



UNIVERSITÀ DEGLI STUDI DI SALERNO
DIPARTIMENTO DI INFORMATICA
DIPARTIMENTO DI ECCELLENZA

Università degli Studi di Salerno

Dipartimento di Informatica

Programmazione ad Oggetti

a.a. 2023-2024

Layouts e Gestione Eventi

Docente: Prof. Massimo Ficco

E-mail: mficco@unisa.it

Gestione eventi

Realizzazione di una interfaccia grafica:

- Disegno grafico: CAD
- Gestione degli eventi Scrittura codice

Ogni componente Swing è in grado di generare degli eventi generici o particolari (ad esempio per il pulsante non siamo particolarmente interessati al fatto che sia attraversato dal mouse, ma solo al click)

Quindi cominceremo col gestire per ogni componente gli eventi principali

Ogni volta che l'utente esegue un'azione (su elementi dell'interfaccia grafica) es. un clic del mouse, la pressione di un tasto sulla tastiera, la modifica di una finestra, la selezione di un elemento da un menu, ... viene generato un evento



Eventi: sorgenti e ricevitori

- **Sorgente di eventi (source):** componente di un'interfaccia grafica che può generare eventi
- **Ricevitore di eventi (listener):** reagisce al verificarsi di eventi attraverso le azioni descritte ne i suoi metodi
- Meccanismo di funzionamento:
 - si implementa un ricevitore per ogni evento che si vuole gestire
 - le istanze di ricevitori vengono collegate alle istanze delle sorgenti di interesse

quando viene generato un evento, tutti ricevitori per quel tipo di evento che sono collegati alla sorgente vengono attivati



Package java.awt.event

- Ogni tipo di evento è descritto da una classe
- Il package `java.awt.event` contiene
 - Le classi per i diversi tipi di eventi
 - Le interfacce relative ai ricevitori di eventi [Listener](#)
 - Alcune classi di adattatori, gli [Adapter](#) che implementano le interfacce [Listener](#)



Tipi di Eventi

- **MouseEvent** (eventi del mouse)
 - click del mouse, spostamento del mouse, etc
- **ActionEvent** (eventi di azione)
 - azioni di specifiche componenti, ad es. pressione di un pulsante
 - può essere generato anche premendo la barra spaziatrice col mouse posizionato su un pulsante
- **AdjustmentEvent** (eventi di modifica)
 - eventi emessi da oggetti Adjustable (che hanno un valore numerico modificabile (ad es. JScrollBar))



Tipi di Eventi

- **FocusEvent** (guadagnare/perdere input focus)
 - generato da componenti quali TextField
- **ItemEvent** (selezione/deselezione elemento)
 - generato da elementi di tipo ItemSelectable (ad es. Button)
- **KeyEvent** (eventi di tastiera)
 - pressione di un tasto su una componente grafica (ad es. TextField)
- **WindowEvent** (eventi relativi a finestre)
 - ad es una finestra dell'interfaccia grafica ha cambiato il suo stato
 - Etc ...



Interfacce Listener

Sono i ricevitori di eventi

- Esiste una interfaccia [Listener](#) per ciascun tipo di evento
- Definiscono i metodi che devono essere implementati da un ricevitore di eventi
 - MouseListener
 - ActionListener
 - AdjustmentListener
 - FocusListener
 - ItemListener
 - KeyListener
 - WindowListener
 - Etc...



ActionListener e JButton

Esempio:

- usare oggetti `JButton` per definire pulsanti
- collegare un `ActionListener` a ogni pulsante

Interfaccia `ActionListener`

```
public interface ActionListener
{
    void actionPerformed(ActionEvent event) ;
}
```

- Serve una classe che implementi l'interfaccia
 - l'implementazione di `actionPerformed` contiene le istruzioni da eseguire quando il pulsante viene pressato



ActionListener e JButton

File *ClickListener.java*

```
01: import java.awt.event.ActionEvent;
02: import java.awt.event.ActionListener;
03:
04: /**
05:     An action listener that prints a message.
06: */
07: public class ClickListener implements ActionListener
08: {
09:     public void actionPerformed(ActionEvent event)
10:     {
11:         System.out.println("I was clicked.");
12:     }
13: }
```

Tale classe per poter essere utilizzata da JButton deve implementare l'interfaccia ActionListener



ActionListener e JButton

- Il parametro `event` contiene dettagli relativi all'evento, quali ad esempio il tempo al quale si è manifestato
- Per collegare una sorgente di eventi ad un ricevitore occorre aggiungere il ricevitore alla sorgente di eventi:

```
ActionListener listener = new ClickListener();  
button.addActionListener(listener);
```

Quando viene pigiato il pulsante automaticamente viene invocato il metodo `ActionPerformed`



Esempio – Button2.java

```
import java.awt.event.ActionListener;  
import javax.swing.JButton;  
import javax.swing.JFrame;
```

```
public class Button2 extends JFrame {  
    private JButton b1 = new JButton("Button 1");  
    JButton b2 = new JButton("Button 2");  
    private JTextField txt = new JTextField(10);
```

```
    public Button2(){  
        super("2 pulsanti");  
        setSize(400,300);  
        JPanel p=new JPanel();  
        ActionListener listener = new ClickListener();  
        b1.addActionListener(listener);  
        b2.addActionListener(new ClickListener());  
        p.add(b1); p.add(b2); p.add(txt);
```

```
    public static void main(String[] args){  
        JFrame frame=new Button2();  
        frame.show();  
    }}
```

Nota: Quando la classe ClickListener utilizza variabili o oggetti interne della classe Button2, deve essere aggiunta come classe interna.

```
public class ClickListener implements ActionListener  
{  
    public void actionPerformed(ActionEvent event)  
    {  
        System.out.println("I was clicked.");  
    }  
}
```



Esempio – Button2.java

Output:



A screenshot of a terminal window titled "Terminal" with a menu bar containing "File", "Edit", "View", "Terminal", "Go", and "Help". The terminal shows the following commands and output:

```
~$ cd BigJava/ch12/button1
~/BigJava/ch12/button1$ java ButtonTester
I was clicked.
I was clicked.
I was clicked.
█
```

Overlaid on the terminal is a small GUI window with a title bar and a button labeled "Click me!".



*Esempio

```
class ButtonListener implements ActionListener {  
    JTextField txt;  
  
    public ButtonListener(JTextField t){txt=t;}  
  
    public void actionPerformed(ActionEvent e) {  
        String name = ((JButton)e.getSource()).getText();  
        txt.setText(name);  
    }  
}
```



Eventi del Mouse

- Per catturare eventi del mouse si usa un `MouseListener`
- L'interfaccia `MouseListener`

```
public interface MouseListener
{
    void mousePressed(MouseEvent event);
    void mouseReleased(MouseEvent event);
    void mouseClicked(MouseEvent event);
    void mouseEntered(MouseEvent event);
    void mouseExited(MouseEvent event);
}
```



Eventi del Mouse

Si aggiunge un `MouseListener` ad una componente con il metodo `addMouseListener`

```
public class MyMouseListener implements MouseListener
{
    // Implementa i cinque metodi
}
MouseListener listener = new MyMouseListener();
component.addMouseListener(listener);
```



Implementazione MyMouseListener

```
class MyMouseListener implements MouseListener
{
    public void mousePressed(MouseEvent event)
    {
        int x = event.getX();
        int y = event.getY();
        component.moveTo(x, y);
    }
    // Do-nothing methods
    public void mouseReleased(MouseEvent event) {}
    public void mouseClicked(MouseEvent event) {}
    public void mouseEntered(MouseEvent event) {}
    public void mouseExited(MouseEvent event) {}
}
```

- Tutti i metodi devono essere implementati
- Metodi inutilizzati possono restare vuoti



Esempio - RectangleComponentViewer2.java

```
01: import java.awt.event.MouseListener;
02: import java.awt.event.MouseEvent;
03: import javax.swing.JFrame;
04:
05: /**
06:  * This program displays a RectangleComponent.
07:  */
08: public class RectangleComponentViewer2
09: {
10:     public static void main(String[] args)
11:     {
12:         final RectangleComponent component
13:             = new RectangleComponent();
14:
15:         // Add mouse press listener
16:
17:         class MousePressListener implements MouseListener
18:         {
19:             public void mousePressed(MouseEvent event)
20:             {
21:                 int x = event.getX();
22:                 int y = event.getY();
23:                 component.moveTo(x, y);
24:             }
25:
26:             // Do-nothing methods
27:             public void mouseReleased(MouseEvent event) {}
28:             public void mouseClicked(MouseEvent event) {}
29:             public void mouseEntered(MouseEvent event) {}
30:             public void mouseExited(MouseEvent event) {}
31:         }
32:
33:         MouseListener listener = new MousePressListener();
34:         component.addMouseListener(listener);
```

- La classe **RectangleComponent** è una classe esterna
- La classe **MousePressListener** viene aggiunta come classe interna per poter utilizzare le variabili (final) component

```
35:         JFrame frame = new JFrame();
36:         frame.add(component);
37:
38:         frame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
39:         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
40:         frame.setVisible(true);
41:     }
42:
43:     private static final int FRAME_WIDTH = 300;
44:     private static final int FRAME_HEIGHT = 400;
45: }
```

Utilizzare Listener Adapters

Se volessi implementare una interfaccia listener dovrei impazzire poiché non dovrei dimenticare nessun metodo

Per ogni listener esiste un adapter che implementa l'interfaccia corrispondente con tutti metodi vuoti

Esempio:

```
class MyMouseListener extends MouseAdapter {  
    public voidMouseClicked(MouseEvent e) {  
        // Respond to mouse click...  
    }  
}
```



Generalizziamo

In conclusione per gestire un evento occorre:

1. Definire una classe che implementa una interfaccia Listener
2. Ridefinire tutti i metodi dell'interfaccia
3. Aggiungere il Listener al componente



Listeners e interfacce

Listener interface w/ adapter	Methods in interface
ActionListener	actionPerformed(ActionEvent)
AdjustmentListener	adjustmentValueChanged(AdjustmentEvent)
ComponentListener ComponentAdapter	componentHidden(ComponentEvent) componentShown(ComponentEvent) componentMoved(ComponentEvent) componentResized(ComponentEvent)
ContainerListener ContainerAdapter	componentAdded(ContainerEvent) componentRemoved(ContainerEvent)
FocusListener FocusAdapter	focusGained(FocusEvent) focusLost(FocusEvent)
KeyListener KeyAdapter	keyPressed(KeyEvent) keyReleased(KeyEvent) keyTyped(KeyEvent)



Tipi di eventi

MouseListener MouseAdapter	mouseClicked(MouseEvent) mouseEntered(MouseEvent) mouseExited(MouseEvent) mousePressed(MouseEvent) mouseReleased(MouseEvent)
MouseMotionListener MouseMotionAdapter	mouseDragged(MouseEvent) mouseMoved(MouseEvent)
WindowListener WindowAdapter	windowOpened(WindowEvent) windowClosing(WindowEvent) windowClosed(WindowEvent) windowActivated(WindowEvent) windowDeactivated(WindowEvent) windowIconified(WindowEvent) windowDeiconified(WindowEvent)
ItemListener	itemStateChanged(ItemEvent)



Tipi di eventi

Event, listener interface and add- and remove-methods	Components supporting this event
ActionEvent ActionListener addActionListener() removeActionListener()	JButton, JList, JTextField, JMenuItem and its derivatives including JCheckBoxMenuItem, JMenu, and JpopupMenu.
AdjustmentEvent AdjustmentListener addAdjustmentListener() removeAdjustmentListener()	JScrollbar and anything you create that implements the Adjustable interface.
ComponentEvent ComponentListener addComponentListener() removeComponentListener()	*Component and its derivatives, including JButton, JCheckBox, JComboBox, Container, JPanel, JApplet, JScrollPane, Window, JDialog, JFileDialog, JFrame, JLabel, JList, JScrollbar, JTextArea, and JTextField.



Tipi di eventi

ContainerEvent ContainerListener addContainerListener() removeContainerListener()	Container and its derivatives, including JPanel , JApplet , JScrollPane , Window , JDialog , JFileDialog , and JFrame .
FocusEvent FocusListener addFocusListener() removeFocusListener()	Component and derivatives*.
KeyEvent KeyListener	Component and derivatives*.



Tipi di eventi

Event, listener interface and add- and remove-methods	Components supporting this event
addKeyListener() removeKeyListener()	
MouseEvent (for both clicks and motion) MouseListener addMouseListener() removeMouseListener()	Component and derivatives*.
MouseEvent⁸ (for both clicks and motion) MouseMotionListener addMouseMotionListener() removeMouseMotionListener()	Component and derivatives*.
WindowEvent WindowListener addWindowListener() removeWindowListener()	Window and its derivatives, including JDialog , JFileDialog , and JFrame .
ItemEvent ItemListener addItemListener() removeItemListener()	JCheckBox , JCheckBoxMenuItem , JComboBox , JList , and anything that implements the ItemSelectable interface.
TextEvent TextListener addTextListener() removeTextListener()	Anything derived from JTextComponent , including JTextArea and JTextField .

