Corso sul linguaggio Java

Modulo JAVA5

B2 – Gestione eventi (2)

Prerequisiti

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

Introduzione

Continuiamo la trattazione della programmazione ad eventi, descrivendo le classi di ascolto di altri componenti grafici di AWT.

In particolare, affrontiamo le classi di ascolto per la gestione del mouse, delle caselle e aree di testo, della tastiera e delle liste.

MouseListener

Questa classe consente di gestire gli spostamenti del mouse e prevede i seguenti ascoltatori:

INTERFACCIA MouseListener	EVENTO
public void mouseClicked (MouseEvent e)	Click del mouse
public void mouseEntered (MouseEvent e)	Mouse entrato
public void mouseExited (MouseEvent e)	Mouse uscito
public void mousePressed (MouseEvent e)	Tasto mouse premuto
public void mouseReleased (MouseEvent e)	Tasto mouse rilasciato

Analogamente alla **WindowListener**, è possibile utilizzare **MouseAdapter** per evitare di riscrivere anche i metodi vuoti.

MouseListener

Creiamo una finestra con un ascoltatore per il mouse e uno per l'uscita dal programma..

Origine: mouse, finestra
Interfaccia: **WindowListener, MouseListener**
Evento: **mouseClicked** (**MouseEvent** e)
mouseEntered (**MouseEvent** e)
mouseExited (**MouseEvent** e)
mousePressed (**MouseEvent** e)
mouseReleased (**MouseEvent** e)
windowClosing (**Windowevent** e)
Gestore di evento: gestoreMouse(), gestoreF()
Registrazione: **addMouseListener** (**new** gestoreMouse())
addWindowListener (**new** gestoreF())

MouseListener

```
import java.awt.*;
import java.awt.event.*;
public class Mouse extends Frame
{  public Mouse()
   { setTitle("Gestione finestra");
     setLocation (200,200);
     setSize (200,200);
     setVisible (true);
     addWindowListener(new gestoreF());
     addMouseListener(new gestoreM());
   }
   public class gestoreF extends WindowAdapter
   { public void windowClosing (WindowEvent e)
     {   System.exit(0);     }
   }
…segue classe
```

MouseListener

```
public class gestoreMouse implements MouseListener
{ public void mouseClicked(MouseEvent e)
  {System.out.println("Tasto premuto"); System.out.println(e.getButton()); }
  public void mouseEntered(MouseEvent e)
  {   System.out.println("Mouse entrato"); System.out.println(e.getX()+" "+ e.getY());
}
  public void mouseExited(MouseEvent e)
  { System.out.println("Mouse uscito");  }
  public void mousePressed(MouseEvent e)
  { System.out.println("Mouse premuto");  }
  public void mouseReleased(MouseEvent e)
  { System.out.println("Mouse rilasciato");  }
}
public static void main (String args[])
{ Mouse f = new Mouse();   }
} // end class
```


TextListener

Questa classe consente di gestire le modifiche in una casella di testo e prevede il seguente ascoltatore:

INTERFACCIA TextListener	EVENTO
void textValueChanged (TextEvent e)	Testo modificato

Questo ascoltatore ci consente di gestire qualunque cambiamento avvenuto all'interno della casella (modifica del testo)

TextListener

Vogliamo realizzare un ascoltatore per il cambiamento del contenuto di una casella di testo.

Origine: casella di testo
Interfaccia: **TextListener**
Evento: textValueChanged (**TextEvent** e)
Gestore di evento: classe stessa
Registrazione: text.addTextListener(this)

TextListener

```
import java.awt.*;
import java.awt.event.*;
import java.io.*;
class TestTextListener extends Frame implements TextListener
{
   TextField text = new TextField (20);
   TestTextListener()
   {
     super ("TextListener Example");
     text.setText ("Questo e' il testo");
     text.addTextListener(this);
     add (text, BorderLayout.CENTER); // posizionamento nel frame.
     setFont (new Font (“Courier", Font.PLAIN, 12));
     pack();
     setVisible(true);
   }    …..segue classe
```

TextListener

```
  public void textValueChanged (TextEvent e)
  {
    System.out.println("Vecchio testo: " + text.getText());
    text.setText("Ora il testo cambia");
    System.out.println("Nuovo testo: " + text.getText());
    text.removeTextListener(this);
  }
 public static void main(String args[])
  {
    new TestTextListener();
  }
}  // end class
```

Una volta servito l'evento, l'ascoltatore può essere rimosso dalla lista degli ascoltatori con il metodo **removeTextListener**()

KeyListener

Questa classe consente di gestire gli eventi prodotto dalla tastiera e prevede i seguenti ascoltatori:

INTERFACCIA KeyListener	EVENTO
void keyPressed (**KeyEvent** e)	Testo premuto
void keyReleased (**KeyEvent** e)	Tasto rilasciato
void keyTyped (**KeyEvent** e)	Tasto premuto e rilasciato

Analogamente alla **WindowListener**, è possibile utilizzare **KeyAdapter** per evitare di riscrivere anche i metodi vuoti.

KeyListener

Vogliamo realizzare un ascoltatore per la pressione di un tasto ed il suo riconoscimento.

Origine: tastiera
Interfaccia: **KeyListener**
Evento: keyPressed (**KeyEvent** evt)
Gestore di evento: KeyEventHandler()
Registrazione: addKeyListener (**new** KeyEventHandler())

KeyListener

```
import java.awt.*;
import java.awt.event.*;
class TestKeyListener extends Frame
{
   TextField tf = new TextField (40);
   TestKeyListener()
   {  super (“KeyListener Example");
      add (tf, BorderLayout.NORTH);
      tf.addKeyListener (new KeyEventHandler());
      pack ();
      setVisible (true);
   }
…segue classe
```

KeyListener

```
class KeyEventHandler implements KeyListener
{
   public void keyPressed (KeyEvent evt)
   {  int key = evt.getKeyCode();
      char ch = evt.getKeyChar();
      System.out.println("evt.getKeyCode(): " + key);
      System.out.println("evt.getKeyChar(): " + ch);
      System.out.println("evt.paramString(): " + evt.paramString());
      System.out.println("evt.getSource(): " + evt.getSource());
   }
      public void keyReleased(KeyEvent evt) { }
      public void keyTyped(KeyEvent evt) { }
}

static public void main(String[] args)
{     new TestKeyListener();       }
}
```

Codice ASCII carattere premuto

carattere premuto

Caratteristiche dell'evento

Elemento origine dell'evento

ItemListener

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

- listbox
- combobox
- checkbox
- menuitem

INTERFACCIA ItemListener	EVENTO
void itemStateChanged(ItemEvent e)	Click su un oggetto

ItemListener

Vogliamo realizzare un ascoltatore per la selezione di opzioni in una checkbox ed il loro riconoscimento.

Origine: checkbox
Interfaccia: **ItemListener**
Evento: itemStateChanged (**ItemEvent** evt)
Gestore di evento: gestoreChk()
Registrazione: addItemListener (**new** gestoreChk())

ItemListener

```
import java.awt.*;
import java.awt.event.ItemListener;
import java.awt.event.ItemEvent;
class TestCheckBox extends Frame
{  Checkbox ac_chk = new Checkbox("Aria condizionata");
   Checkbox ta_chk = new Checkbox("Tetto apribile");
   Checkbox ss_chk = new Checkbox("Servosterzo");
   Checkbox ve_chk = new Checkbox("Vetri elettrici");
   Label status = new Label("Totale prezzo: e " + 20000);
   TestCheckBox()
   {  super("Esempio ascoltatore ItemListener");
      impostazione frame;
      Panel gridPanel = new Panel (new GridLayout (0, 1, 50,20));
      aggiunta elementi al pannello;
      posizionamento pannello al centro;
      registrazione ascoltatori;
}
```

Impostazione checkbox

ItemListener

```
public class gestoreChk implements ItemListener
{
  public void itemStateChanged(ItemEvent evt)
{         computeTotal();   }
   void computeTotal()
   {
       int total = 25000;
       if (ac_chk.getState()) total += 510;
       if (ta_chk.getState()) total += 822;
       if (ss_chk.getState()) total += 150;
       if (ve_chk.getState()) total += 320;
       status.setText("Totale prezzo: e " + total);
   }
}
static public void main(String[] args)
{    new TestCheckBox();      }
} // end class
```

Metodo da eseguire in corrispondenza all'evento

Argomenti

- MouseListener
- TextListener
- KeyListener
- ItemListener

Altre fonti di informazione

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