Aconsejada)"

    };

    //Creamos Combo y Etiqueta

    JComboBox cajaFormato = new JComboBox();

    JLabel etiquetaDescripcion = new JLabel("");


    public ElegirFormato() {

        super("Formato de Sindicación");

        setSize(420, 125);

        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        setLayout(new BorderLayout());

        //añadimos a Combo items del String Formatos

        for (int i = 0; i < formatos.length; i++) {

            cajaFormato.addItem(formatos[i]);

        }

        cajaFormato.addItemListener(this); // Añadimos listener a Combo

        add(BorderLayout.NORTH, cajaFormato); //Añadimos Combo arriba en gestor
diseño

        add(BorderLayout.CENTER, etiquetaDescripcion);//Añadimos etiqueta a centro en
gestor diseño

        setVisible(true);

    }

    // implementamos método itemStateChanged de interface ItemListener

    public void itemStateChanged(ItemEvent event) {
```

```
        int eleccion = cajaFormato.getSelectedIndex();//obtenemos index del item
seleccionado

        if (eleccion > 0) {

             etiquetaDescripcion.setText(descripciones[eleccion-1]); //configuramos
texto de etiqueta según index del combo.

        }

    }







    public static void main(String[] arguments) {

        ElegirFormato fc = new ElegirFormato();

    }

}
```

KeyListener

Per detectar si s'ha polsat una tecla tenim en aquesta interface keyPressed, keyReleased i keyTyped.

```
package Events;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//creamos clase ComprobarTecla subclase de JFrame
public class ComprobarTecla extends JFrame {
    JLabel etiquetaTecla = new JLabel("Pulsa cualquier tecla");

    public ComprobarTecla() {
        super("Pulsa una tecla");
        setSize(300, 200);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLayout(new FlowLayout(FlowLayout.CENTER));
        MonitorearTecla monitor = new MonitorearTecla(this); //creamos objeto
de la clase MonitorearTecla,
        setFocusable(true);
        addKeyListener(monitor); //Añadimos Listener
        add(etiquetaTecla);
        setVisible(true);
    }

    public static void main(String[] arguments) {
        new ComprobarTecla();
    }
}
//creamos la clase MonitorearTecla, subclase de KeyAdapter. Sólo
implementamos un método de la interface KeyListener.
//implementa la interface a través de su superclase, así que no necesitamos
usar implements aquí de nuevo.
```

```
class MonitorearTecla extends KeyAdapter {
    ComprobarTecla display;

    MonitorearTecla(ComprobarTecla display) {
        this.display = display;
    }

    public void keyTyped(KeyEvent event) {
        display.etiquetaTecla.setText("" + event.getKeyChar());
        display.repaint();
    }
}
```

MouseListener

Per detectar events de mouse, teni els mètodes son getClickCount() per saber número de polsacions, getPoint per saber coordenades del punter o getX i getY

```
package Events;


import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
//clase AtrapaMouse implementa ActionListener con dos componentes (JLabel y
JButton) y gestor de diseño BorderLayout.
public class AtrapaMouse extends JFrame implements ActionListener {
    public AtrapaMouse() {
        super("Mensaje");
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setSize(420, 220);
        BorderLayout borde = new BorderLayout();
        setLayout(borde);
        JLabel mensaje = new JLabel("Pulsa Aceptar para cerrar este
programa.");
        add(BorderLayout.NORTH, mensaje);
        AtrapaPanel atrapa = new AtrapaPanel();
        atrapa.aceptar.addActionListener(this);
        add(BorderLayout.CENTER, atrapa);
        setVisible(true);
    }

    public void actionPerformed(ActionEvent event) {
        System.exit(0);
    }



    public static void main(String[] arguments) {
        new AtrapaMouse();
    }
}
//clase AtrapaPanel subclase de JPanel e implementa interface
MouseMotionListener
class AtrapaPanel extends JPanel implements MouseMotionListener {
```

```
    //colocamos botón con valores absolutos.
    JButton aceptar = new JButton("Aceptar");
    int botonX, botonY, ratonX, ratonY;
    int ancho, alto;

    AtrapaPanel() {
        super();
        setLayout(null);
        addMouseMotionListener(this);
        botonX = 110;
        botonY = 110;
        aceptar.setBounds(new Rectangle(botonX, botonY,90, 20));
        add(aceptar);
    }

    public void mouseMoved(MouseEvent event) {
        ratonX = event.getX();
        ratonY = event.getY();
        ancho = (int)getSize().getWidth();
        alto = (int)getSize().getHeight();
        if (Math.abs((ratonX + 35) - botonX) < 50) {
            botonX = moverBoton(ratonX, botonX, ancho);
            repaint();
        }
        if (Math.abs((ratonY + 10) - botonY) < 50) {
            botonY = moverBoton(ratonY, botonY, alto);
            repaint();
        }
    }

    public void mouseDragged(MouseEvent event) {
        // ignorar este evento
    }

    private int moverBoton(int ratonAt, int botonAt, int borde) {
        if (botonAt < ratonAt) {
            botonAt--;
        } else {
            botonAt++;
        }
        if (botonAt > (borde - 20)) {
            botonAt = 10;
        }
        if (botonAt < 0) {
            botonAt = borde - 80;
        }
        return botonAt;
    }

    public void paintComponent(Graphics comp) {
        super.paintComponent(comp);
        aceptar.setBounds(botonX, botonY, 90, 20);
    }
}
```