Unità E2

Java e le interfacce grafiche

Obiettivi

- Conoscere le origini e l'approccio di Swing
- Creare una semplice applicazione a finestre.
- Distinguere l'organizzazione delle coordinate tra il sistema cartesiano e il monitor del computer
- Saper utilizzare etichette,campi di testo, liste, combo box, caselle di selezione
- Conoscere il concetto di contenitore
- Conoscere il concetto di layout manager e sapere utilizzare quelli principali presenti nella piattaforma
- Saper comporre interfacce complesse a partire dai componenti di base
- Conoscere i pannelli principali

Swing

- Swing è l'evoluzione di JFC (Java Foundation Classes per ovviare alle limitazioni di AWT
- Le AWT (Abstract Windowing Toolkit) sono classi che si limitano a richiamare i corrispondenti elementi grafici del sistema operativo ospite. Per esempio, in un programma Windows, un campo di testo AWT non fa altro che invocare il campo di testo nativo del sistema operativo di Microsoft.
- 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**

Look&feel

- Le interfacce Swing supportano il concetto di look&feel configurabile.
- Il termine "look&feel" è utilizzato per descrivere l'aspetto e le modalità di interazione specifiche di un sistema operativo.
- Esistono diversi look&feel disponibili: Windows, GTK, Mac OS X.
- Quando un'applicazione Swing utilizza uno specifico look&feel, i suoi componenti visuali sono visualizzati, e si comportano, come se fossero componenti nativi.
- Swing dispone anche di un look&feel nativo per la piattaforma Java, chiamato **Metal.**

Finestre

- 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 );
```

Coordinate sullo schermo

- Il computer misura le coordinate dello schermo utilizzando i pixel.
- Risoluzione tipiche per personal computer sono 800×600 pixel, 1024×768 oppure 1280×854.
- 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.

Etichette

- 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);
```

Campi di testo

- 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" );
```

- E' 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);
```

Liste

- Una lista è caratterizzata da un riquadro con un elenco di informazioni, si utilizza quando si desidera mostrare un elenco di dati (esempio l'elenco delle regioni italiane).
- La modalità più semplice per costuire una lista consiste nel passare al costruttore un array di oggetti.

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

Liste - metodi

- **`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:
- **`getSelectedIndicies()`** 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, ma questo aspetto è configurabile

Combo box (caselle combinate)

- La caratteristica peculiare 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);
```

Combo box – metodi principali

- 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.

Caselle di selezione

- Nella logica di Swing, 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.

Tooltip

- Tutti i componenti visuali della libreria Swing supportano la funzionalità di base.
- 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)`**.

Abilitazione

- Un altro aspetto comune a tutti i componenti è la possibilità di abilitare o disabilitare il componente.
- In questo modo, è possibile utilizzarlo solo a scopo di visualizzazione, impedendo le modifiche dei dati da parte dell'utente.
- Lo stato di abilitazione/disabilitazione è impostato con il metodo **`setEnabled(boolean)`.**

Contenitori

- Un contenitore è un componente visuale che ne contiene altri.
- La finestra è essa stessa un contenitore.
- Un altro contenitore molto utilizzato è il **pannello, che le Swing implementano con** la classe **JPanel.**
- I contenitori dispongono del metodo **add(), che consente di aggiungere un componente** o altri contenitori.
- Esempio:

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

Layout manager

- La disposizione dei 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()`.**

FlowLayout

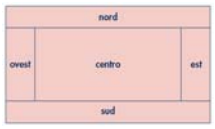
- distribuisce i componenti uno di seguito all'altro, lasciando uno spazio configurabile tra un oggetto e l'altro.

```
JFrame finestra = new JFrame("Prova");
finestra.getContentPane().setLayout( new FlowLayout() );
finestra.getContentPane().add(unoLabel);
finestra.getContentPane().add(dueLabel);
finestra.getContentPane().add(treLabel);
```

BorderLayout

- 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.

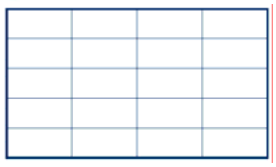
```
finestra.getContentPane().add(nordLabel,
    BorderLayout.NORTH);
```



GridLayout

- 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);
```



Pannelli a scorrimento

- 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;
```

Pannelli a schede

- 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") );
```