Java Swing ed Eventi

Prof. Francesco Accarino
IIS Altiero Spinelli via Leopardi 132
Sesto san Giovanni

Gestione degli eventi

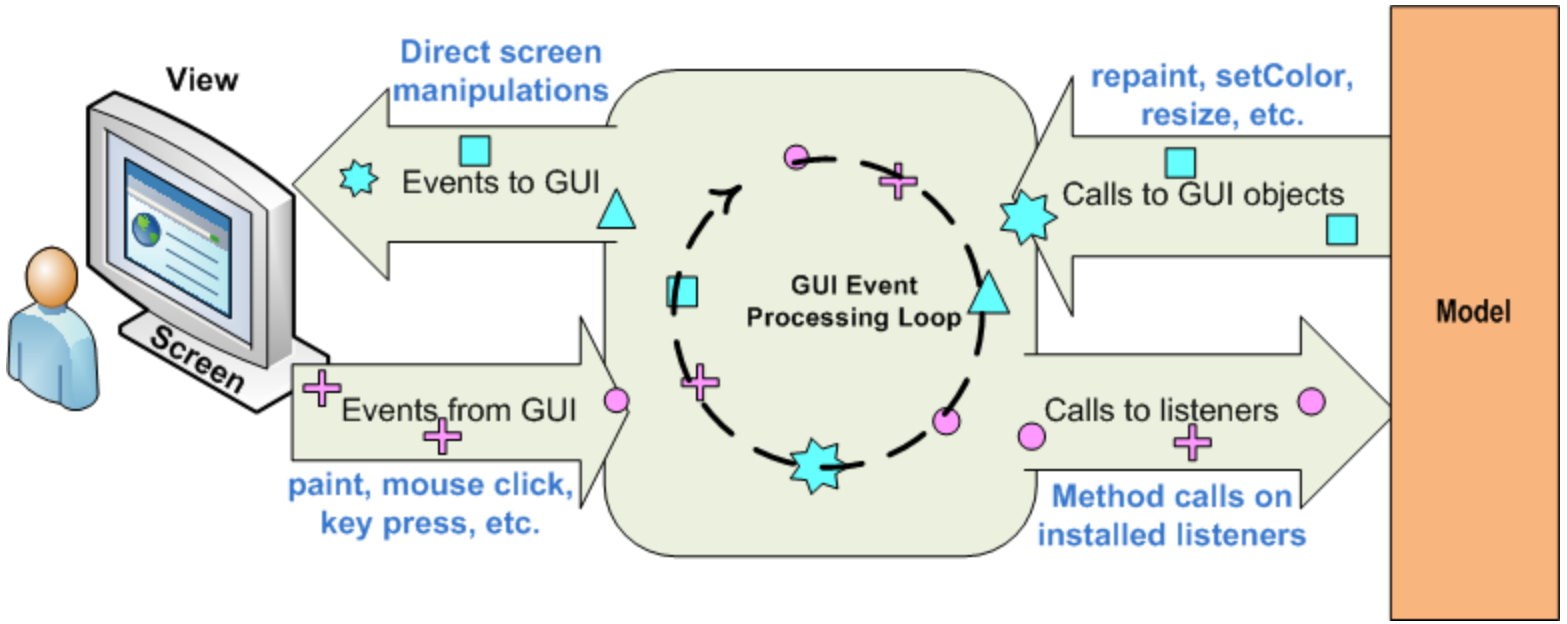
- Per gestione degli eventi, intendiamo la possibilità di associare l'esecuzione di una certa parte di codice, in corrispondenza ad un certo evento sulla GUI.
- Un esempio di evento potrebbe essere la pressione di un bottone.
- Questa caratteristica viene implementata con il cosiddetto modello a delega (Event Delegation Model)

Event Delegation Model

- In questo modello ci sono tre soggetti:
  - Event Source: è l'oggetto il cui stato cambia e genera l'evento
  - Event Object: è l'evento che incapsula il cambiamento di stato dell'Event Source
  - Event Listener: è l'oggetto che viene delegato dall'Event Source a svolgere l'attività di ascolto degli eventi e alla notifica di essi.

Event Delegation Model

- Il tutto è coordinato dalla JVM che effettua il cosiddetto Event Proccessing Loop



Event Delegation Model

- Per esempio se premiamo su di un bottone e vogliamo che appaia una scritta su unaLabel della stessa interfaccia, allora:

  1) il bottone è la sorgente dell'evento

  2) l'evento è la pressione del bottone, che sarà un oggetto istanziato direttamente dalla JVM dalla classe ActionEvent

  3) il gestore dell'evento sarà un oggetto istanziato da una classe a parte che implementa un interfaccia ActionListener (in italiano "ascoltatore d'azioni"). Quest'ultima ridefinirà il metodo actionPerformed(ActionEvent ev), con il quale sarà gestito l'evento.

la JVM, invocherà automaticamente questo metodo su quest'oggetto, quando l'utente premerà il bottone.

Sviluppare un applicazione GUI con eventi

- Le operazioni da svolgere sono le seguenti:

  1) creare la GUI
  2) creare un listener
  3) registrare il componente interessato con il rispettivo listener.

Esempio

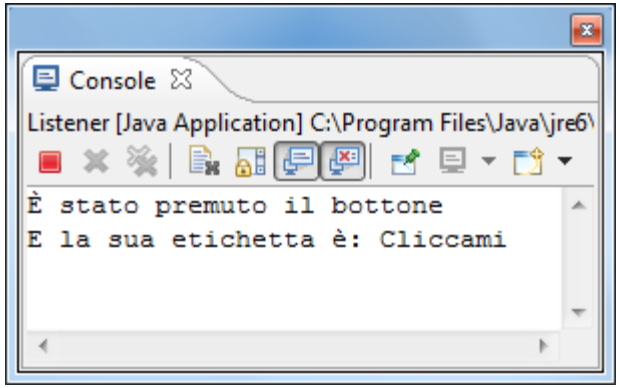
```
import java.awt.event.*;
public class ButtonHandler implements ActionListener
{
     public void actionPerformed(ActionEvent e) {
     System.out.println("È stato premuto il
     bottone");
     System.out.println("E la sua etichetta è: "
     + e.getActionCommand());}
     }
```

Oggetto Listener

```
import java.awt.BorderLayout;
import javax.swing.*;
public class Listener {
private JFrame f;
private JButton b;
public Listener() {
f = new JFrame("ProvaListener");
b = new JButton("Cliccami");
setup();
}
public void setup() {
b.addActionListener(new ButtonHandler());
f.add(b,BorderLayout.CENTER);
f.pack();
f.setVisible(true);
}
public static void main(String args[]) {
new Listener();
}
}
```

Creazione GUI

Registrazione Componente con il suo listener



Principali tipi di eventi e gestori

Evento	Descrizione	Interfaccia	Metodi
`ActionEvent`	Azione (generica)	`ActionListener`	`actionPerformed`
`ItemEvent`	Selezione	`ItemListener`	`itemStateChanged`

`MouseEvent`	Azioni effettuate con il mouse	`MouseListener`	`mousePressed`
			`mouseReleased`
			`mouseEntered`
			`mouseExited`
			`mouseClicked`

`MouseEvent`	Movimenti del mouse	`MouseMotionListener`	`mouseDragged` `mouseMoved`

`KeyEvent`	Pressione di tasti	`KeyListener`	`keyPressed`
			`keyReleased`
			`keyTyped`

`WindowEvent`	Azioni effettuate su finestre	`WindowListener`	`windowClosing`
			`windowOpened`
			`windowIconified`
			`windowDeiconified`

Event Listener (uso di *Inner Class*)

```
Import javax.swing.*;
import java.awt.*;
import java.awt.event.*;
public class MioEvento {
        JFrame fr;      // visibili all'interno della classe evento
        JButton b1;     // quindi anche nella classe INNER mioButt
        public static void main(String[] args) {
                MioEvento e = new MioEvento();
                e.apri();
        }
        void apri() {
                fr = new JFrame("Mio Evento");
                Container cp = fr.getContentPane();
                cp.setLayout( new FlowLayout());
                b1 = new JButton("On");
                b1.addActionListener( new mioButtListener() );
                fr.add( b1 );     // aggiungo il bottone al Frame
                fr.setVisible(true);
        }
        class mioButtListener implements ActionListener {
                public void actionPerformed( ActionEvent e) {
                        if ( b1.getText().equals("On") )   b1.setText("Off");
                        else                               b1.setText("On");
                }
        }
}
```

Aggiungere per la gestione degli eventi

Creazione e registrazione EventListener

Creazione del Gestore dell'evento

Implementazione del metodo

Event Listener: schema riassuntivo

mioButtListener implementa l'interfaccia ActionListener

L'interfaccia ActionListener ha il solo metodo actionPerformed() che sta per: *"questa è l'azione da compiere quando accade questo evento"*

L'argomento del metodo actionPerformed è un oggetto di tipo java.awt.event.ActionEvent

Quando l'utente preme il bottone

- viene creato un oggetto istanza di ActionEvent
- viene eseguito il metodo actionPerformed con tale oggetto come parametro

EventListener: classi adapter

- L'interfaccia *ActionListener* ha il solo metodo *ActionPerformed*. Ci sono interfacce con più di un metodi, come *MouseListener*;
- Le classi che estendono tali interfacce devono definire tutti di metodi, anche se non li usano.
- Per esse esistono le classi *adapter* che:
  - implementano tali interfacce
  - definiscono tutti i loro metodi, vuoti
- Il programmatore può:
  - scrivere una classe event listener che estende la classe *adapter e* ridefinire (*override*) solo i metodi di suo interesse
    - class mioListener extends MouseAdapter
  - Implementare l'interface e definire TUTTI i metodi
    - class mioListener implements MouseListener