

Eventos

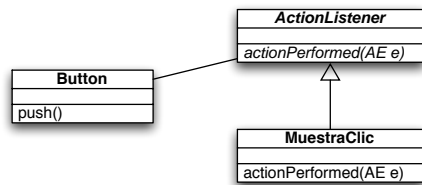
Eventos, acciones y ActionPerformed

Cátedra 18

Otoño 2005

1 El patrón de delegación

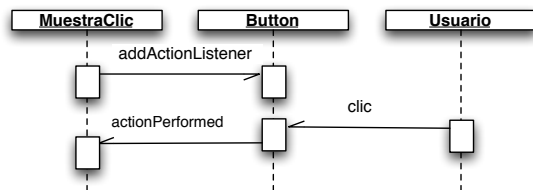
Todos los componentes tienen un evento principal. Por ejemplo, los botones son presionados y los campos de texto reciben un "enter". Cuando este evento ocurre, los componentes notifican a todos los interesados. La notificación se lleva a cabo mediante la invocación del método `actionPerformed`, que tiene que ser implementado por el objeto interesado (que llamaremos "suscriptor").



2 Suscripción

Como se acaba de mencionar, los suscriptores deben avisar en algún momento que quieren ser notificados de los eventos que ocurren en los componentes. Por ejemplo, si un objeto `muestraClic` quiere escribir **Clic** cada vez que el botón `b` es presionado, debería escribirse el siguiente código para que se le avise a `x` que `b` fue presionado:

```
b.addActionListener(muestraClic);
```



3 ActionPerformed

El objeto `muestraClic`, debe tener implementado el método `actionPerformed`, que es invocado cada vez que acaece un evento

```

class MuestraClic implements ActionListener {
    void actionPerformed(ActionEvent e) {
        con.println("Clic");
    }
}
  
```

El método recibe como único parámetro un objeto de la clase `ActionEvent`. Este objeto tiene toda la información del evento, particularmente quién fue la fuente del evento (`e.getSource()`).

4 Ejemplo completo

```

import java.awt.*;
import java.awt.event.*;

class Ejemplo {
    public static void main(String args[]) {
        Frame f=new Frame("Mi ventana");
        Contador c=new Contador();
        Button b=new Button("Hazme clic");
        f.add("Center",b);
        b.addActionListener(c);
        f.pack();
        f.show();
    }
}

class Contador implements ActionListener{
    int cont;
    Contador() {
        this.cont=0;
    }
}
  
```

```

    public void actionPerformed(ActionEvent e) {
        this.cont++;
        System.out.println("Me has hecho clic "+this.cont+" veces");
    }
}

```

Todo se podría poner también dentro de la misma clase

```

import java.awt.*;
import java.awt.event.*;
class Ejemplo implements ActionListener {
    int cont;
    public static void main(String args[]) {
        new Ejemplo();
    }
    Ejemplo () {
        this.cont=0;
        Frame f=new Frame();
        Button b=new Button("Hazme clic");
        f.add("Center",b);
        b.addActionListener(this);
        f.pack();
        f.show();
    }
    public void actionPerformed(ActionEvent e) {
        this.cont++;
        System.out.println("Me has hecho clic "+this.cont+" veces");
    }
}

```

5 Eventos en TextFields

Los TextFields generan los mismos eventos que los botones. Cada vez que uno presiona la tecla Enter dentro de un TextField se genera un evento que debe ser comunicado a los suscriptores.

AL MARGEN

Canvas

La clase abstracta Canvas representa objetos que se pintan a si mismos. Tienen dos métodos principales: `getPreferredSize`, que devuelve el tamaño del componente y `paint`, que se encarga de dibujar el objeto. El siguiente código dibuja un componente con forma de X:

```

class X extends Canvas {
    public Dimension getPreferredSize() {
        return new Dimension(100,100);
    }
    public void paint(Graphics g) {
        g.drawLine(0,0,100,100);
        g.drawLine(100,0,0,100);
    }
}

```

Otros Listeners

Los eventos de Ventana, como cerrar, abrir, iconificar son escuchados mediante un `WindowListener`.

Los eventos del Mouse, como un clic en un Canvas, pueden ser escuchados mediante un `MouseListener`. Eventos más sutiles, como saber cuando el mouse entra en un componente o ha sido movido, se pueden escuchar con `MouseMotionListener`.