

# AWT e Swing

Java

- Abstract Window Toolkit (**AWT**) è la libreria Java contenente le classi e le interfacce fondamentali per la creazione di elementi grafici
- è stata inserita nelle **API** standard di Java per lo sviluppo di applicazioni **GUI** in modo veloce ed efficiente
- la creazione di un elemento AWT comporta la chiamata alla procedura del sistema operativo per la creazione dello stesso elemento, rendendolo così graficamente uguale all'elemento nativo
  - la stessa finestra se creata su Microsoft Windows assomiglia a una finestra di Windows, se creata su una piattaforma Mac assomiglia a ogni altra finestra Mac

- Swing è un framework per Java, appartenente alle Java Foundation Classes (*JFC*) e orientato allo sviluppo di interfacce grafiche
- parte delle classi del framework Swing sono implementazioni di widget (oggetti grafici) come caselle di testo, pulsanti, pannelli e tabelle
- Swing non riutilizza gli elementi del sistema operativo, ma implementa il disegno e la gestione dell'interazione dell'utente completamente con codice Java (l'aspetto dei componenti visuali di Swing è *uniforme* sulle diverse piattaforme)
- le principali classi di Swing sono comprese nel package **`javax.swing`**

- la classe principale per implementare una finestra in Swing è `JFrame`  
`JFrame finestra = new JFrame("Prova");`
- per rendere visibile una finestra si utilizza il metodo `setVisible()`  
`finestra.setVisible(true);`
- per determinare la posizione e la dimensione di una finestra:  
`finestra.setLocation(75, 50);`  
`finestra.setSize(300, 250);`
- ogni finestra ha i pulsanti di riduzione, ripristino e chiusura che risultano attivi. Per modificare il comportamento del pulsante di chiusura è possibile utilizzare:  
`finestra.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);`

- il computer misura le coordinate dello schermo utilizzando i pixel
- maggiore è la risoluzione, più numerosi sono gli elementi grafici e le finestre che possono essere disegnati a video
- lo schermo è accessibile per coordinate simili a quelle del piano cartesiano, con la differenza che il punto iniziale (0,0) non è posizionato in basso a sinistra, ma in alto a sinistra



- un'etichetta può contenere un testo e/o un'immagine
- è possibile decidere l'allineamento del testo nello spazio disponibile (a sinistra, a destra o al centro)  

```
JLabel etichetta = new JLabel("Testo di prova");
```
- un componente visuale deve essere inserito in un *contenitore* che consenta la sua visualizzazione (esempio in una finestra)
- per effettuare questa operazione è necessario chiamare il metodo **getContentPane()** per ottenere il contenitore della finestra, quindi chiamare il metodo **add()** per aggiungere il componente  

```
finestra.getContentPane().add(etichetta);
```
- è possibile inserire direttamente il componente alla finestra  

```
finestra.add(etichetta);
```

- un campo di testo contiene informazioni testuali che possono essere modificate dall'utente

```
JTextField campoTesto = new JTextField();
```
- il testo del campo di testo può essere ottenuto con il metodo **getText()**
- il metodo **setText()** consente invece di impostarlo

```
JTextField campoTesto = new JTextField();  
campoTesto.setText( "Testo di prova" );
```
- è possibile impostare una dimensione indicativa dello spazio disponibile per la digitazione del testo da parte dell'utente, misurato in numero di colonne

```
JTextField campoTesto = new JTextField(10);
```
- oppure, usando il metodo **setColumns()**:

```
JTextField campoTesto = new JTextField();  
campoTesto.setColumns(10);
```

- una lista è caratterizzata da un riquadro con un elenco di informazioni, si utilizza quando si desidera mostrare un elenco di dati
- la modalità più semplice per costruire una lista consiste nel passare al costruttore un array di oggetti
- la lista delle voci può essere definita editando la proprietà model

```
String[] v = { "Uno", "Due", "Tre" }  
JList lista = new JList(v);
```





- **getSelectedIndex()** ritorna l'indice dell'elemento selezionato (gli indici partono da zero)
- **getSelectedValue()** ritorna il valore selezionato (l'oggetto presente nella lista)
- se gli elementi selezionati sono più di 1:
  - **getSelectedIndices()** ritorna un array di interi con gli indici selezionati
  - **getSelectedValues()** ritorna un array di oggetti che contiene gli elementi selezionati
- di default un oggetto **JList** consente la selezione di più di un elemento (questo aspetto è configurabile)

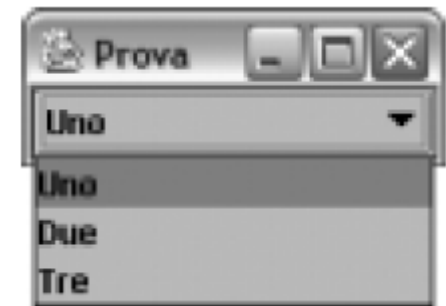
# combo box (caselle combinate)

- la caratteristica è quella di unire un campo di testo e una lista di selezione.

```
String[] v = { "Uno", "Due", "Tre" };
```

```
JComboBox combo = new JComboBox(v);
```

- per sapere quale elemento è stato scelto, è possibile utilizzare il metodo **getSelectedIndex()**, che ritorna l'indice dell'elemento selezionato, oppure **getSelectedItem()**, che ritorna l'oggetto selezionato
- per rendere editabile il componente:  
**combo.setEditable(true);**



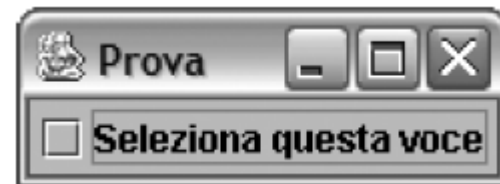
- **addItem(Object)** aggiunge il parametro al termine dell'elenco
- **insertItem(Object,int)** inserisce l'oggetto alla posizione specificata
- **getItemAt(int)** ritorna l'oggetto presente all'indice passato come parametro
- **removeAllItems()** rimuove tutti gli oggetti dall'elenco
- **removeItemAt(int)** rimuove l'elemento alla posizione passata come parametro
- **removeItem(Object)** rimuove l'oggetto passato come parametro
- **getItemCount()** ritorna il numero di elementi nella lista

- le caselle di selezione sono un tipo particolare di pulsante

`JCheckBox` selezione;

```
selezione = new JCheckBox("Seleziona questa voce");
```

- per sapere se la casella è spuntata o meno, è possibile utilizzare il metodo `isSelected()`, che ritorna un valore booleano



- *tutti i componenti* visuali della libreria Swing supportano la funzionalità di base **tooltip**
- **tooltip**, un riquadro di piccole dimensioni contenente un testo, che viene visualizzato quando l'utente indugia con il mouse sul componente senza interagire con esso
- per impostare il testo di un tooltip di qualsiasi componente si utilizza il metodo **setToolTipText (String)**

- un altro aspetto *comune a tutti i componenti* è la possibilità di abilitare o disabilitare il componente
  - è possibile utilizzare il componente al solo scopo di visualizzazione, impedendo le modifiche dei dati da parte dell'utente
- lo stato di abilitazione/disabilitazione è impostato con il metodo **setEnabled(boolean)**

- un contenitore è un componente visuale che ne ***contiene*** altri
  - la ***finestra*** è un contenitore
  - un altro contenitore molto utilizzato è il ***pannello*** (**Jpanel**)
- i contenitori dispongono del metodo **add()**, che consente di aggiungere un componente o altri contenitori

```
JLabel etichetta = new JLabel("Testo di prova");  
JPanel pannello = new JPanel();  
pannello.add(etichetta);  
JFrame finestra = new JFrame("Prova");  
finestra.getContentPane().add(pannello);
```

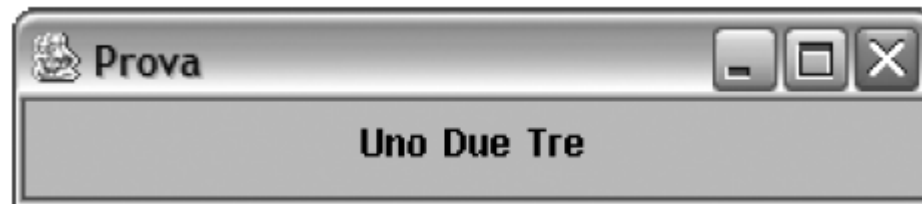
- la disposizione dei componenti visuali all'interno di un contenitore è coordinata da un layout manager
- le finestre e i pannelli hanno un layout manager predefinito
- i layout manager sono realizzati da classi che implementano l'interfaccia **LayoutManager**
- per impostare il layout manager di un pannello, è possibile passare l'oggetto che lo implementa nel costruttore di **JPanel**, come nell'esempio:  

```
FlowLayout flowLayout = new FlowLayout();  
JPanel pannello = new JPanel(flowLayout);
```
- in alternativa, è possibile utilizzare il metodo **setLayout()**



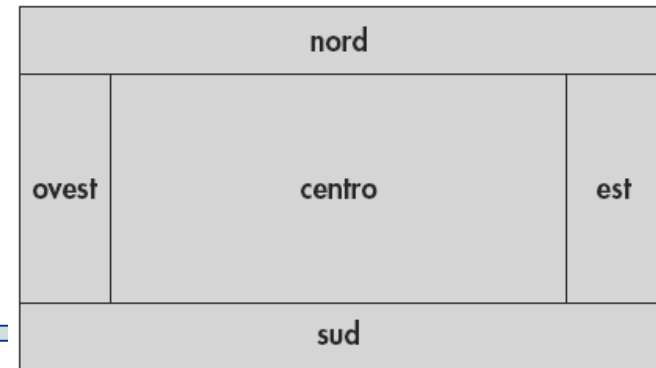
- distribuisce i componenti uno di seguito all'altro, lasciando uno spazio configurabile tra un oggetto e l'altro

```
JFrame fin = new JFrame("Prova");  
fin.getContentPane().setLayout(new FlowLayout());  
fin.getContentPane().add(unolabel);  
fin.getContentPane().add(duelabel);  
fin.getContentPane().add(trelabel);
```



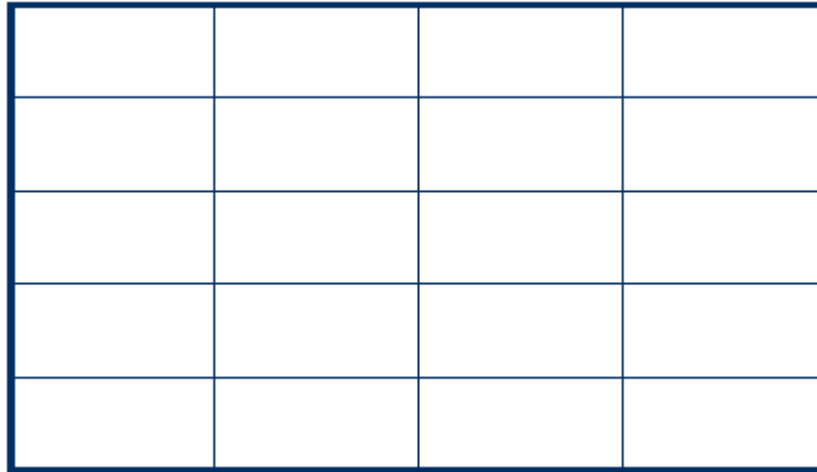
- suddivide lo spazio disponibile in cinque aree: nord, sud, est, ovest e centro
- è possibile inserire un singolo componente in ciascuna area
- i componenti a nord e a sud ottengono lo spazio desiderato in orizzontale e lo spazio minimo in verticale
- i componenti ai lati invece ottengono lo spazio desiderato in verticale e quello minimo in orizzontale
- il componente al centro ottiene tutto lo spazio restante

```
finestra.getContentPane().add(nordLabel,  
BorderLayout.NORTH);
```



- **GridLayout** distribuisce i componenti in una *griglia* invisibile dotata di un numero di celle definite dall'utente
- lo spazio disponibile è distribuito in modo uniforme a tutti i componenti contenuti
- quando si istanzia un oggetto GridLayout è necessario indicare il numero di *righe* e *colonne* che dovrà avere la griglia

```
GridLayout gridLayout = new GridLayout(5, 4);
```




- i pannelli a scorrimento sono contenitori che si occupano di visualizzare il contenuto di un altro componente in un'area di spazio inferiore rispetto a quella che occuperebbe l'intero componente e di fornire gli strumenti necessari alla navigazione in tutto il contenuto, generalmente con barre di scorrimento

```
JList lungaLista = new JList(v);  
JScrollPane pannelloScorrevole = new  
JScrollPane(lungaLista);  
JFrame finestra = new JFrame("Prova");  
finestra.getContentPane().add(pannelloScorrevole);
```

- il pannello a schede consente di sfruttare la stessa area di visualizzazione per contenere un alto numero di componenti
- i diversi elementi sono organizzati in diverse schede che vengono visualizzate sullo schermo una alla volta
- il contenitore Swing che implementa il pannello a schede è **JTabbedPane** che dispone del metodo **addTab()** per aggiungere nuove schede, specificandone il nome e il componente visuale contenuto.

```
JTabbedPane pannelloSchede = new JTabbedPane();  
pannelloSchede.addTab( "Uno", new JLabel("Uno") );  
pannelloSchede.addTab( "Due", new JLabel("Due") );  
pannelloSchede.addTab( "Tre", new JLabel("Tre") );
```



- la classe **JOptionPane** definisce i seguenti metodi statici la cui invocazione visualizza una finestra popup:

Metodo	Descrizione
<i>showConfirmDialog</i>	Visualizza una finestra di conferma con le opzioni Sì/No
<i>showInputDialog</i>	Visualizza una finestra di richiesta di un input: il metodo restituisce una stringa contenente l'input fornito dall'utente
<i>showMessageDialog</i>	Visualizza una finestra per la comunicazione di un messaggio
<i>showOptionDialog</i>	Unificazione dei tre tipi precedenti