UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI **MATEMATICA E INFORMATICA**

Interfacce Grafiche e Programmazione ad Eventi

Carmine Dodaro

Anno Accademico 2019/2020

Progettazione di interfacce utente

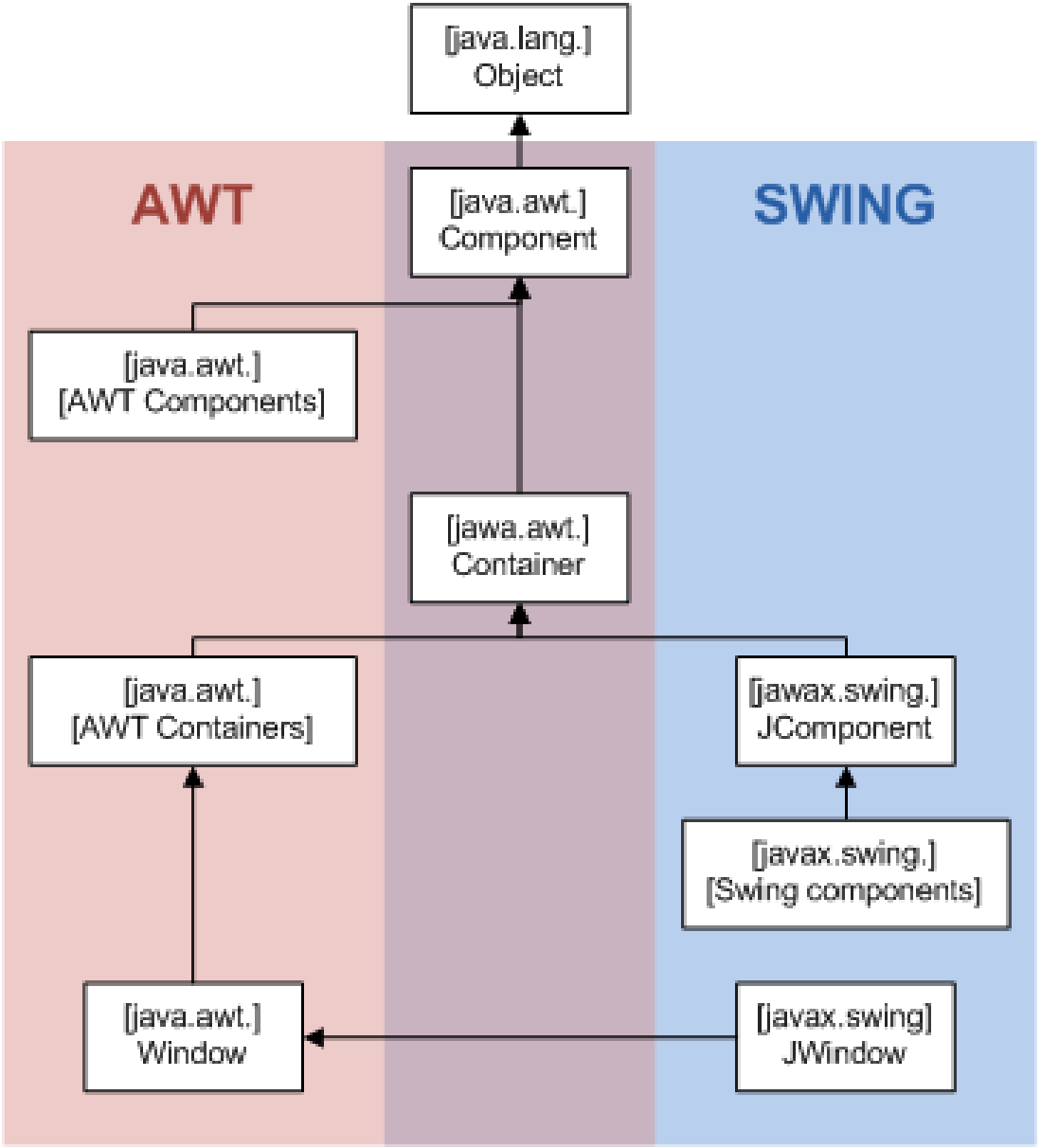
Interfacce utente

- Librerie Swing, che fanno parte di un insieme di librerie chiamate Java Foundation Classes (JFC)
- Sono state introdotte come evoluzione delle librerie Abstract Window Toolkit (AWT)
- La differenza tra Swing e AWT è nel modo in cui i componenti sono stati progettati e come si comportano in un ambiente nativo di esecuzione

  Componenti AWT: hanno un aspetto legato alla piattaforma su cui girano. Sono chiamati controlli pesanti (heavyweight), perché sono totalmente dipendenti dal sistema grafico su cui vengono eseguiti. Allo stesso tempo sono componenti veloci

  Componenti Swing: indipendenti dal sistema grafico e presentano un aspetto diverso tra le diverse piattaforme. Sono chiamati controlli leggeri (lightweight), visto che sono portabili in termini di aspetto in vari sistemi grafici. Allo stesso tempo sono meno veloci
- Le Swing non sostituiscono le AWT ma si considerano complementari, anche se hanno funzionamenti diversi!

Una visione d'insieme



Fonte: wikipedia.it

Container

Cosa sono

I container rappresentano delle aree all'interno delle quali si inseriscono le componenti dell'interfaccia. Esistono due tipologie di container:

- container top-level: sono rappresentati dalle classi JFrame, JApplet e JDialog. Per costruire un'interfaccia utente deve essere presente almeno un container top-level
- altri container: sono rappresentati da varie classi, tra cui JPanel, JTabbedPane, ecc. La loro presenza non è obbligatoria

Per apparire sullo schermo, ogni componente grafico deve essere parte di una qualche gerarchia di componenti. Una gerarchia di componenti è un albero che ha un container top-level come radice. Ogni componente grafico può essere contenuto solo una volta: se un componente è già contenuto in un container e si prova ad aggiungerlo in un altro container, l'effetto sarà quello di rimuoverlo dal primo e aggiungerlo al secondo.

Ogni container top-level ha un content pane, cioè un riquadro che contiene tutti gli elementi visibili in questo container. Come vedremo, questo sarà la componente principale su cui lavoreremo

Top-level container

Composizione dei top-level container

I top-level container hanno quattro strati (detti layer oppure pane):

1. root pane: è lo strato principale e si occupa di gestire gli altri strati
2. layered pane: è lo strato che racchiude il content pane e la barra dei menu (nel caso in cui fosse specificata) e serve per decidere la posizione di questi due elementi
3. content pane: contiene tutte le componenti visibili del root pane
4. glass pane: è uno strato superiore a tutti gli altri ed è nascosto di default. Nel caso in cui fosse reso visibile è come un blocco di vetro su tutti gli altri componenti del root pane. Ad esempio, può essere utile per disegnare delle immmagini su più componenti

Il container JFrame

La classe JFrame

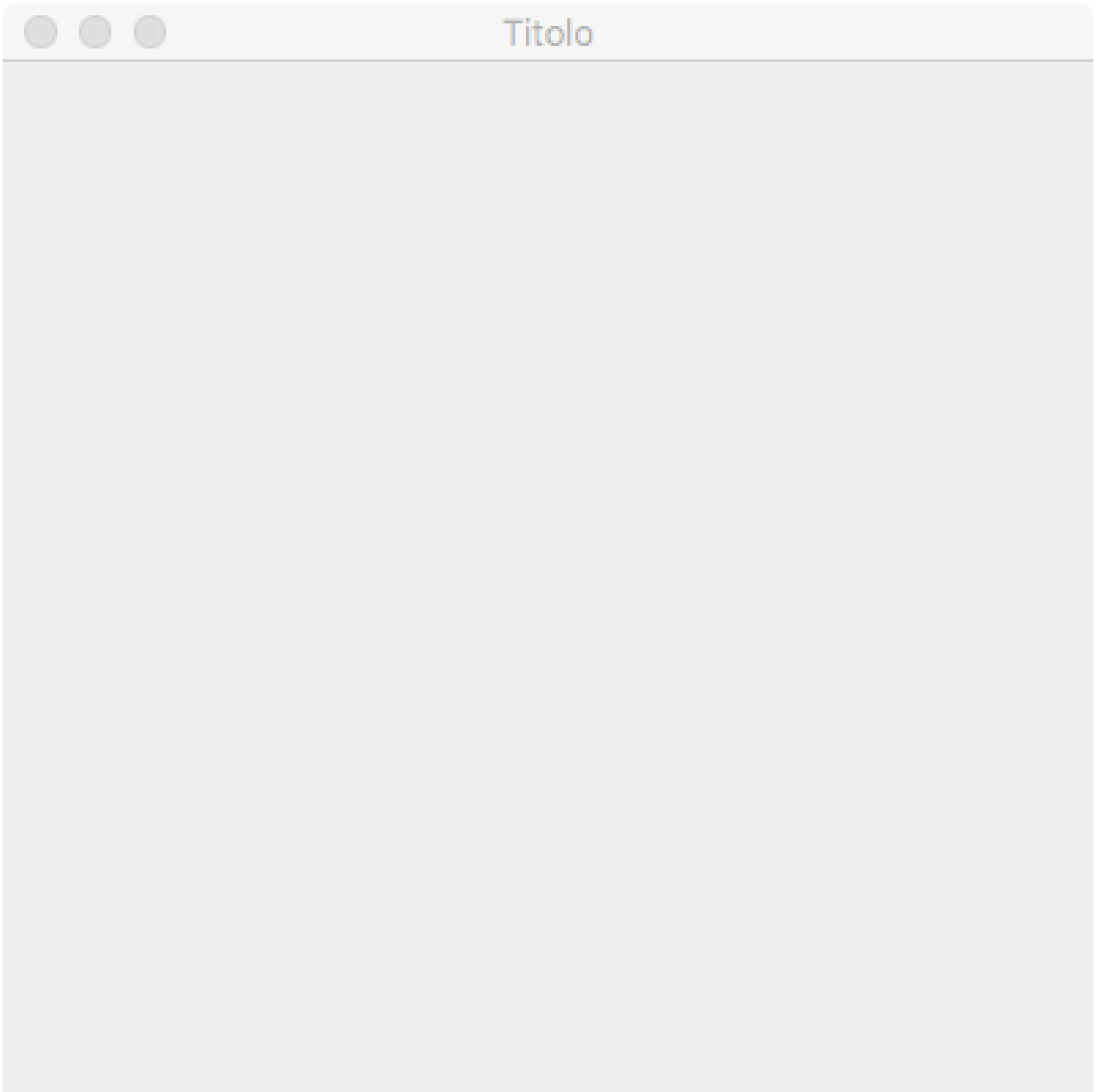
Un oggetto di tipo JFrame permette di avere una finestra con un titolo, un'icona e i vari pulsanti di riduzione a icona, di chiusura, ecc.

```
public static void main(String[] args) {
  JFrame f = new JFrame("Titolo");
  //Dimensioni della finestra (larghezza x altezza)
  f.setSize(400, 400);

  //Per creare una finestra che copra lo schermo intero
  //f.setExtendedState(JFrame.MAXIMIZED_BOTH);
  f.setVisible(true);

  //Quando chiudiamo la finestra il programma termina
  f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
```

JFrame: esempio



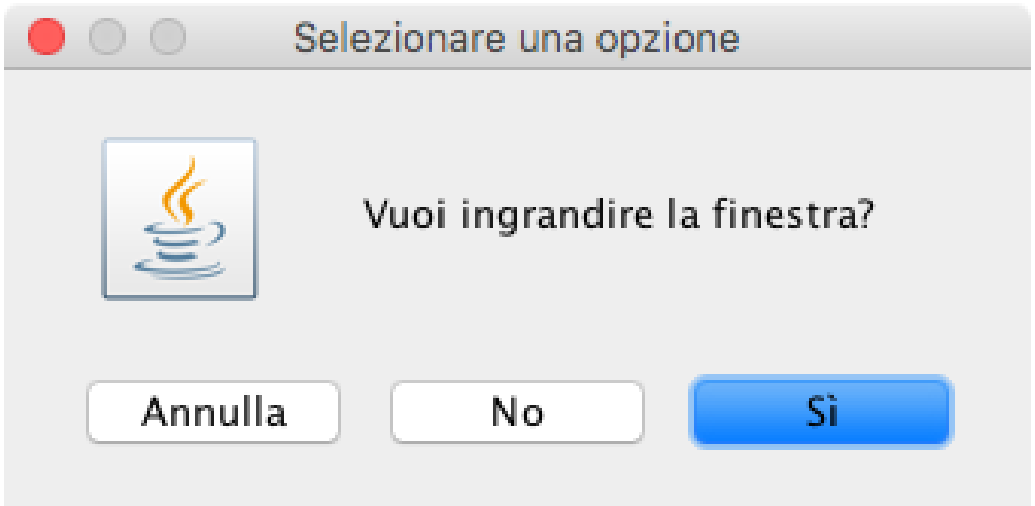
Il container JDialog

La classe JOptionPane offre la possibilità di creare delle finestre di dialogo, utili nel caso in cui si vogliano mostrare oppure ottenere delle informazioni. Agiscono come finestre temporanee che possono essere visualizzate a schermo.

```
public static void main(String[] args) {
  JFrame f = new JFrame("Titolo");
  f.setSize(400, 400);
  f.setVisible(true);
  f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

  int scelta = JOptionPane.showConfirmDialog(f, "Vuoi
      ingrandire la finestra?");
  if(scelta == JOptionPane.YES_OPTION)
    f.setExtendedState(JFrame.MAXIMIZED_BOTH);
}
```

JDialog: esempio



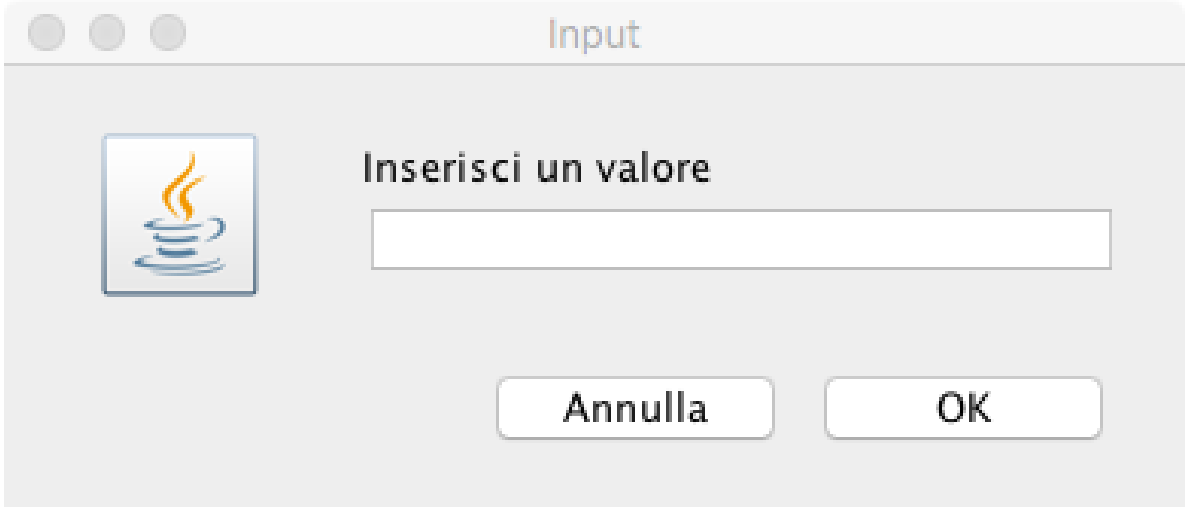
Altre opzioni

Metodo showInputDialog

Permette all'utente di inserire un valore da input

```
public static void main(String[] args) {
  String a = JOptionPane.showInputDialog("Inserisci un
      valore");
  int res = JOptionPane.showConfirmDialog(null, "Hai
      inserito " + a + "?", "Titolo", JOptionPane.
      YES_NO_OPTION,
  JOptionPane.QUESTION_MESSAGE);
}
```

JDialog: esempio



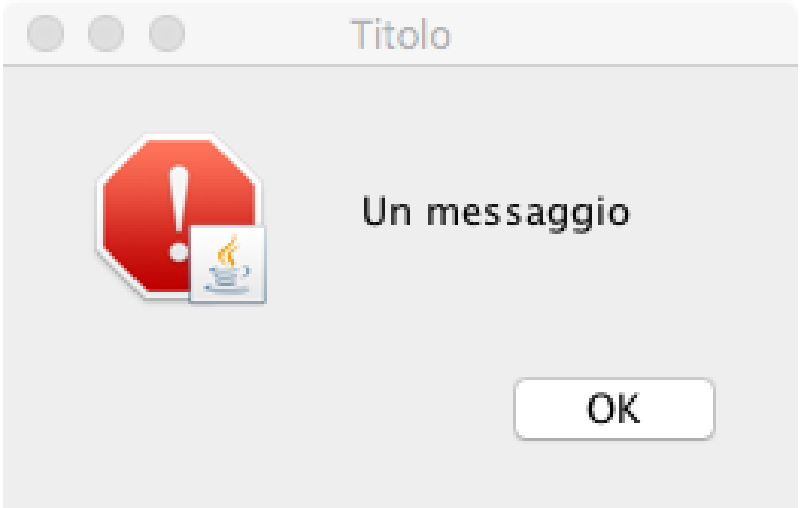
Altre opzioni

Metodo showMessage

Permette di mostrare un messaggio all'utente

```
JOptionPane.showMessageDialog(null, "Testo messaggio", "
    Titolo", JOptionPane.ERROR_MESSAGE);
  }
```

JDialog: esempio



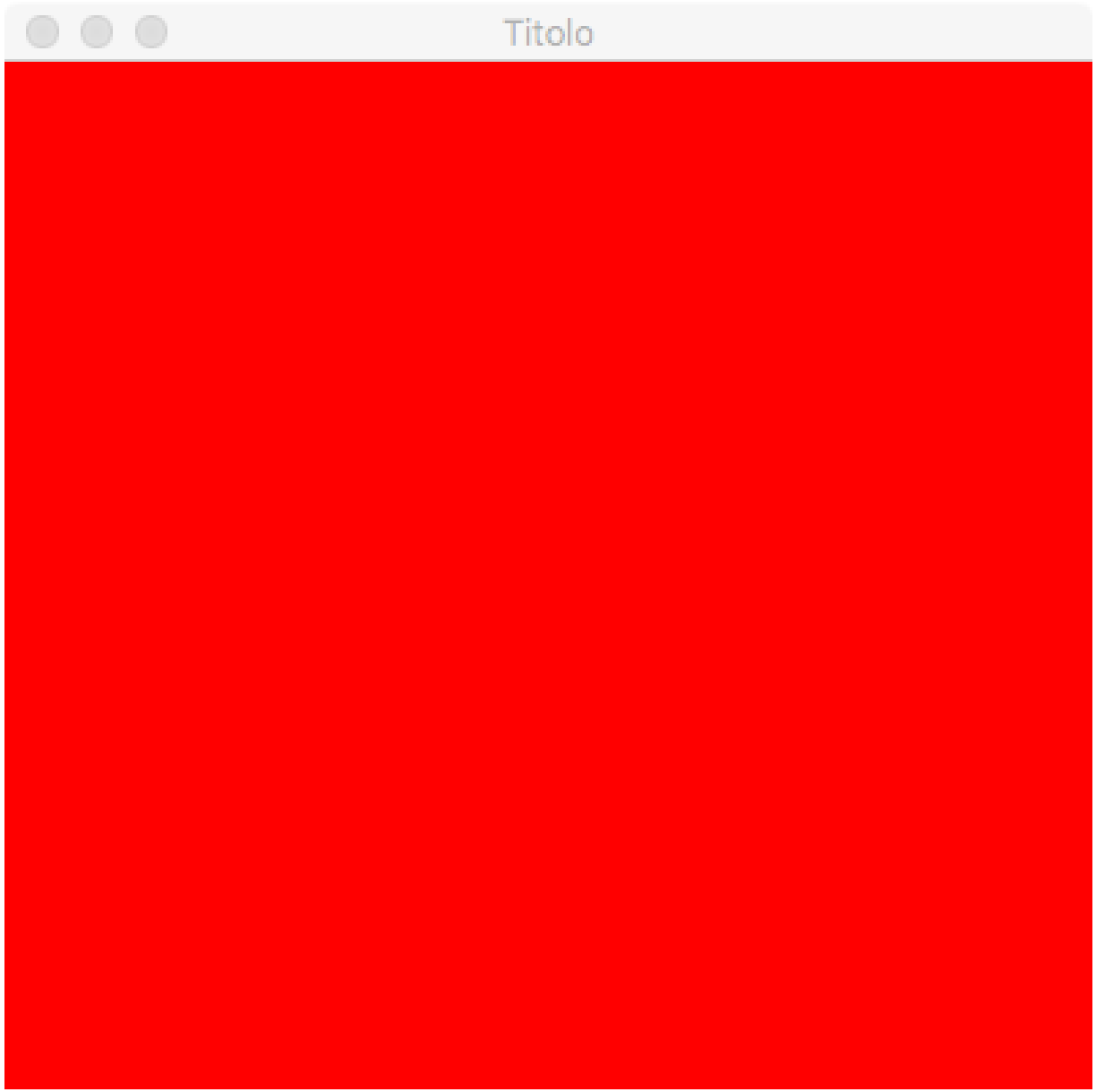
Il container JPanel

Descrizione

Il container di tipo JPanel rappresenta un pannello a cui si possono aggiungere altri componenti grafici. Tipicamente è usato per dividere la finestra in diverse aree, ognuna delle quali contiene una componente specializzata

```
JFrame f = new JFrame("Titolo");
JPanel mainPanel = new JPanel();
mainPanel.setBackground(Color.RED);
f.add(mainPanel);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JPanel: esempio



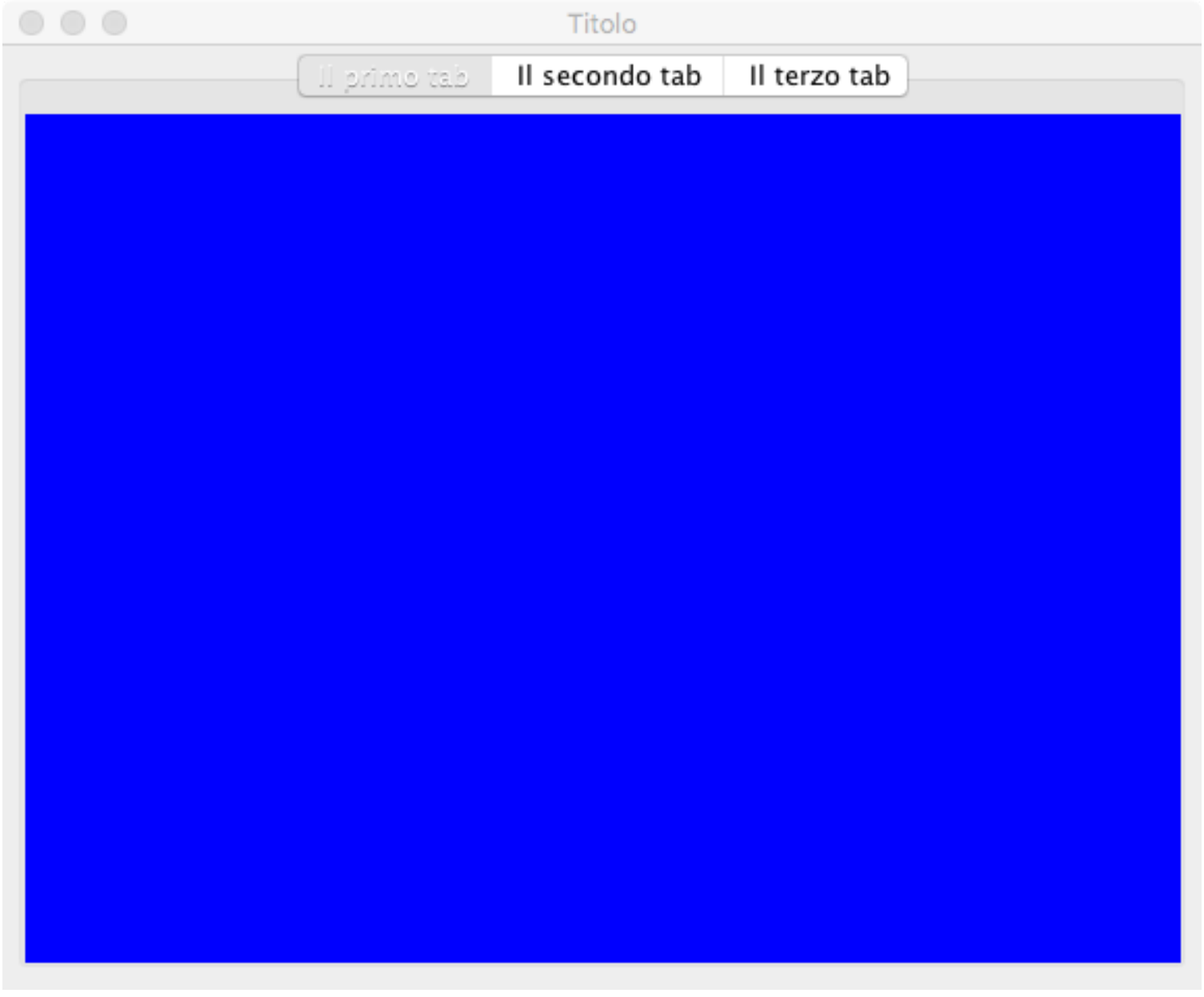
Il container JTabbedPane

Descrizione

Suddivide l'area in tante schede (o tab)

```
JFrame f = new JFrame("Titolo");
JPanel tab1 = new JPanel();
tab1.setBackground(Color.BLUE);
JPanel tab2 = new JPanel();
tab2.setBackground(Color.RED);
JPanel tab3 = new JPanel();
tab3.setBackground(Color.YELLOW);
JTabbedPane tabbedPane = new JTabbedPane();
tabbedPane.addTab("Il primo tab", tab1);
tabbedPane.addTab("Il secondo tab", tab2);
tabbedPane.addTab("Il terzo tab", tab3);
f.add(tabbedPane);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JTabbedPane: esempio



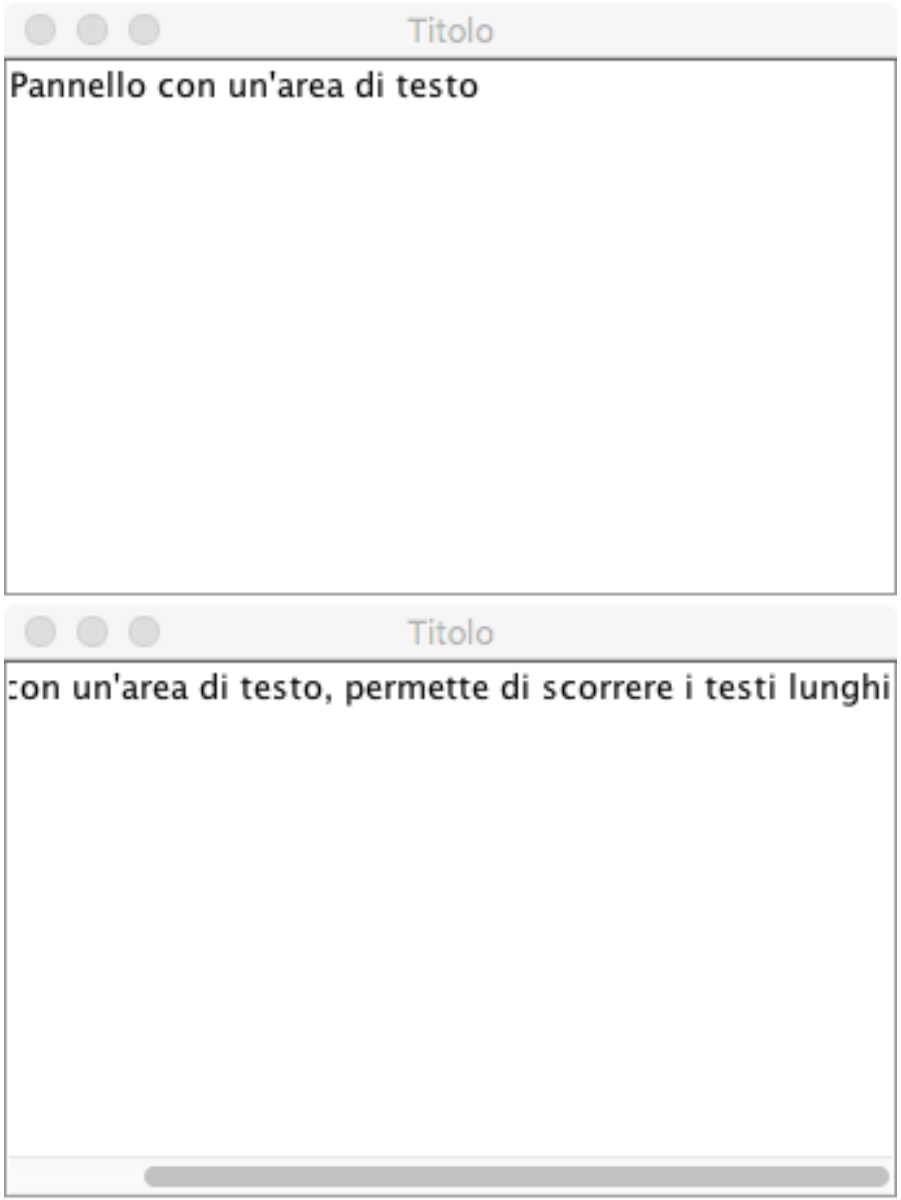
JScrollPane

Descrizione

JScrollPane offre un'area di scorrimento ad altri componenti

```
JFrame f = new JFrame("Titolo");
JTextArea textArea = new JTextArea();
JScrollPane scrollPane = new JScrollPane(textArea);
f.add(scrollPane);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JScrollPane: esempio



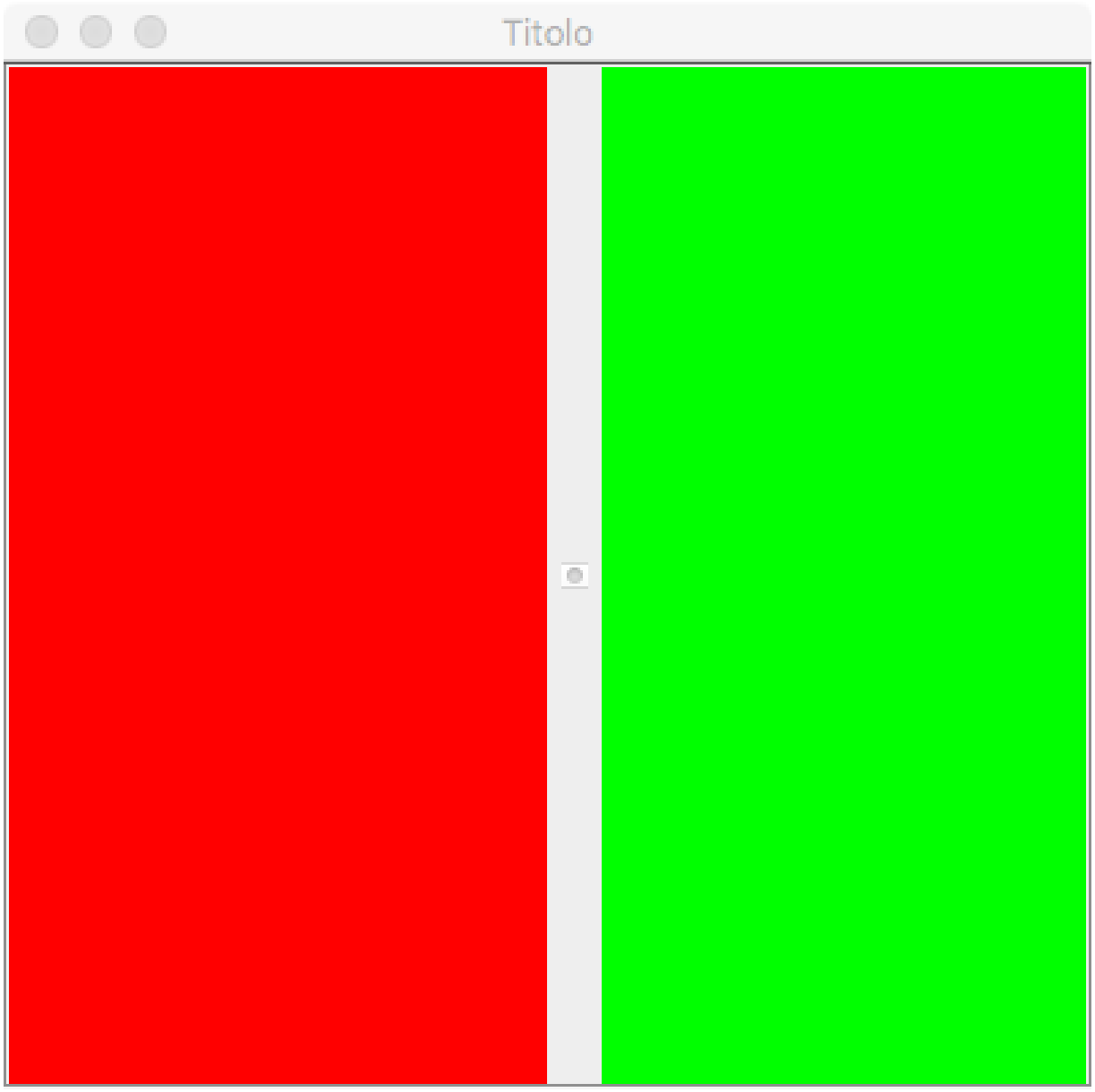
JSplitPane

Descrizione

JSplitPane permette di dividere l'area in due sottoaree separate da un bordo, la cui dimensione può essere anche modificata

```
JFrame f = new JFrame("Titolo");
JPanel p1 = new JPanel();
p1.setBackground(Color.RED);
JPanel p2 = new JPanel();
p2.setBackground(Color.GREEN);
JSplitPane splitPane = new JSplitPane(JSplitPane.
    HORIZONTAL_SPLIT,p1,p2);
splitPane.setDividerLocation(200);
splitPane.setDividerSize(20);
f.add(splitPane);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

JSplitPane: esempio



I layout manager

Cosa sono

I layout manager sono oggetti che si occupano di posizionare i vari componenti di un container seguendo delle indicazioni precise. Il vantaggio principale di usare un layout manager è che i componenti al suo interno rispetteranno il layout imposto anche in caso di ridimensionamento.

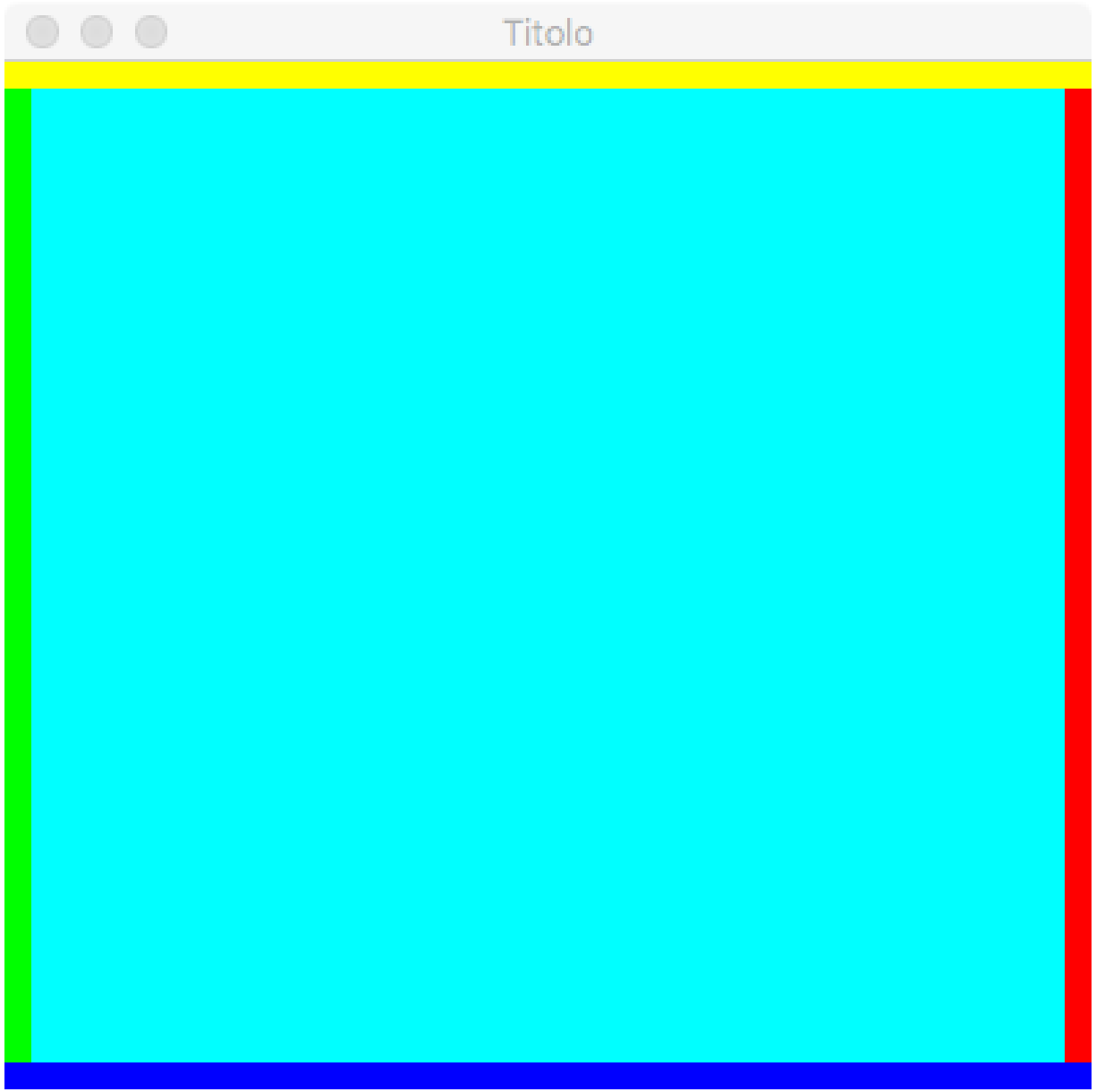
BorderLayout

Descrizione

Il BorderLayout suddivide il container in cinque aree: nord, sud, est, ovest e centro. In ogni area si può aggiungere un determinato componente grafico. È il layout di default di JFrame e JDialog.

```
JFrame f = new JFrame("Titolo");
JPanel sopra = new JPanel();
sopra.setBackground(Color.YELLOW);
JPanel sinistra = new JPanel();
sinistra.setBackground(Color.GREEN);
JPanel centro = new JPanel();
centro.setBackground(Color.CYAN);
JPanel destra = new JPanel();
destra.setBackground(Color.RED);
JPanel sotto = new JPanel();
sotto.setBackground(Color.BLUE);
f.add(sopra, BorderLayout.PAGE_START);
f.add(sinistra, BorderLayout.LINE_START);
f.add(centro, BorderLayout.CENTER);
f.add(destra, BorderLayout.LINE_END);
f.add(sotto, BorderLayout.PAGE_END);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

BorderLayout: esempio



BorderLayout: dimensionamento dei componenti

Dimensionamento

I componenti del BorderLayout sono dimensionati seguendo queste regole:

1. Il componente superiore e inferiore occupa tutta l'area in orizzontale
2. I componenti laterali occupano tutto lo spazio verticale (ad esclusione di quello occupato dai componenti orizzontali)
3. l'area centrale occupa lo spazio rimanente
4. ogni area può contenere un solo componente, nel caso di inserimenti multipli, solo l'ultimo sarà visualizzato

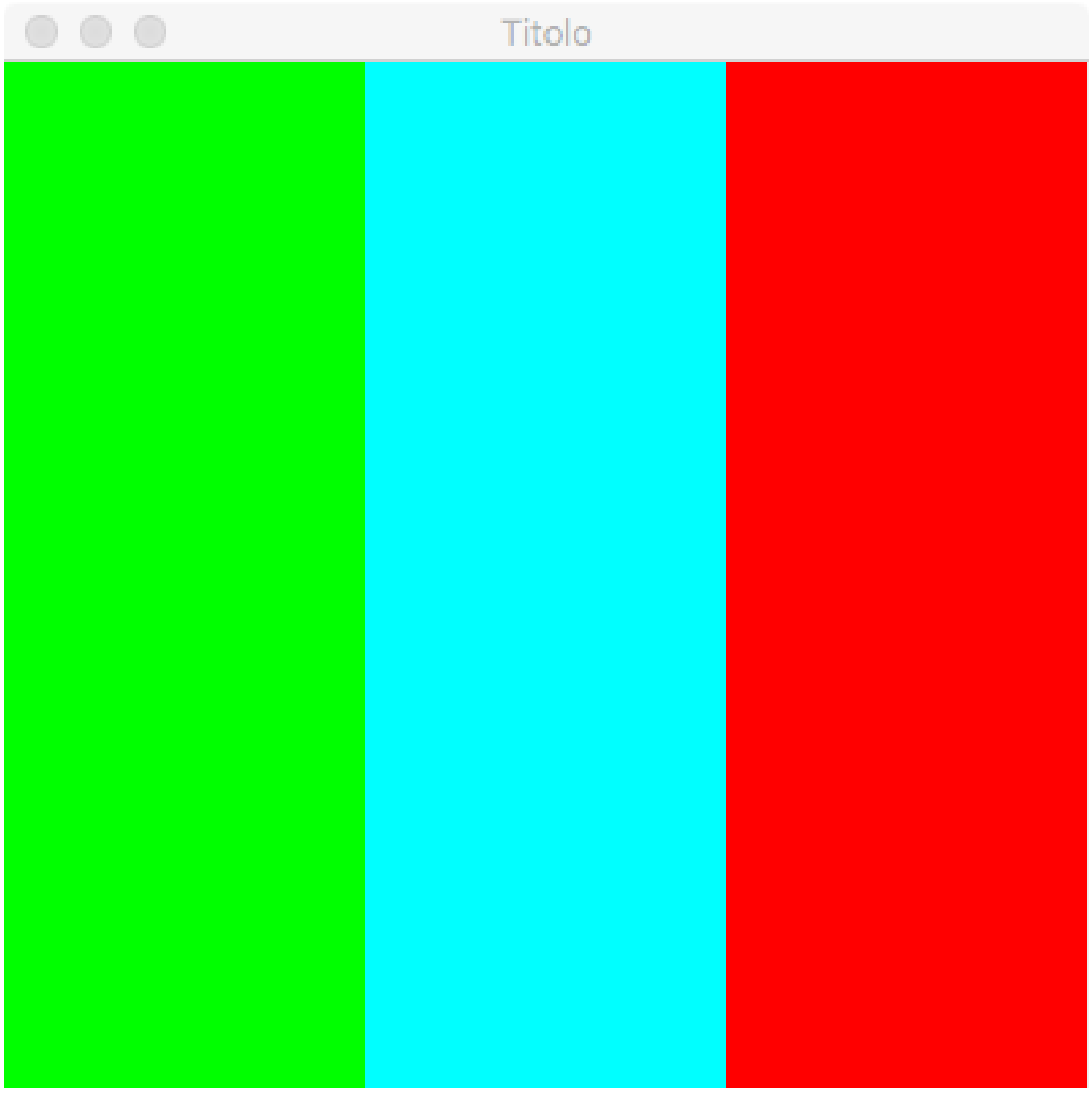
BoxLayout

Descrizione

Il BoxLayout consente di posizionare i componenti grafici in orizzontale oppure in verticale

```
JFrame f = new JFrame("Titolo");
// verticale
BoxLayout b = new BoxLayout(f.getContentPane(), BoxLayout.
    LINE_AXIS);
// orizzontale
//  BoxLayout b = new BoxLayout(f.getContentPane(), BoxLayout.
    PAGE_AXIS);
f.setLayout(b);
JPanel sinistra = new JPanel();
sinistra.setBackground(Color.GREEN);
JPanel centro = new JPanel();
centro.setBackground(Color.CYAN);
JPanel destra = new JPanel();
destra.setBackground(Color.RED);
f.add(sinistra);
f.add(centro);
f.add(destra);
f.setSize(400, 400);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

BoxLayout: esempio



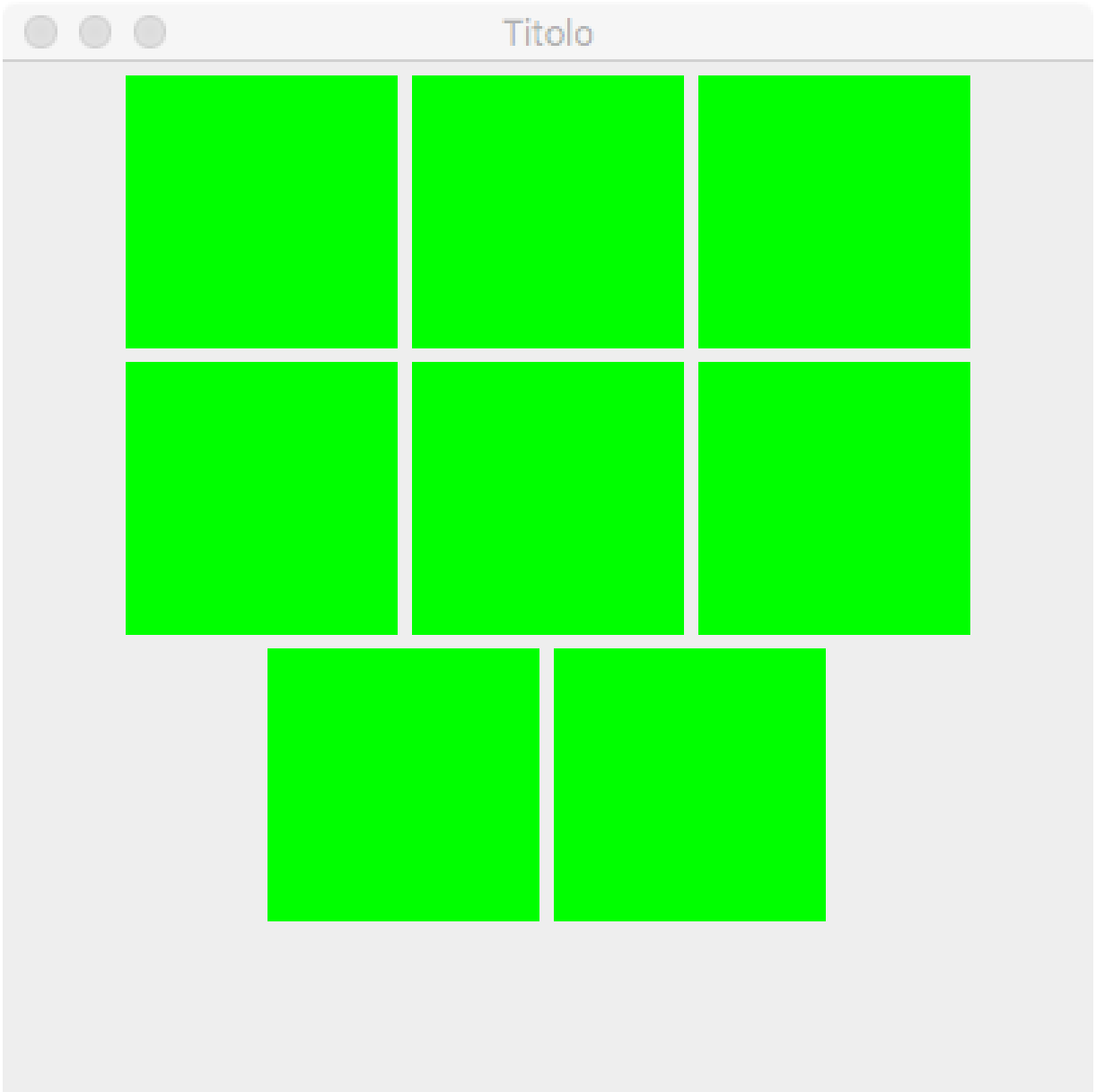
FlowLayout

Descrizione

Il FlowLayout inserisce gli elementi in una linea continua da sinistra a destra.
È il layout di default per JPanel

```
JFrame f = new JFrame("Titolo");
FlowLayout flow = new FlowLayout();
f.setLayout(flow);
for(int i = 0; i < 8; i++) {
JPanel pannello = new JPanel();
   pannello.setBackground(Color.GREEN);
   pannello.setPreferredSize(new Dimension(100, 100));
   f.add(pannello);
}
f.setSize(400, 400);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setVisible(true);
```

FlowLayout: esempio



GridLayout

Descrizione

Il GridLayout consente di disporre gli elementi in righe e colonne, quindi in forma tabellare

```
JFrame f = new JFrame("Titolo");
//2 righe e 4 colonne, 5 pixel di spazio tra ogni cella
GridLayout grid = new GridLayout(2,4,5,5);
f.setLayout(grid);
for(int i = 0; i < 8; i++) {
   JPanel pannello = new JPanel();
   pannello.setBackground(Color.GREEN);
   f.add(pannello);
}
f.setSize(400, 400);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
f.setVisible(true);
```

GridLayout: esempio