Corso sul linguaggio Java

Modulo JAVA5

B1 – Gestione eventi 1

Prerequisiti

- Programmazione base in Java
- Utilizzo di classi e oggetti AWT o Swing
- Programmazione ad eventi

Introduzione

Le interfacce create con gli elementi grafici descritti nelle Unità A1 non sono interattive in quanto gli elementi grafici sono statici e non reagiscono alle azioni dell'utente (es. aprire o chiudere una finestra). In questa Unità vediamo come rendere attivi gli elementi grafici attraverso la **gestione degli eventi**.

Creare e gestire gli eventi

La programmazione **OOP** è basata sullo scambio di messaggi tra oggetti e questa caratteristica si manifesta anche quando si vogliono gestire eventi.

1. l'utente compie una **azione** su un **oggetto origine dell'evento** (ad es. invia un messaggio *chiudi* ad una finestra);
2. l'oggetto origine manda un messaggio ad un **generatore di eventi** (ad es. **WMS**, *Window Management System*, per le finestre) che a sua volta genera *automaticamente* un **oggetto evento**;
3. l'**oggetto evento** manda a sua volta un messaggio ad un altro oggetto, **oggetto ascoltatore** (*listener*); se il progettista ha implementato il codice relativo a questo oggetto ascoltatore e lo ha **registrato**, l'applicazione eseguirà questo codice.
4. l'oggetto ascoltatore, infine, manderà all'**oggetto destinatario** (deciso dal programmatore, ad esempio l'oggetto **output**) il messaggio che rappresenta l'effetto dell'azione eseguita dall'utente.

Le classi di ascolto

Mentre il punto 1. viene svolto dall'utente e il punto 2. viene automaticamente eseguito dal sistema di generazione degli eventi, al progettista spetta la realizzazione dei punti 3. e 4.

Java raccoglie gli ascoltatori in varie **classi di ascolto** aventi le seguenti caratteristiche:

- sono contenute in **java.awt.event**
- sono tutte **interfacce** (classi astratte in cui nessun metodo è implementato); il progettista deciderà, in base all'applicazione, quali metodi di ascolto implementare;
- una stessa classe può gestire eventi di diversi elementi grafici

Le classi di ascolto

Le classi d'ascolto di Java principali sono riassunte in tabella.

Come si nota il nome è composto dall'oggetto origine seguito dal suffisso *listener* per ricordarle facilmente.

Classe d'ascolto	Evento generato	Origine (*fire*)
FocusListener	Focus preso Focus perso	Tutti
MouseListener	Mouse entrato Mouse uscito Tasto mouse premuto Tasto mouse rilasciato Click	Tutti
MouseMotionListener	Mouse premuto e mosso Mouse mosso	Tutti
KeyListener	Tasto premuto Tasto rilasciato Tasto premuto e rilasciato	Tutti
WindowListener	Finestra attivata Finestra disattivata Finestra chiusa Chiudi finestra Riduci a icona Finestra aperta Finestra deiconificata	Window
WindowFocusListener	Fuoco preso Fuoco perso	Window
ActionListener	Azione su oggetto	Button TextField Menu
ItemListener	Click su un oggetto	Checkbox Choice CheckboxGroup Listbox
TextListener	Testo modificato	TextField TextArea
ContainerListener	Oggetto aggiunto Oggetto tolto	Tutti
ComponentListener	Oggetto nascosto Oggetto mostrato Oggetto ridimensionato Oggetto mosso	Tutti

Package da importare

Indifferentemente si può importare il *package*

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

oppure la sola parte relativa agli ascoltatori da creare (ad esempio per le sole finestre):

- **import java.awt.event.WindowListener;**
- **import java.awt.event.WindowEvent;**

WindowListener

Prevede i seguenti ascoltatori:

INTERFACCIA WindowListener	EVENTO
void windowActivated(WindowEvent e) { }	Finestra attivata
void windowDeactivated(WindowEvent e) { }	Finestra disattivata
void windowClosed(WindowEvent e) { }	Finestra chiusa
void windowIconified(WindowEvent e) { }	Riduci a icona
void windowOpened(WindowEvent e) { }	Finestra aperta
void windowDeiconified(WindowEvent e) { }	Finestra deiconificata
void windowClosing(WindowEvent e)	Chiudi finestra

WindowListener

Vogliamo realizzare un ascoltatore per attivare il pulsante di chiusura di una finestra.

Schema dell'ascoltatore

Origine: finestra
Interfaccia: **WindowListener**
Evento: windowClosing(WindowEvent e)
Gestore di evento: gestoreF()
Registrazione: addWindowListener (**new** gestoreF());

Poiché le classi di ascolto sono interfacce, dobbiamo comunque implementare tutti i metodi, lasciando vuoti quelli eventualmente non usati.

WindowListener

implements si usa per ereditare le interfacce

Fine programma

```
// gestoreF.java
import java.awt.*;
import java.awt.event.*;
public class gestoreF implements WindowListener
  { public void windowClosing(WindowEvent e) { System.exit(0); }
    public void windowIconified(WindowEvent e) { }
    public void windowDeiconified(WindowEvent e) { }
    public void windowActivated(WindowEvent e) { }
    public void windowDeactivated(WindowEvent e) { }
    public void windowOpened(WindowEvent e) { }
    public void windowClosed(WindowEvent e) { }
  }
```

WindowListener

L'applicazione che usa il *gestoreF* ha la struttura seguente:

```
// window.java
import java.awt.*;
import java.awt.event.*;
public class Window extends Frame
{    public Window()
     {    setTitle("Gestione finestra");
          setLocation(200,200);
          setSize(200,200);
          setVisible(true);
          addWindowListener (new gestoreF());
     }
     public static void main(String args[])
     {
          Window w = new Window();
     }
}
```

Impostazioni finestra

Registrazione dell'ascoltatore

WindowAdapter

In questo esempio l'ascoltatore è inserito nella classe dell'applicazione.

```
import java.awt.*;
import java.awt.event.*;
public class Window extends Frame
{    public Window()
     {    setTitle("Gestione finestra");
          setLocation(200,200);
          setSize(200,200);
          setVisible(true);
          addWindowListener (new gestoreF());
     }
     public class gestoreF extends WindowAdapter
     {    public void windowClosing(WindowEvent e)
          {         System.exit(0);    }
     }
     public static void main (String args[])
     {    new Window();    }
}
```

Per evitare la scrittura dei metodi vuoti si usano le classi **Adapter**

WindowFocusListener

Consideriamo questa classe d'ascolto per vedere un'altra caratteristica del gestore di eventi, sempre quando si scrive l'applicazione come un unico file.
Gli ascoltatori sono i seguenti:

INTERFACCIA WindowFocusListener	EVENTO
void windowGainedFocus(WindowEvent e) { }	Fuoco preso
void windowLostFocus(WindowEvent e) { }	Fuoco perso

Origine: finestra
Interfaccia: WindowFocusListener
Evento: windowGainedFocus (WindowEvent e)
windowLostFocus (**WindowEvent** e)
Gestore di evento: la classe stessa
Registrazione: addWindowFocusListener (**this**);

WindowFocusListener

La classe è ascoltatore di se stessa

```
import java.awt.*;
import java.awt.event.*;
public class gestoreFocus extends Frame implements WindowFocusListener
{   TextField stato=new TextField ();
    public gestoreFocus()
    {   super("Prova focus");  setSize (200,200);  setVisible (true);
        stato.setBounds (20,40, 100,20);  add (stato);
        addWindowFocusListener (this);
        addWindowListener (new gestoreF());
    }
    public void windowGainedFocus (WindowEvent e)
    {   stato.setText ("Focus on!");  }
    public void windowLostFocus (WindowEvent e)
    {   stato.setText("Focus off!"); }
    public static void main (String args[])
    {   new gestoreFocus(); }
}
```

Utilizza anche *gestoreF* creato in precedenza

Gli ascoltatori vengono implementati direttamente nella classe

ActionListener

Questa interfaccia è in grado di gestire gli eventi relativi a:

- pulsanti
- caselle di testo
- menu

Nonostante possa servire per rendere interattivi diversi elementi, questa classe prevede un solo metodo:

INTERFACCIA ActionListener	EVENTO
void ActionPerformed (ActionEvent e)	Azione sul pulsante

ActionListener

Pulsanti

Creiamo una finestra con un pulsante che, quando premuto, restituisca il nome in una casella di testo

Origine: pulsante
Interfaccia: **ActionListener**
Evento: actionPerformed (**ActionEvent** e)
Gestore di evento: gestorePulsante()
Registrazione: addActionListener (**new** gestorePulsante())

ActionListener

Pulsanti

```
import java.awt.event.*;
import java.awt.*;
public class gestorePulsante implements ActionListener
{  TextField txt;   // attributo privato necessario al costruttore
   public gestorePulsante (TextField tf)
   {  this.txt=tf;   }
   public void actionPerformed (ActionEvent e)
   {       txt.setText("Premuto " + e.getActionCommand()); }
}
```

Necessita un costruttore con parametri per passare il nome della casella di testo al gestore

Restituisce il nome del pulsante premuto, per poter svolgere azioni successive

ActionListener

Pulsanti

ATTIVITA': scrivere un'applicazione che crei una finestra con controllo sulla chiusura e due pulsanti **Annulla** e **Esci** che rispettivamente, fanno continuare l'esecuzione oppure la terminano.

```
import java.awt.*;
public class ButtonEvent extends Frame
{  Button btn = new Button("Premi");
   TextField btn_txt = new TextField (50);
   ButtonEvent()
   { super("Gestione pulsante");
     setLocation (300,200);  setSize (300,100);  setLayout (null);
     btn.setBounds (10, 30, 60,30);
     btn_txt.setBounds(50, 70, 170, 120);
     add(btn_txt);  add(btn);
     btn.addActionListener(new gestorePulsante(btn_txt));
     setVisible(true);
   }
   public static void main (String args[])
   { new ButtonEvent();    }
}
```

Impostazioni frame

Registrazione ascoltatore con parametro

ActionListener

Caselle di testo

Creiamo una finestra con due caselle di testo editabili, in modo che quando l'utente digita il testo in una di queste e prema INVIO, venga visualizzato il contenuto attuale della casella selezionata.

Origine: casella di testo

Interfaccia: **ActionListener**

Evento: actionPerformed (**ActionEvent** e)

Gestore di evento: gestoreCasella()

Registrazione: addActionListener (**new** gestoreCasella())

ActionListener

Caselle di testo

```
import java.awt.event.*;
public class gestoreCasella implements ActionListener
{
    public void actionPerformed (ActionEvent e)
    {
        System.out.println("Dato inserito: " + e.getActionCommand());
        System.out.println (e.getSource());
    }
}
```

Restituisce gli attributi della casella (dimensioni, nome, contenuto, stato, ecc)

Restituisce il contenuto della casella attiva, dopo aver premuto INVIO

ActionListener

Caselle di testo

```
import java.awt.*;
public class TextFieldEvent extends Frame
{  TextField cognome_txt = new TextField ("Cognome");
   TextField nome_txt = new TextField("Nome");
   TextFieldEvent()
   { super("Gestione caselle di testo");  setLocation (200,200);
     setSize (100,100);
     setLayout(null);
     cognome_txt.setBounds(10, 30, 60,30);nome_txt.setBounds(10,60, 60,30);
     add (cognome_txt);  add (nome_txt);
     cognome_txt.addActionListener (new gestoreCasella());
     nome_txt.addActionListener (new gestoreCasella());
     setVisible(true);
   }
   public static void main(String args[])
   { TextFieldEvent t = new TextFieldEvent();  }
}
```


ActionListener

Menu

Creiamo una finestra con un menu di opzioni da ascoltare.

Origine: menu
Interfaccia: **ActionListener**
Evento: actionPerformed (**ActionEvent** e)
Gestore di evento: gestoreMenu()
Registrazione: addActionListener (**new** gestoreMenu())

ActionListener

Menu

```
import java.awt.*;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
public class MenuSemplice extends Frame
{  public MenuSemplice()
   { super("Ascolto di menu");
     Panel p =new Panel();
     setLocation (200,100);
     setSize (300,150);
     setVisible (true);
     MenuBar menu= new MenuBar(); // impostazione menu
     creazione menu File e menu Help con alcune opzioni
     file.addActionListener(new gestoreMenu()); // registrazione ascoltatori
     help.addActionListener(new gestoreMenu());
   }
…..segue classe….
```


ActionListener

Menu

```
  public class gestoreMenu implements ActionListener
  {
    public void actionPerformed(ActionEvent evt)
    {     String s=evt.getActionCommand();
          System.out.println (s);
          if (s.equals("Exit"))
                System.exit(0);
    }
  }
  public static void main (String args[])
  {
    new MenuSemplice();
  }
}
```

Rileva l'opzione selezionata e la stampa. In caso di "Exit" il programma termina

Argomenti

- Creare e gestire gli eventi
- Le classi di ascolto
- *Package* da importare
- WindowListener
- WindowAdapter
- WindowFocusListener
- ActionListener

Altre fonti di informazione

- P.Gallo, F.Salerno – Java, ed. Minerva Italica
- M. Bigatti – Il linguaggio Java, ed. Hoepli