ActionListener: como detectar acciones del usuario.

Auxiliar: Carla Marina Vairetti

Las aplicaciones han de responder a las acciones que realiza el usuario, por ejemplo: cerrar la ventana al pulsar un botón. Es necesario poder detectar las acciones de usuario y convertirlas en acciones de programa. Java dispone de varias formas de implementar este funcionamiento.
Las Interfaces son clases "prefabricadas" con propósitos definidos. Cada Interface contiene declaraciones de métodos (nombre del método, lista de argumentos y tipo de retorno) pero estos métodos no están implementados (carecen de cuerpo); todos los métodos de las Interfaces son public. Las Interfaces también pueden declarar constantes (public final static).

Importante:

> ActionListener es una Interface del grupo de los Listeners (escuchadores). ActionListener tiene un sólo método: void actionPerformed(ActionEvent e). ActionListener se usa para detectar y manejar eventos de acción (ActionEvent): los que tienen lugar cuando se produce una acción sobre un elemento del programa.

Un evento ActionEvent se produce:

- al pulsar un botón (Button)
- al hacer doble clic en un elemento de lista (List)
- al pulsar INTRO en una caja de texto (TextFiel)
- al elegir un menú (MenuItem)
- entre otros.

Los distintos elementos del programa (un botón, por ejemplo) están vigilados por Listeners que detectan las acciones que tienen lugar sobre el elemento vigilado. Cuando ActionListener detecta una acción (por ejemplo: pulsar sobre un botón) se genera un evento de acción (ActionEvent) en el elemento (botón). Los ActionEvent invocan el método actionPerformed(ActionEvent e) que realiza las acciones programadas ante ese evento.
Es necesario importar el paquete java.awt.event.*.

A tener en cuenta:

1. el botón sobre el que se produce una acción y genera un evento necesita estar previamente registrado con algún objeto que gestione el evento (ActionListener en este caso) mediante el método:

   ```
   boton.addActionListener(ActionListener nombre_de_actionListener)
   ```

2. el botón genera un ActionEvent (clase del paquete java.awt.event)

3. ActionEvent invoca al método actionPerformed que realiza las acciones programadas

4. si la clase implementa ActionListener, necesitamos implementar el método public void actionPerformed(ActionEvent e).

Distintas versiones:

- la clase pricipal implementa ActionListener:

```
…….
boton.addActionListener(this);
……
public void actionPerformed(ActionEvent e) {
... código ... ;
}
```

- la clase principal no implementa ActionListener pero sí lo hace otra clase:

```
……
boton.addActionListener(new ClaseClicEnBoton());
……
class ClaseClicEnBoton implements ActionListener {
public void actionPerformed(ActionEvent e) {
... código ... ;
 }
   }
```

- la clase principal no implementa ActionListener:

```
……
boton.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
    clicEnBoton(e); }
});
……
private void clicEnBoton(ActionEvent e) {
      ... código ... ;
   }
```

- en caso de tener que detectar el botón que origina un evento entre varios botones registrados con un ActionListener, puede ser útil recurrir al método setActionCommand, que establece un nombre, independiente de su etiqueta, para el botón:

```
boton1.setActionCommand ("nombre_ActionCommand1");
boton2.setActionCommand ("nombre_ActionCommand2");
```

Posteriormente se emplea una estructura condicional para elegir la acción dependiendo del botón origen del evento:

```
public void actionPerformed(ActionEvent e) {
   ……..

    if ("nombre_ActionCommand1".equals(e.getActionCommand())) {
     ... código de que se apreto por ejemplo el boton 1 ... ; }
    else if
         ("nombre_ActionCommand2".equals(e.getActionCommand()))
          {
              ... código del clic del boton 2 ... ; }
          }
```