

# Appunti di Progettazione del Software

Canale 1 · Prof. Paolo Liberatore

## Capitolo 8: Interfacce grafiche con Swing

**Autore: Tommaso C.**

---

Appunti tratti dalle lezioni in aula e dal materiale del corso

---

Argomenti: Finestre e componenti · Layout · Pannelli · Eventi e ascoltatori ·  
Componenti condivisi · Input non valido

## Indice

|          |                                               |           |
|----------|-----------------------------------------------|-----------|
| <b>1</b> | <b>Finestre e componenti</b>                  | <b>1</b>  |
| <b>2</b> | <b>La disposizione dei componenti</b>         | <b>2</b>  |
| 2.1      | BorderLayout . . . . .                        | 2         |
| 2.2      | GridLayout e FlowLayout . . . . .             | 3         |
| <b>3</b> | <b>Pannelli</b>                               | <b>4</b>  |
| <b>4</b> | <b>La programmazione guidata dagli eventi</b> | <b>5</b>  |
| 4.1      | L'ascoltatore . . . . .                       | 5         |
| 4.2      | Quale pulsante è stato premuto . . . . .      | 6         |
| <b>5</b> | <b>Componenti condivisi con l'ascoltatore</b> | <b>6</b>  |
| <b>6</b> | <b>Esempio: la somma di due numeri</b>        | <b>9</b>  |
| <b>7</b> | <b>Input non valido</b>                       | <b>11</b> |
| <b>8</b> | <b>Esercizi</b>                               | <b>12</b> |
| <b>9</b> | <b>Soluzioni</b>                              | <b>13</b> |

# 1 Finestre e componenti

## Definizione – Finestra e componenti

Una **finestra** è un oggetto della classe `JFrame`. I **componenti** (pulsanti, etichette, campi e aree di testo) sono oggetti che si aggiungono alla finestra con `finestra.add(componente)`.

## Proposizione – Aprire una finestra

1. `new JFrame("titolo")` crea la finestra, con il titolo nella barra.
2. Si creano i componenti e si aggiungono con `add`.
3. `finestra.pack()` dà alla finestra la dimensione che serve ai componenti.
4. `finestra.setVisible(true)` la mostra.

```
import javax.swing.JButton;
import javax.swing.JFrame;

public class Principale {
    public static void main(String[] args) {
        JFrame finestra = new JFrame("Prova");
        JButton pulsante = new JButton("OK");
        finestra.add(pulsante);
        finestra.pack();
        finestra.setVisible(true);
    }
}
```

Una finestra appena creata è vuota e invisibile. In questo modo il programma può preparare all'avvio tutte le sue finestre e mostrare solo la principale, e le altre quando servono. Senza `pack()` la finestra resta troppo piccola e il pulsante, che c'è, non si vede.

I componenti di base sono quattro. Il costruttore riceve il testo da mostrare oppure la dimensione, in caratteri.

| Classe                 | Costruttore                        | Uso                                   |
|------------------------|------------------------------------|---------------------------------------|
| <code>JButton</code>   | <code>new JButton("OK")</code>     | pulsante, si preme                    |
| <code>JLabel</code>    | <code>new JLabel("Nome")</code>    | etichetta, descrive e non si preme    |
| <code>TextField</code> | <code>new JTextField(20)</code>    | una riga di testo, larga 20 caratteri |
| <code>TextArea</code>  | <code>new JTextArea(20, 20)</code> | più righe: 20 righe di 20 caratteri   |

Un campo di testo serve per un singolo valore, come un nome o un numero. L'area di testo serve per un testo su più righe.

## 2 La disposizione dei componenti

### Definizione – Gestore di layout

Il **gestore di layout** di una finestra decide la posizione e la dimensione dei componenti, anche quando la finestra cambia dimensione. Si sceglie con `finestra.setLayout(...)`.

I componenti non hanno una posizione fissa: il layout li dispone in modo relativo, e allarga solo quelli che devono allargarsi. In un editor di testo, per esempio, il menu resta una striscia e l'area in cui si scrive occupa lo spazio in più.

### 2.1 BorderLayout

#### Definizione – BorderLayout

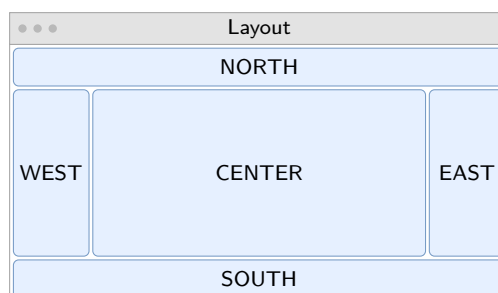
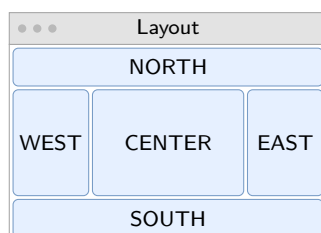
Il **BorderLayout** ha cinque posizioni: **NORTH**, **SOUTH**, **EAST**, **WEST** e **CENTER**. È il layout di una `JFrame`, e la posizione si dà come secondo argomento di `add`:

```
finestra.add(pulsante, BorderLayout.NORTH);
```

Senza posizione il componente va al centro. Ogni posizione contiene un solo componente, e un secondo `add` nella stessa posizione sostituisce il primo. Così, con due `add` senza posizione, si vede solo l'ultimo pulsante.

#### Proposizione – Ridimensionamento nel BorderLayout

- **NORTH** e **SOUTH** hanno l'altezza che serve al componente e si allargano a tutta la larghezza.
- **WEST** e **EAST** hanno la larghezza che serve al componente e si allungano in altezza.
- **CENTER** prende tutto lo spazio che resta.



*Cinque pulsanti nelle cinque posizioni. Allargando la finestra, nord e sud restano alti uguali, ovest ed est restano larghi uguali e il centro prende il resto.*

Di solito a nord va un menu, a sud una barra di stato, al centro il contenuto e ai lati i comandi. Un componente va messo dove la sua forma si adatta: un campo di testo a est diventa alto quanto la finestra, mentre a nord resta una riga.

## 2.2 GridLayout e FlowLayout

### Definizione – GridLayout

`new GridLayout(r, c)` dispone i componenti in una griglia di  $r$  righe e  $c$  colonne, nell'ordine degli `add` e riga per riga. Tutte le celle hanno la stessa dimensione e si ridimensionano insieme.

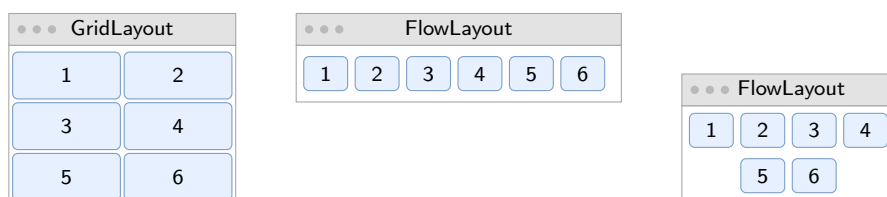
### Proposizione – Righe o colonne a zero

- `new GridLayout(0, c)`:  $c$  colonne e tutte le righe che servono.
- `new GridLayout(r, 0)`:  $r$  righe e tutte le colonne che servono.
- Con  $r$  e  $c$  entrambi diversi da zero conta solo  $r$ , e il numero di colonne si ricava dal numero dei componenti.

Con `GridLayout(0, 1)` i componenti stanno in colonna, uno sotto l'altro, e se ne aggiunge uno non si deve cambiare il numero di righe.

### Definizione – FlowLayout

`new FlowLayout()` dispone i componenti uno dopo l'altro su una riga, centrati e con la loro dimensione. Se la riga non basta, i componenti vanno a capo su più righe.



*Sei pulsanti con `GridLayout(3, 2)`, a sinistra, e con `FlowLayout` in una finestra larga e in una stretta.*

| Layout                    | Posizione in <code>add</code> | Componenti             | Spazio in più         |
|---------------------------|-------------------------------|------------------------|-----------------------|
| <code>BorderLayout</code> | una delle cinque              | secondo la posizione   | soprattutto al centro |
| <code>GridLayout</code>   | nessuna, in ordine            | tutti uguali           | a tutte le celle      |
| <code>FlowLayout</code>   | nessuna, in ordine            | con la loro dimensione | resta vuoto           |

**Nota – Provare un layout con i pulsanti**

Per capire un layout conviene riempirlo di pulsanti e ridimensionare la finestra: i pulsanti si vedono bene e mostrano quanto spazio riceve ogni posizione.

### 3 Pannelli

**Definizione – Pannello**

Un **pannello** è un oggetto **JPanel**: un contenitore di componenti, con un suo layout, che si aggiunge a una finestra o a un altro pannello come un singolo componente.

Con i pannelli una posizione del **BorderLayout** contiene più componenti. Per esempio il centro della finestra diventa una griglia di quattro pulsanti:

```
JPanel pannello = new JPanel();
pannello.setLayout(new GridLayout(2, 2));
pannello.add(new JButton("1"));
pannello.add(new JButton("2"));
pannello.add(new JButton("3"));
pannello.add(new JButton("4"));
finestra.add(pannello, BorderLayout.CENTER);
```

Restano libere le altre posizioni della finestra. Dentro un pannello si può mettere un altro **BorderLayout**, una griglia o un **FlowLayout**, oppure un altro pannello (Esercizio 1).

**Osservazione – Errori tipici**

- Il pannello va aggiunto alla finestra con `finestra.add(pannello)`: se no, lui e i suoi componenti non si vedono.
- I componenti del pannello si aggiungono con `pannello.add(...)`, non con `finestra.add(...)`.
- Un pulsante creato dentro `add`, come `add(new JButton("1"))`, non ha una variabile, e quindi non gli si può associare un comportamento. Un pulsante che deve fare qualcosa va prima messo in una variabile.

## 4 La programmazione guidata dagli eventi

### Definizione – Programmazione guidata dagli eventi

Nella **programmazione guidata dagli eventi** il **main** costruisce l'interfaccia e termina. Il programma resta in esecuzione e aspetta gli **eventi**, cioè le azioni dell'utente, e a ogni evento esegue il codice che gli è stato associato.

Un `System.out.println("main terminato")` come ultima istruzione del **main** lo mostra: la scritta compare nella console subito, con la finestra già aperta, e il programma continua. Per farlo terminare quando si chiude la finestra serve

```
finestra.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

altrimenti la X chiude la finestra ma il programma resta in esecuzione.

### 4.1 L'ascoltatore

Un metodo riceve come argomenti solo valori primitivi e riferimenti a oggetti, non istruzioni. Il comportamento di un pulsante si passa quindi come un oggetto, che serve solo per il suo metodo: il codice da eseguire alla pressione.

### Definizione – Ascoltatore

Un **ascoltatore** è un oggetto di una classe che implementa l'interfaccia `ActionListener`, cioè dichiara `implements ActionListener` e definisce il metodo

```
public void actionPerformed(ActionEvent e)
```

Con `pulsante.addActionListener(a)`, alla pressione del pulsante viene eseguito `a.actionPerformed(e)`.

```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

public class Ascoltatore implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        System.out.println("metodo eseguito");
    }
}
```

```
// nel main, dopo aver creato il pulsante
Ascoltatore a = new Ascoltatore();
pulsante.addActionListener(a);
```

Il metodo `actionPerformed` lo chiama Swing, alla pressione del pulsante, e non il programma. Un `ActionListener` è come una classe astratta con il solo metodo astratto `actionPerformed` (Capitolo 5): se la classe non lo definisce, con quell'intestazione e `public`, il programma non compila.

Il `main` del programma resta corto, perché costruisce solo la finestra. Quello che il programma fa sta negli ascoltatori.

#### Osservazione – Errore tipico: l'ascoltatore non associato

Se si dimentica `addActionListener`, il pulsante non fa nulla e non compare nessun errore. Per accorgersene, la prima istruzione di `actionPerformed` è una stampa: se premendo il pulsante non compare, l'ascoltatore non è associato.

## 4.2 Quale pulsante è stato premuto

Lo stesso ascoltatore si può associare a più pulsanti. L'evento `e` dice quale è stato premuto: `e.getActionCommand()` restituisce il testo del pulsante. È una stringa, e si confronta con `equals` (Capitolo 6).

```
public void actionPerformed(ActionEvent e) {  
    if (e.getActionCommand().equals("OK")) {  
        System.out.println("premuto OK");  
    } else {  
        System.out.println("premuto un altro pulsante");  
    }  
}
```

In alternativa, ogni pulsante ha il suo ascoltatore, di una classe diversa (Esercizio 2).

## 5 Componenti condivisi con l'ascoltatore

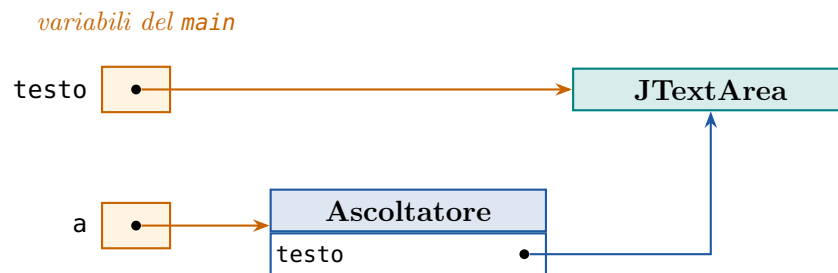
#### Proposizione – Componenti nell'ascoltatore

Un ascoltatore che modifica un componente ha un campo dello stesso tipo. Il `main` vi copia il riferimento al componente che ha messo nella finestra:

```
a.testo = testo;
```

Così il campo dell'ascoltatore e la variabile del `main` si riferiscono allo stesso oggetto.

Il componente non si può passare come argomento di `actionPerformed`, perché Swing chiama il metodo con il solo evento. La variabile `testo` del `main` è locale e l'ascoltatore non la vede. Il punto in comune è l'oggetto ascoltatore, che il `main` crea.



Dopo `a.testo = testo` la variabile del `main` e il campo dell'ascoltatore si riferiscono alla stessa area di testo, quella nella finestra.

Per modificare un'area di testo:

- `testo.append("linea\n")` aggiunge in fondo. Come `print`, non va a capo da solo, per cui il ritorno a capo va nel testo.
- `testo.setText("")` sostituisce tutto il testo, e con la stringa vuota cancella l'area.

### Esempio – Aggiungere e cancellare righe

Una finestra con un'area di testo al centro, un pulsante *linea* che vi aggiunge una riga e un pulsante *cancella* che la svuota.

```

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JTextArea;

public class Ascoltatore implements ActionListener {
    JTextArea testo; // la stessa area mostrata nella finestra

    public void actionPerformed(ActionEvent e) {
        System.out.println("premuto " + e.getActionCommand());
        if (e.getActionCommand().equals("linea")) {
            testo.append("linea\n");
        } else {
            testo.setText("");
        }
    }
}
  
```

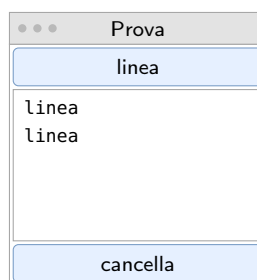
```
import java.awt.BorderLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JTextArea;

public class Principale {
    public static void main(String[] args) {
        JFrame finestra = new JFrame("Prova");
        JButton linea = new JButton("linea");
        JButton cancella = new JButton("cancella");
        JTextArea testo = new JTextArea(20, 20);

        finestra.add(linea, BorderLayout.NORTH);
        finestra.add(cancella, BorderLayout.SOUTH);
        finestra.add(testo, BorderLayout.CENTER);

        Ascoltatore a = new Ascoltatore();
        a.testo = testo;
        linea.addActionListener(a);
        cancella.addActionListener(a);

        finestra.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        finestra.pack();
        finestra.setVisible(true);
    }
}
```



*La finestra dopo due pressioni di linea.*

### Osservazione – Errore tipico: due aree diverse

Il campo dell'ascoltatore deve riferirsi all'area che sta nella finestra.

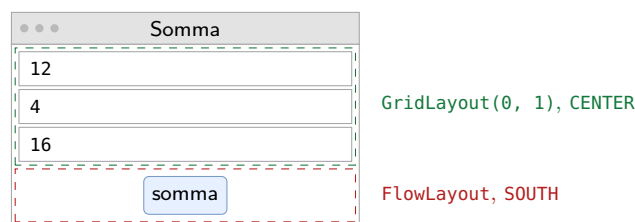
- Senza `a.testo = testo` il campo vale `null`, e `append` genera un'eccezione.
- Con un `new JTextArea(...)` nell'ascoltatore il programma modifica un'area che nessuno vede.

## 6 Esempio: la somma di due numeri

### Esempio – La somma

Una finestra con tre campi di testo e un pulsante *somma*. Alla pressione, il terzo campo mostra la somma dei numeri interi scritti nei primi due.

I campi stanno in un pannello con `GridLayout(0, 1)` al centro, e il pulsante in un pannello con `FlowLayout` a sud, dove mantiene la sua dimensione.



```
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JTextField;

public class Somma implements ActionListener {
    JTextField primo;
    JTextField secondo;
    JTextField risultato;

    public void actionPerformed(ActionEvent e) {
        System.out.println("premutato " + e.getActionCommand());
        int p = Integer.parseInt(primo.getText());
        int s = Integer.parseInt(secondo.getText());
        risultato.setText(Integer.toString(p + s));
    }
}
```

`getText()` restituisce una stringa, e `Integer.parseInt` la converte in `int`. Con le stringhe il `+` concatena: `"12" + "4"` vale `"124"`. Il risultato si riconverte in stringa per `setText`.

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class Principale {
    public static void main(String[] args) {
        JFrame finestra = new JFrame("Somma");

        JTextField primo = new JTextField(20);
        JTextField secondo = new JTextField(20);
        JTextField risultato = new JTextField(20);
        JPanel campi = new JPanel();
        campi.setLayout(new GridLayout(0, 1));
        campi.add(primo);
        campi.add(secondo);
        campi.add(risultato);
        finestra.add(campi, BorderLayout.CENTER);

        JButton somma = new JButton("somma");
        JPanel pulsanti = new JPanel();
        pulsanti.setLayout(new FlowLayout());
        pulsanti.add(somma);
        finestra.add(pulsanti, BorderLayout.SOUTH);

        Somma a = new Somma();
        a.primo = primo;
        a.secondo = secondo;
        a.risultato = risultato;
        somma.addActionListener(a);

        finestra.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        finestra.pack();
        finestra.setVisible(true);
    }
}
```

## 7 Input non valido

### Proposizione – Input non valido

Se un campo non contiene un intero, per esempio 12a, `Integer.parseInt` genera un'eccezione (`NumberFormatException`). L'esecuzione di `actionPerformed` si interrompe e il messaggio va solo nella console.

A differenza delle eccezioni del Capitolo 4, il programma non termina: la finestra resta aperta e alla pressione successiva, con dati corretti, la somma funziona. Nell'interfaccia però non cambia nulla, e il terzo campo mostra ancora il risultato precedente.

### Osservazione – L'errore va mostrato nell'interfaccia

Fuori dall'ambiente di sviluppo la console di solito non si vede, quindi l'utente non riceve nessun segnale. Un inserimento sbagliato va segnalato nell'interfaccia, dicendo quale campo non va bene. Per farlo il programma deve intercettare l'eccezione invece di lasciarla arrivare alla console.

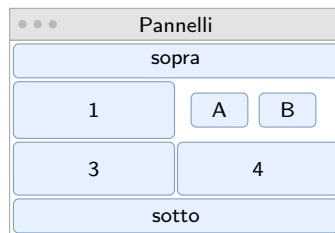
### Riepilogo

- **Finestra:** `new JFrame("titolo")`, componenti con `add`, poi `pack()` e `setVisible(true)`.
- **Componenti:** `JButton` si preme, `JLabel` descrive, `TextField` è una riga, `TextArea` più righe.
- **BorderLayout:** cinque posizioni, una per componente. Nord e sud si allargano, ovest ed est si allungano, il centro prende il resto.
- **Grid e Flow:** `GridLayout(r, c)` fa celle uguali, con uno dei due a zero; `FlowLayout` mette in riga e va a capo.
- **Pannello:** `JPanel` con un suo layout, aggiunto alla finestra; i componenti si aggiungono al pannello.
- **Eventi:** il `main` costruisce e termina; `EXIT_ON_CLOSE` chiude il programma con la finestra.
- **Ascoltatore:** classe che implementa `ActionListener` e definisce `actionPerformed`, associata con `addActionListener`. Il pulsante premuto si legge con `getActionCommand()`.
- **Condivisione e input:** campi dell'ascoltatore assegnati nel `main`; `parseInt` su testo non numerico genera un'eccezione che l'utente non vede.

## 8 Esercizi

### Esercizio 1 – Pannelli annidati

Scrivere il programma che apre la finestra in figura: un pulsante a nord, uno a sud e al centro una griglia  $2 \times 2$ . La seconda cella della griglia contiene i due pulsanti *A* e *B*, affiancati.



*Soluzione a pagina 14 →*

### Esercizio 2 – Un ascoltatore per pulsante

Riscrivere l'esempio della Sezione 5 (pulsanti *linea* e *cancella*) con due ascoltatori di due classi diverse, uno per pulsante, senza usare `getActionCommand`.

*Soluzione a pagina 15 →*

## 9 Soluzioni

## Soluzione dell'Esercizio 1 – Pannelli annidati

La finestra usa il **BorderLayout**. Al centro va un pannello con **GridLayout(2, 2)**, e nella sua seconda cella un altro pannello, con **FlowLayout**, che contiene *A* e *B*. I pannelli si riempiono in ordine, e ognuno si aggiunge al suo contenitore.

```
import java.awt.BorderLayout;
import java.awt.FlowLayout;
import java.awt.GridLayout;
import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JPanel;

public class Principale {
    public static void main(String[] args) {
        JFrame finestra = new JFrame("Pannelli");

        JPanel interno = new JPanel();
        interno.setLayout(new FlowLayout());
        interno.add(new JButton("A"));
        interno.add(new JButton("B"));

        JPanel griglia = new JPanel();
        griglia.setLayout(new GridLayout(2, 2));
        griglia.add(new JButton("1"));
        griglia.add(interno);
        griglia.add(new JButton("3"));
        griglia.add(new JButton("4"));

        finestra.add(new JButton("sopra"), BorderLayout.NORTH);
        finestra.add(griglia, BorderLayout.CENTER);
        finestra.add(new JButton("sotto"), BorderLayout.SOUTH);

        finestra.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        finestra.pack();
        finestra.setVisible(true);
    }
}
```

Con il **FlowLayout**, *A* e *B* mantengono la loro dimensione e restano centrati nella cella, mentre *1*, *3* e *4* riempiono la loro. I pulsanti non hanno una variabile perché qui conta solo la disposizione.

[← Torna all'Esercizio 1, pagina 12](#)

## Soluzione dell'Esercizio 2 – Un ascoltatore per pulsante

Ogni classe fa una sola cosa, e ognuna ha il suo campo per l'area di testo:

```
public class Linea implements ActionListener {
    JTextArea testo;

    public void actionPerformed(ActionEvent e) {
        testo.append("linea\n");
    }
}
```

```
public class Cancella implements ActionListener {
    JTextArea testo;

    public void actionPerformed(ActionEvent e) {
        testo.setText("");
    }
}
```

Le due classi importano `ActionEvent`, `ActionListener` e `JTextArea` come `Ascoltatore`. Nel `main` entrambi i campi ricevono la stessa area:

```
Linea l = new Linea();
l.testo = testo;
Cancella c = new Cancella();
c.testo = testo;
linea.addActionListener(l);
cancella.addActionListener(c);
```

Il pulsante premuto non va riconosciuto, perché ogni ascoltatore è associato a un solo pulsante.

[← Torna all'Esercizio 2, pagina 12](#)