

Programmazione guidata dagli eventi e interfaccia grafica

1 L'interfaccia per l'utente

I programmi eseguiti da un elaboratore possono essere divisi in due grandi categorie: i programmi che eseguono i loro compiti senza richiedere l'intervento degli utenti, e altri che hanno bisogno di interagire con gli utenti stessi.

Per esempio, un programma che deve tenere aggiornato un orologio, non ha bisogno di alcun input per eseguire i suoi compiti. Al contrario, un programma che esegue i calcoli sui valori inseriti da un utente e ne restituisce il risultato, ha bisogno di interagire con l'utente. Il programma deve essere in grado di chiedere all'utente l'inserimento dei valori e nello stesso tempo deve prevedere le modalità con cui mostrare i risultati dell'elaborazione.

La parte di un'applicazione che interagisce con l'utente prende il nome di **interfaccia utente**.

La presenza di questa interfaccia distingue i programmi interattivi da quelli non interattivi. L'interfaccia utente indica tutto ciò che serve per la comunicazione tra un programma e il suo utilizzatore (apparecchiature hardware e risorse software). In questa definizione rientra quello che l'utente vede sullo schermo, quello che sente oppure quello che può toccare. Il mouse è una parte dell'interfaccia utente, come pure uno schermo sensibile al tatto (*touchscreen*). Questi rientrano sia nella parte visiva che tattile di un'interfaccia. Il microfono e i programmi che gestiscono la voce fanno parte di un'interfaccia sonora.

È possibile classificare le interfacce utente in base agli strumenti che usano per interagire con gli utenti. Supponendo di considerare lo schermo come l'elemento principale di interazione tra le persone e l'elaboratore, le interfacce si possono distinguere in *interfacce a caratteri* e *interfacce grafiche*.

Le **interfacce a caratteri** sono molto semplici perché possono visualizzare solo i caratteri. Solitamente, l'interazione avviene inserendo i comandi attraverso la tastiera.

Le **interfacce grafiche** sono più complesse da realizzare ma rendono più piacevole e semplice il modo con cui l'utente interagisce. L'interfaccia grafica è composta da immagini, icone, finestre, pulsanti e molti altri elementi. Le interfacce grafiche si indicano con il termine **GUI**, acronimo di *Graphical User Interface*.

Le pagine seguenti sono dedicate all'interfaccia di tipo GUI e al modo con il quale può essere realizzata nel linguaggio Java.

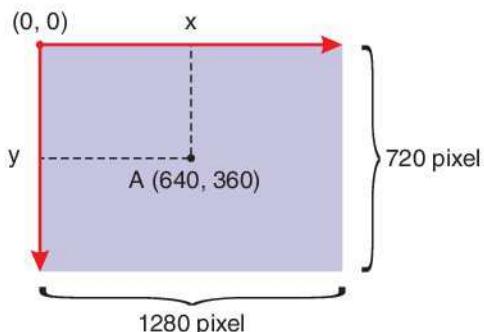
Le interfacce grafiche sono un'evoluzione delle interfacce a caratteri. Questo passaggio è stato reso possibile dallo sviluppo dei terminali video e dei dispositivi di puntamento oltre che da una riduzione del loro costo. Si è passati da schermi che visualizzavano solo caratteri a schermi grafici.

La videata degli schermi grafici è composta da una matrice di punti, chiamati **pixel** (*picture element*). A ogni pixel può essere associato un colore e la loro unione forma l'immagine completa. Le dimensioni della matrice indicano la risoluzione del particolare schermo. Dimensioni comuni in pixel per gli attuali monitor sono 1366x768, 1280x720 (*HD*), 1920x1080 (*Full HD*) dove il primo numero indica il numero di pixel che compongono una linea orizzontale, mentre il secondo numero indica i pixel presenti su una linea verticale.

Queste dimensioni sono indicate come **risoluzione** dello schermo.

Un punto all'interno di una videata può essere indicato usando una coppia di numeri (x,y). Queste rappresentano le coordinate del punto.

Il sistema di riferimento che si adotta con lo schermo ha come origine il punto (0,0) che si trova in alto a sinistra. Spostandosi verso destra viene incrementato il valore delle x, mentre spostandosi in basso viene incrementato il valore delle y.



In figura viene mostrato il sistema di riferimento di uno schermo di dimensione 1280x720. Il punto A, che si trova nel centro dello schermo, ha come coordinate (640,360).

Il principale dispositivo di puntamento che viene usato con le interfacce grafiche è il **mouse**. Le interfacce che supportano questo dispositivo vengono anche chiamate *point and click* (*punta e fai clic* sul tasto del mouse). Questo dispositivo consente infatti di muovere un puntatore che viene visualizzato sullo schermo. Il puntatore si trova, in ogni istante, in un certo punto dello schermo individuato dalla sue coordinate lungo i due assi. Dopo aver posizionato il puntatore, si possono premere i tasti del mouse per attivare particolari operazioni: il mouse può essere usato per spostarsi da una finestra all'altra, per indicare o selezionare gli oggetti oppure per attivare i programmi o le azioni dell'utente.

2 Gli elementi dell'interfaccia grafica

L'elemento principale di un'interfaccia grafica è la **finestra**. Rappresenta un'area dello schermo all'interno della quale vengono eseguite le applicazioni. Può coincidere o meno con l'intera videata dello schermo a seconda di come vengono impostate le sue dimensioni. La finestra è usata per contenere e posizionare gli altri elementi grafici. Tra i compiti della progettazione di un'interfaccia grafica c'è la scelta della disposizione (in inglese **layout**) degli elementi grafici all'interno di una finestra. Questa scelta non è banale e da questa può dipendere il successo o meno dell'applicazione.

Una finestra possiede alcune funzionalità caratteristiche: si può ingrandire a pieno schermo, ridimensionare oppure ridurre a icona; la sua chiusura corrisponde solitamente alla terminazione del programma.

Un altro elemento grafico è il **pulsante** (o *bottone*). Può essere etichettato con una stringa oppure con un'immagine. Le applicazioni associano ai pulsanti le particolari azioni che l'utente è abilitato a svolgere. L'attivazione di queste azioni avviene con un clic del mouse sul pulsante.

Gli elementi grafici che gestiscono l'inserimento del testo sono le *caselle di testo* e le *aree di testo*.

La **casella di testo** è formata da una cella in cui è possibile inserire solo una riga, è usata dalle applicazioni per richiedere all'utente l'inserimento di numeri o stringhe.

L'area di testo, invece, è formata da più linee, viene utilizzata per inserire o per mostrare le righe di testi particolarmente lunghi.

Un'applicazione può interagire con l'utente offrendo la possibilità di scegliere tra un numero prefissato di alternative. Questa scelta si può realizzare all'interno di un'interfaccia grafica usando diversi elementi grafici.

La **casella combinata** mostra le alternative tramite una casella di testo che incorpora un menu a tendina. L'utente, usando il mouse, può aprire e scorrere la tendina per indicare una delle alternative.

Un altro modo molto comune per effettuare le scelte, soprattutto quando sono legate all'esecuzione di particolari operazioni, è rappresentato dai **menu**. I menu sono solitamente posizionati nella parte alta della finestra. Si attivano con un clic del mouse che mostra l'elenco di voci contenute nel menu: scegliendo una voce si attiva l'azione corrispondente.

Il **pulsante di opzione** (avente la forma di un piccolo cerchio) permette di effettuare la scelta tra valori booleani, cioè gestisce solo due stati, corrispondenti al valore *true* se il cerchio è annerito e al valore *false* in caso contrario. Si possono raggruppare più pulsanti di opzione in un unico gruppo: in questo modo può essere effettuata una scelta esclusiva tra gli elementi del gruppo.

La figura mostra, a partire dall'alto a sinistra, un'etichetta, un pulsante, una casella di testo, un'area di testo, una casella combinata e due pulsanti di opzione.



Altri elementi grafici che possono far parte di un'interfaccia grafica sono: le barre di scorrimento, le immagini e i menu *popup*, cioè i menu di scelte che vengono aperti all'interno della finestra (come i menu di scelta rapida attivabili in Windows con il tasto destro del mouse).

Avendo a disposizione tutte le componenti precedentemente elencate, è possibile combinarle per **progettare un'interfaccia grafica**.

La realizzazione di una GUI dipende molto dal tipo di programma a cui è destinata. Il numero di menu, di pulsanti, di caselle di testo va scelto in base a quello che l'applicazione deve fare. Se per esempio è richiesto l'inserimento di due valori, sarà necessario predisporre due caselle di testo.

Anche la disposizione degli elementi grafici ha una sua importanza che è sempre legata agli scopi del programma. Un programma per l'elaborazione dei testi riserva la parte centrale a un'area di testo, disponendo i menu e i pulsanti nella parte superiore o inferiore della finestra.

Per costruire un'applicazione grafica bisogna quindi eseguire due operazioni:

- individuare gli elementi grafici che servono
- disporre questi elementi all'interno di una finestra.

Queste operazioni devono essere svolte durante la fase di analisi del problema. È utile dare una descrizione dell'interfaccia grafica che si vuole utilizzare, indicando quali sono gli elementi e quale posizione essi devono occupare.

Un linguaggio di programmazione che gestisce le interfacce grafiche deve consentire l'utilizzo degli elementi grafici precedentemente descritti. Il linguaggio Java consente al programmatore di creare gli oggetti da inserire nella GUI; offre anche diversi modi per disporre e dimensionare questi oggetti all'interno di una finestra.

I programmi Java, presentati nei capitoli precedenti, utilizzano un'interfaccia a caratteri. L'input e l'output vengono eseguiti dal *Prompt dei comandi*. Questo modo di interagire con il programma è semplice e molto limitato. Le uniche cose che si possono fare sono la visualizzazione di messaggi testuali e l'inserimento di stringhe utilizzando la tastiera. Non è permesso l'utilizzo del colore, del mouse, di immagini, dei pulsanti e di tutti gli altri elementi grafici.

Le applicazioni che usano una GUI risultano più complesse e di dimensioni maggiori rispetto a quelle realizzate con un'interfaccia a caratteri. Si ha però il vantaggio di creare programmi *user friendly* che facilitano l'interazione con l'utente.

I paragrafi successivi mostrano come costruire un'interfaccia grafica usando l'ambiente di sviluppo grafico **NetBeans**.



GLI ELEMENTI GRAFICI COME OGGETTI DELLA OOP

I diversi elementi grafici della GUI possono essere rappresentati in modo naturale utilizzando gli **oggetti**.

Infatti una finestra può essere vista come un oggetto i cui *attributi* sono la dimensione, la posizione e il titolo. I *metodi* di una finestra consentono di modificarla, cambiandone le dimensioni, spostandola in primo o in secondo piano, rendendola visibile o nascondendola. Anche le caselle di testo possono essere considerate degli oggetti: hanno un *attributo* che è usato per memorizzare il valore digitato nella casella; i *metodi* consentono di cancellare il contenuto della casella, rendere la casella editabile o meno, oppure impostare il numero di caratteri che possono essere inseriti nella casella.

In generale tutti gli elementi grafici possono essere considerati oggetti. All'interno di un programma che usa una GUI, le operazioni che coinvolgono gli oggetti grafici vengono attivate con i metodi degli oggetti coinvolti. Per esempio, l'operazione di chiusura di una finestra viene eseguita con il metodo di chiusura presente nell'oggetto che rappresenta la finestra.

La programmazione ad oggetti è quindi l'ambiente ideale per lo sviluppo di interfacce utente di tipo grafico. Ogni elemento grafico è un oggetto e ha una sua classe corrispondente: istanziando queste classi si possono creare gli oggetti da inserire nell'interfaccia grafica per l'utente.

3 Programmazione guidata dagli eventi

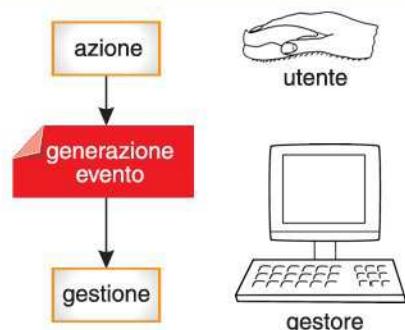
Un'interfaccia grafica non può essere realizzata usando soltanto gli elementi grafici. Questi rappresentano solo figure, linee colorate, scritte che vengono visualizzate sullo schermo. L'utente le può vedere e può usare il mouse o la tastiera per interagire con esse. Occorre però aggiungere le modalità con le quali viene gestita l'interazione tra l'utente e l'interfaccia.

La OOP, per gestire le interfacce grafiche, usa la **programmazione guidata dagli eventi**: a ogni interazione dell'utente viene associato un evento che deve essere gestito richiamando un'azione appropriata.

Un **evento** è un avvenimento asincrono, cioè può manifestarsi senza rispettare tempi prefissati.

Per esempio l'accensione del televisore o l'apertura del frigorifero sono eventi, mentre non possiamo considerare come evento il compleanno di una persona, perché si verifica a una scadenza precisa.

Un evento, riferito a una GUI sul computer, si verifica in seguito all'interazione dell'utente: possibili eventi sono l'apertura o la chiusura di una finestra, il clic su un pulsante oppure la pressione di un tasto del mouse.



La generazione di un evento è causata da un'azione effettuata dall'utente. L'evento generato contiene quindi tutte le informazioni riguardanti questa azione. Una volta generato un evento, si deve provvedere alla sua gestione, che viene demandata ad un **gestore di eventi**.

Per esempio, l'evento *accensione del televisore* viene generato in seguito all'azione compiuta premendo un particolare tasto. La risposta a questo evento, viene gestita dal televisore che si attiva per visualizzare un determinato canale televisivo.

Un utente che preme un tasto del mouse causa la generazione dell'evento *pressione del tasto del mouse*: esso viene gestito eseguendo un insieme di operazioni associate a quell'evento. In un programma che usa una GUI gli eventi possono nascere dall'interazione dell'utente con gli elementi grafici. La finestra può generare eventi quando viene chiusa, ridotta a icona oppure ingrandita. I pulsanti generano un evento se si fa clic su di essi.

La parte di un'applicazione, destinata alla gestione degli eventi, viene esplicitamente dichiarata. Se da un lato la generazione degli eventi è causata dall'interazione dell'utente, dall'altro la gestione è affidata a un gestore di eventi.

Un **gestore**, detto anche *handler*, è la parte dell'applicazione software che si preoccupa di dare una risposta in base al tipo di evento ricevuto.

Poiché ogni elemento grafico può generare un evento, bisogna associare ad ognuno un gestore. Se non si vuole gestire l'evento, basta non specificare il gestore. I gestori di un particolare elemento grafico sono *registriati* per gestire quell'elemento e si preoccupano di eseguire le operazioni in risposta ai vari eventi che ricevono.

Ogni volta che un evento viene generato, l'elemento grafico che lo genera richiama un metodo particolare del gestore che è stato registrato per esso.

L'obiettivo fondamentale di un gestore degli eventi, e in generale di un sistema di gestione degli eventi, è di rispondere istantaneamente all'azione di un utente. Se il gestore è occupato a gestire un evento, gli eventi che vengono successivamente generati non devono essere scartati, devono aspettare il loro turno per essere gestiti quando il gestore si libera. Questa strategia viene implementata con una *coda*: gli eventi, man mano che vengono generati, vengono inseriti in modo ordinato in una coda di eventi. Da questa vengono poi prelevati e assegnati al loro gestore. La gestione di questa coda deve essere fatta in modo efficiente per rispondere tempestivamente alle interazioni degli utenti.

4 Le librerie grafiche AWT e Swing

La parte grafica di Java è contenuta nei package **java.awt** e **javax.swing**. È quindi necessario importare entrambe le librerie in ogni programma in cui viene utilizzata la grafica per realizzare una GUI.

L'**AWT** (*Abstract Window Toolkit*) è il sistema utilizzato da Java per definire un insieme di elementi grafici indipendenti dalla piattaforma su cui vengono visualizzati. Questo meccanismo garantisce la portabilità delle applicazioni Java tra piattaforme diverse.

Windows ha un suo modo di visualizzare le finestre, i pulsanti e gli altri elementi grafici che è diverso dal modo usato da *Unix/Linux* oppure da *Mac OS*, ma per tutte queste piattaforme, la costruzione dell'interfaccia grafica prevede l'utilizzo delle stesse classi definite nella AWT.

Swing rappresenta l'evoluzione di AWT nella costruzione delle interfacce grafiche. Il package `javax.swing` mette a disposizione contenitori e componenti, come il package AWT, ma si distingue da quest'ultimo per l'aggiunta di nuove componenti, metodi e funzionalità: per esempio, i pulsanti Swing possono contenere un'immagine anziché una sola scritta, come per i pulsanti AWT. In generale, le componenti Swing sono molto più numerose e sono da preferirsi nella realizzazione delle interfacce utente rispetto ad AWT.

In Java, l'interfaccia utente è formata da due elementi principali: le *componenti* e i *contenitori*. Una **componente** è un oggetto con una rappresentazione grafica: la sua caratteristica principale è di offrire un'interazione con l'utente. Esempi di componenti sono: i pulsanti, i menu, le caselle per l'inserimento del testo, le barre di scorrimento e così via. Un pulsante, sul quale viene premuto il tasto del mouse, interagisce cambiando forma e facendo capire che è stato azionato. Un campo di testo, a seguito dei tasti digitati sulla tastiera, interagisce mostrando i caratteri corrispondenti ai tasti.

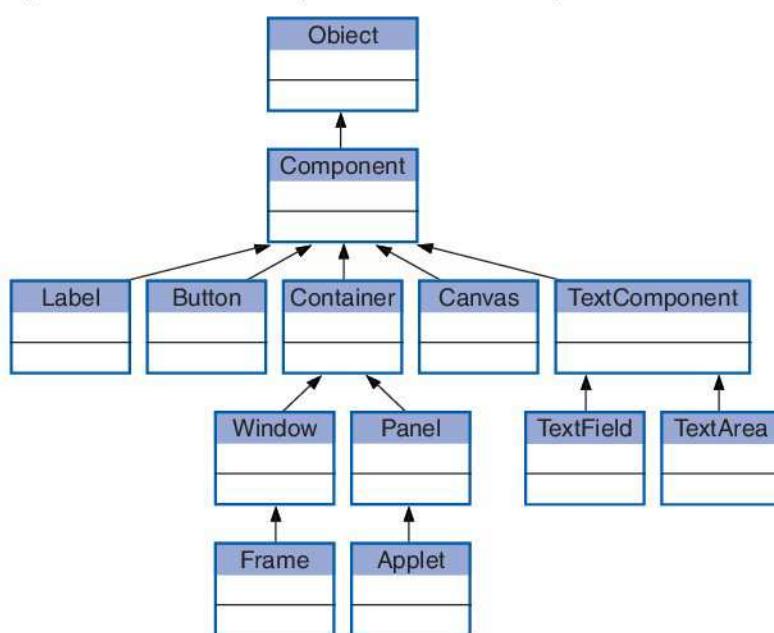
Ogni componente gestisce il modo con cui viene visualizzata, conosce il suo contenuto e reagisce alle eventuali interazioni con l'utente.

Unendo varie componenti si costruisce un'interfaccia utente grafica.

Un **contenitore** è un oggetto che può contenere le componenti. Usando opportuni metodi, si possono aggiungere o togliere le componenti dal contenitore. Il compito del contenitore è quello di posizionare e dimensionare le componenti all'interno del contenitore stesso (**layout** delle componenti). Esistono vari modi con cui questa operazione viene eseguita e dipende dal tipo di gestore del *layout* (in inglese *Layout Manager*) che viene assegnato a quel particolare contenitore. Un contenitore comunemente usato è la **finestra**: è rappresentata da un'area rettangolare, con un titolo, i pulsanti che gestiscono la chiusura e il ridimensionamento e può contenere varie componenti, per esempio un'area di testo oppure altri pulsanti.

Tutte le componenti, insieme al contenitore, possono essere viste come un'unica entità e considerate come una componente speciale. In questo modo, il contenitore diventa a sua volta una componente e può essere inserita in altri contenitori.

Le classi di Java che realizzano le componenti e i contenitori, sono organizzate in una **gerarchia delle componenti** che ha come padre la classe **Component**.



In figura, per semplificare la rappresentazione, sono mostrate solo le componenti della libreria AWT: *Label*, *Button*, *Canvas*, *Frame*, *Panel*, *TextField* e *TextArea*.

La classe *Component* è una classe astratta e da questa derivano tutte le classi che realizzano concretamente una componente. Ogni componente viene realizzata creando un'istanza di queste classi.

Si noti che la classe **Container** è una sottoclasse astratta derivata dalla classe *Component* e rappresenta i contenitori. Le sue sottoclassi implementano concretamente i contenitori che possono essere utilizzati per costruire un'interfaccia grafica. Tra i contenitori è presente anche la classe *Applet* che sarà presentata nel prossimo capitolo.

Nella libreria Swing i nomi delle componenti e dei contenitori iniziano con la lettera J e si possono classificare in:

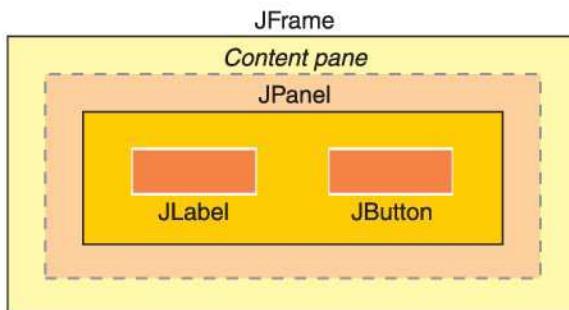
- **contenitori principali** (*top-level container*): finestra principale (*JFrame*), finestra di dialogo (*JDialog*) e finestra di applet (*JApplet*);
- **contenitori intermedi** (*intermediate container*, indicati anche con il termine *pane*), cioè contenitori di altre componenti: pannello (*JPanel*), pannello con barre di scorrimento (*JScrollPane*) o scheda con etichette (*JTabbedPane*);
- **componenti atomiche**: etichetta (*JLabel*), pulsante (*JButton*), casella di testo (*JTextField*), area di testo (*JTextArea*), combo box (*JComboBox*), tabelle (*JTable*).

I **pannelli** (*pane*) hanno lo scopo di organizzare le componenti atomiche nella finestra, semplificandone il posizionamento.

A differenza di AWT, ogni contenitore *top-level* di Swing contiene in senso figurato un contenitore intermedio, detto **content pane** (pannello del contenuto).

Le componenti dell'interfaccia non possono essere inserite direttamente nel contenitore principale, ma solo nel contenitore intermedio. Fa eccezione la barra dei menu che, per convenzione, è collocata in una posizione predefinita della finestra principale, al di fuori del contenitore intermedio.

Lo schema seguente rappresenta l'organizzazione delle componenti Swing per una generica finestra contenente un pannello, al cui interno sono collocati un'etichetta e un pulsante.



La costruzione delle interfacce utente, tramite le componenti e i contenitori AWT e Swing, può essere fatta in modo visuale, usando l'ambiente di programmazione *NetBeans*, oppure in modo testuale scrivendo direttamente il codice Java.

La realizzazione di interfacce utente con *NetBeans* sarà presentata nel paragrafo successivo. Il progetto seguente, invece, mostra come utilizzare le classi AWT e Swing per istanziare gli oggetti grafici e realizzare un'applicazione scrivendo direttamente il codice Java.

PROGETTO 1

Creare una finestra grafica composta da un'etichetta e un pulsante.

L'interfaccia grafica da creare è quella rappresentata con lo schema precedente, e cioè un'etichetta e un pulsante all'interno di un pannello, inserito all'interno di una finestra. Per utilizzare le classi grafiche Swing bisogna importare entrambi i package Swing e AWT nel seguente modo:

```
import javax.swing.*;
import java.awt.*;
```

Per ogni elemento grafico, viene creato un oggetto come istanza della relativa classe. In particolare i contenitori vengono costruiti con le seguenti istruzioni:

```
JFrame f = new JFrame();
JPanel p = new JPanel();
```

Le componenti invece sono costruite istanziando la classe e indicando come parametro il testo da visualizzare nella componente, come mostrano le seguenti istruzioni:

```
JLabel l = new JLabel("Etichetta");
JButton b = new JButton("Bottone");
```

Per aggiungere una componente al contenitore, si usa il metodo **add**. In sequenza, le componenti atomiche (*l* e *b*) vengono prima aggiunte al contenitore intermedio *p*, che a sua volta è aggiunto al contenitore principale *f*.

In Swing, l'aggiunta al contenitore principale viene eseguita tramite un ulteriore contenitore intermedio (oggetto **Container**) che rappresenta il *content pane*, pannello del contenuto della finestra *JFrame*. Il metodo **getContentPane** rende disponibile questo contenitore, al cui interno poi vengono aggiunte le altre componenti nel seguente modo:

```
Container c = f.getContentPane();
c.add(p);
```

In AWT, l'aggiunta al contenitore principale viene eseguita direttamente con l'istruzione:

```
f.add(p);
```

I codici completi per realizzare la finestra con le componenti Swing e con le AWT sono riportati di seguito.

PROGRAMMA JAVA con Swing (*FinestraSwing.java*)

```
import javax.swing.*;
import java.awt.*;

class FinestraSwing
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame();
        JPanel p = new JPanel();
```

```

JLabel l = new JLabel("Etichetta");
JButton b = new JButton("Bottone");
p.add(l);
p.add(b);

Container c = f.getContentPane();
c.add(p);
f.setSize(300,200);
f.setVisible(true);
}
}

```

PROGRAMMA JAVA con AWT (*FinestraAwt.java*)

```

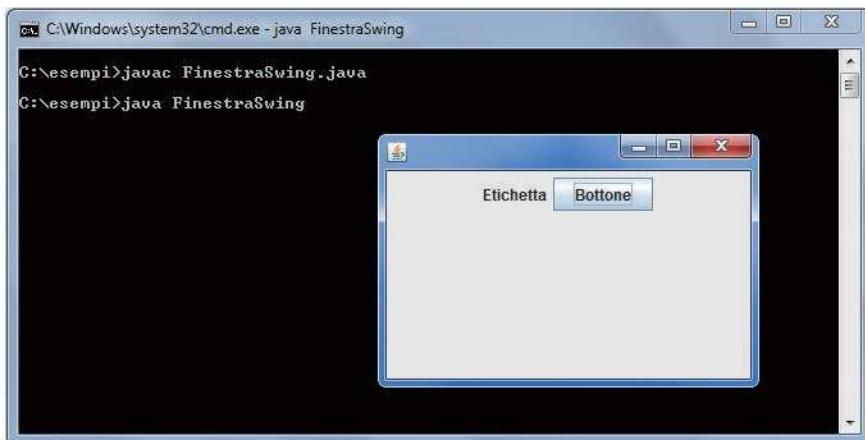
import java.awt.*;
class FinestraAwt
{
    public static void main(String argv[])
    {
        Frame f = new Frame();
        Panel p = new Panel();
        Label l = new Label("Etichetta");
        Button b = new Button("Bottone");
        p.add(l);
        p.add(b);

        f.add(p);
        f.setSize(300,200);
        f.setVisible(true);
    }
}

```

Si noti che il metodo **setSize** imposta le dimensioni della finestra: i due numeri, passati come parametro, rappresentano la larghezza e l'altezza espresse in *pixel*.

La finestra che viene creata rimane non visibile finché viene richiamato il metodo **setVisible(boolean)**. Con il valore booleano *true*, la finestra viene resa visibile, con il valore *false* la finestra viene nascosta.



Il programma genera una finestra con le dimensioni indicate, con un'etichetta e un pulsante. La finestra può essere ridimensionata, ingrandita o ridotta a icona, però non può essere chiusa. Per chiudere la finestra, occorre gestire l'evento di chiusura associando ad esso un'istruzione che causa la fine del programma. Non avendo previsto nel programma la gestione di questa operazione, per chiudere la finestra si deve interrompere l'esecuzione del programma premendo la combinazione di tasti **Ctrl + C** dal *Prompt dei comandi*.

AUTOVERIFICA

Problemi da 1 a 4 pag. 347

**1 I contenitori in AWT**

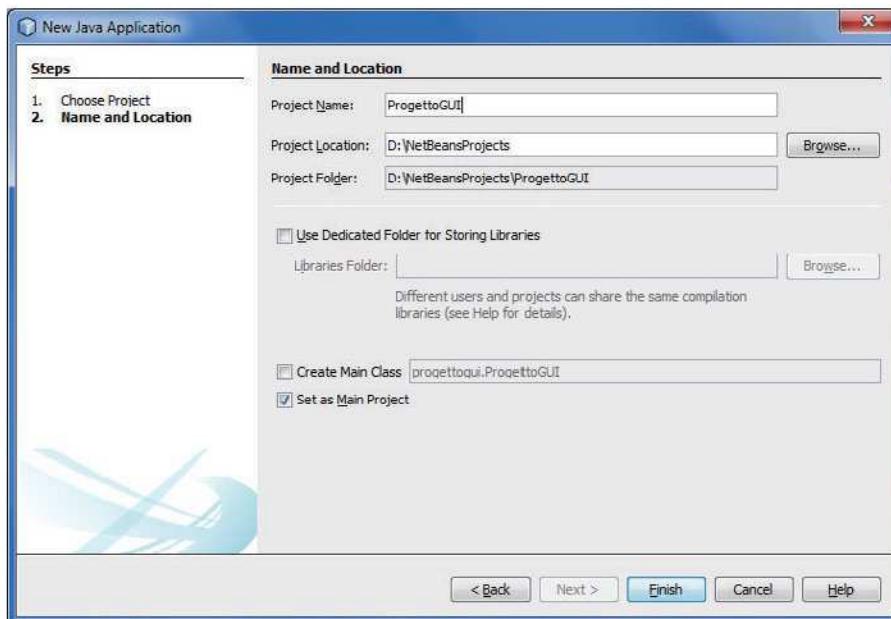
5 L'ambiente di programmazione

La programmazione delle interfacce grafiche richiede che il programmatore definisca gli elementi grafici e li disponga all'interno di una finestra. **L'ambiente di programmazione e di sviluppo grafico** facilita il lavoro del programmatore in quanto offre gli strumenti per disegnare l'interfaccia grafica e vedere in modo immediato l'aspetto dell'applicazione come apparirà all'utente finale. Questi ambienti di programmazione favoriscono uno stile di **programmazione visuale**, in cui il programmatore manipola direttamente gli elementi dell'interfaccia grafica senza la necessità di scrivere il codice Java, che viene autogenerato.

Nelle pagine seguenti sono presentate le modalità operative per realizzare programmi guidati dagli eventi e interfacce grafiche per l'utente attraverso l'ambiente di sviluppo **NetBeans**, già presentato nell'inserto dopo il Capitolo 3.

Per creare un nuovo progetto basato su un'interfaccia grafica, si deve fare clic sul menu **File**, **New Project**, poi nella categoria **Java**, scegliere il tipo di progetto **Java Application** e infine fare clic su **Next**.

Nella successiva pagina, scriviamo il nome del progetto nella casella **Project Name** e verifichiamo che non sia stata selezionata l'opzione **Create Main Class**, in quanto la classe principale verrà creata successivamente.

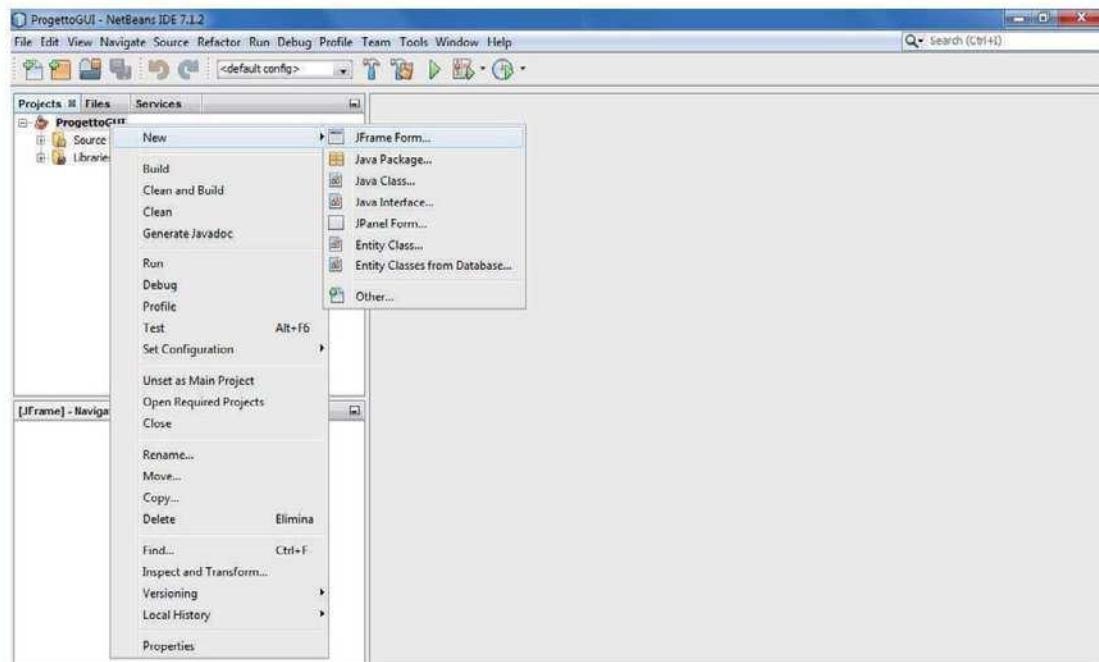


Per completare la creazione del progetto facciamo clic sul pulsante **Finish**. Il nuovo progetto viene visualizzato nella lista dei *Projects* in alto a sinistra.

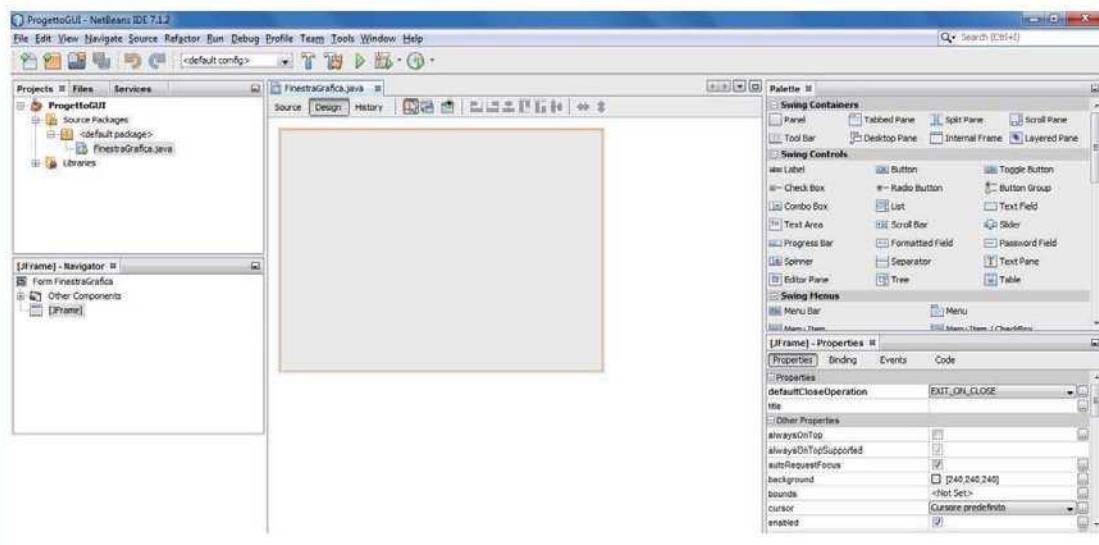
Ciascuna finestra utilizzata nel programma corrisponde a un oggetto di classe **JFrame**, detto anche *contenitore*.

Un **contenitore** (*container*) è una finestra, o più in generale un elemento grafico sul quale il programmatore inserisce gli elementi grafici che l'utente finale vede. Gli oggetti dell'interfaccia grafica che vengono inseriti in un *container* vengono chiamati **componenti** (*component*) o **controlli** (*control*).

Per aggiungere una finestra al progetto si deve fare clic con il tasto destro sul nome del progetto, poi su **New** e infine su **JFrame Form**.



Viene aperta una finestra in cui si deve indicare il nome della classe principale dell'applicazione. Per completare la creazione della finestra facciamo clic sul pulsante **Finish**.



La finestra di lavoro principale mostra, nella parte centrale, l'anteprima dell'unico *container* del progetto appena creato.

Nella parte superiore si trova la **Barra dei menu**. Al di sotto della Barra dei menu si trova la **Barra degli strumenti** contenente un insieme di icone che rendono più veloce l'accesso alle scelte di uso più frequente.

Si può modificare l'insieme delle barre visualizzate facendo clic con il tasto destro sulla Barra degli strumenti e selezionando le barre preferite.

L'area sotto la Barra degli strumenti è solitamente divisa in tre zone (sinistra, centrale, destra) che si caratterizzano per la presenza di **finestre operative**. Ogni finestra può essere spostata tra le varie zone, può essere chiusa e nascosta alla visualizzazione del programmatore. L'elenco delle finestre operative che possono essere visualizzate è contenuto nella voce di menu **Window**.

Come impostazione predefinita, nella zona a sinistra è visualizzata la finestra **Projects** che mostra l'elenco dei file del progetto e le librerie utilizzate. Solitamente è anche presente la finestra **Navigator** che, in base all'elemento selezionato, permette di muoversi velocemente tra le sue componenti. Per esempio, se viene selezionato un file *.java*, nella finestra *Navigator* vengono elencati i metodi presenti nel file.

Nella zona centrale viene mostrato sia il codice che l'anteprima del *container* grafico che si sta progettando. Si possono aprire più file contemporaneamente e navigare tra di essi utilizzando le lingue.

All'interno di ogni linguetta, il pulsante **Source** mostra il codice sorgente, mentre il pulsante **Design** mostra l'area in cui si può costruire in modo visuale l'interfaccia utente.

L'anteprima dell'interfaccia utente viene attivata facendo clic sul pulsante



che si trova alla destra del pulsante *Design*.

Nella zona a destra sono presenti le finestre **Palette** e **Properties**. La finestra *Palette* contiene tutte le componenti standard dell'interfaccia utente (caselle di testo, pulsanti, etichette, ecc.) che possono essere trascinate sul *container* per comporre la maschera visualizzata dall'utente finale.

La finestra *Properties* mostra le caratteristiche degli oggetti grafici che possono essere modificate: per esempio la posizione, la dimensione e il tipo di font.

6 Creazione di applicazioni in NetBeans

In generale, per realizzare un'applicazione dotata di interfaccia grafica, si seguono le seguenti fasi:

1. progettare l'**applicazione**
2. progettare le **finestre** che compongono l'interfaccia per l'utente
3. disegnare l'interfaccia, inserendo tutte le **componenti** (pulsanti, menu, ecc.)
4. definire le **proprietà** degli oggetti (posizione, dimensione, ecc.)
5. associare alle componenti il codice di **gestione degli eventi** (per esempio: quando si fa clic sul pulsante X viene eseguito il metodo Y)
6. eseguire il **test** dell'applicazione.

PROGETTO 2

Creare un'applicazione con interfaccia grafica per calcolare la somma di due numeri.

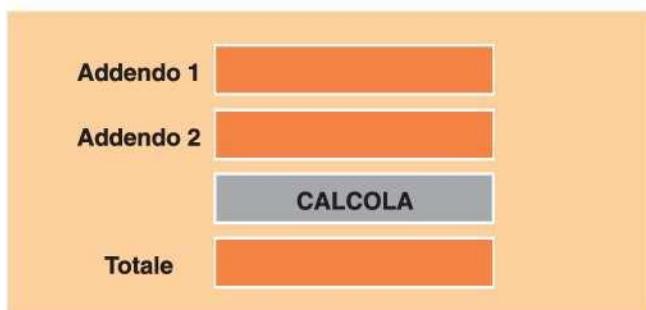
Per creare l'applicazione seguiremo i sei punti elencati in precedenza.

1. Progettare l'applicazione

L'applicazione deve mostrare due caselle in cui l'utente inserisce i numeri da sommare. Deve essere presente un pulsante per attivare il calcolo e un'ulteriore casella per visualizzare il totale.

2. Progettare le finestre

L'applicazione è composta da una finestra schematizzata nel seguente modo:



Facendo clic sul pulsante *Calcola*, si ottiene la somma dei due addendi.

3. Disegnare l'interfaccia

La creazione dell'interfaccia utente prevede i seguenti passi:

- Aprire il programma *NetBeans* e creare un nuovo progetto attraverso la scelta *New* nel menu *File*.
 - Assegnare il nome *CalcolatriceProject* al progetto.
 - Aggiungere una *JFrame* facendo clic con il tasto destro sul nome del progetto e poi su *New, JFrame Form*.
 - Assegnare il nome *SommaFrame* alla classe del *JFrame*.
- A questo punto viene visualizzata la finestra per disegnare l'interfaccia.

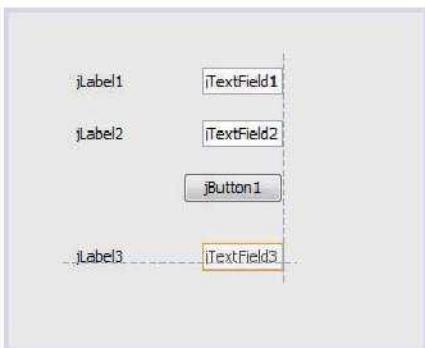
Nella finestra di progettazione aggiungiamo le componenti, selezionandole dalla **Paletta**. Il riquadro *Paletta* compare solitamente nella parte destra della finestra di *NetBeans*. Se non fosse visibile, lo si può selezionare dal menu *Window*. Le seguenti componenti vanno trascinate dal riquadro *Paletta* alla finestra di *Design*:

- due **JLabel** (etichetta) e due **JTextField** (casella di testo) per i dati di input,
- un **JButton** (pulsante) per il comando di azione,
- una **JLabel** e un **JTextField** per il dato di output.



Le componenti *JLabel*, *JTextField* e *JButton* vengono visualizzate sulla finestra di *Design* con nomi predefiniti seguiti da un numero progressivo. Facendo clic su una componente, vengono mostrati, sui bordi dell'oggetto, i quadratini bianchi per il ridimensionamento dell'elemento grafico con il mouse.

Trascinando l'oggetto si può modificarne la posizione e, durante questa operazione, *NetBeans* facilita il loro posizionamento suggerendo l'allineamento migliore per la componente tramite linee tratteggiate.



4. Definire le proprietà delle componenti

Molte proprietà, per esempio la posizione all'interno del *JFrame* o le dimensioni della componente, possono essere direttamente modificate agendo sull'oggetto con il mouse. Le altre proprietà vengono visualizzate nella finestra **Properties**, automaticamente visualizzata non appena si seleziona una componente grafica. Se non è visibile nella finestra di *NetBeans*, il riquadro *Properties* si può selezionare dal menu *Window*.

Il testo visualizzato nell'etichetta *jLabel1* può essere modificato con i seguenti passi:

- selezionare l'etichetta dalla finestra di *Design*,
- selezionare la proprietà **Text** nel riquadro *Properties*,
- impostare il valore *Addendo1*.



Ogni controllo posizionato nella finestra di *Design* rappresenta un oggetto, inteso come istanza della classe della relativa componente grafica. Per poterlo utilizzare all'interno del codice Java, occorre attribuire ad esso un nome impostando la proprietà **Variable Name**, che si trova nel riquadro *Properties* all'interno della linghetta *Code*.

Soltanamente vengono assegnati nomi con un prefisso che permette di individuare il tipo di oggetto (per esempio **lbl** per le *JLabel*, **btn** per i *JButton* e **txt** per i *JTextField*) e una o più parole di senso compiuto che indicano la funzione (per esempio *lblAddendo1* e *btnCalcola*).

La seguente tabella riassume le proprietà impostate per le componenti dell'interfaccia grafica.

Component	Variable Name	Text
JLabel	lblAddendo1	Addendo1
JLabel	lblAddendo2	Addendo2
JLabel	lblTotale	Totale
JButton	btnCalcola	Calcola
JTextField	txtAddendo1	
JTextField	txtAddendo2	
JTextField	txtTotale	

Oltre alle proprietà *Text* e *Variable Name*, le altre principali proprietà degli oggetti precedenti sono:

background	Imposta il colore di sfondo dell'oggetto.
editable	Determina se la casella di testo può essere modificabile.
font	Specifica le caratteristiche del font utilizzato per i caratteri.
foreground	Imposta il colore di primo piano utilizzato per il testo dell'oggetto.
tooltip	Indica il testo da visualizzare quando l'utente si sposta con il mouse sopra l'oggetto.
border	Imposta lo stile del bordo dell'oggetto.
enabled	Determina se il controllo è attivato, cioè se è in grado di rispondere agli eventi generati dall'utente. Per esempio, se un pulsante non è attivo resta visibile, ma assume un aspetto più sfumato e l'utente non può fare clic.
Horizontal Size	Determina la larghezza dell'oggetto in pixel.
Vertical Size	Determina l'altezza dell'oggetto in pixel.

5. Gestire gli eventi

L'evento che si vuole gestire è il clic sul pulsante *btnCalcola*. A seguito del clic, la somma dei due addendi deve essere assegnata alla proprietà *Text* della componente *txtTotale*. Per fare questo occorre fare doppio clic sul pulsante *btnCalcola* nella finestra di *Design*. In alternativa si può fare clic con il tasto destro, sempre sul pulsante *btnCalcola*, e poi, nel menu di scelta rapida, selezionare *Events*, *Action* e *actionPerformed*.

Si apre la finestra del codice e viene creato automaticamente un metodo *btnCalcolaActionPerformed* che viene eseguito quando l'utente fa clic sul pulsante, cioè quando si attiva l'evento **actionPerformed**.

L'intestazione e i parametri del metodo vengono scritti automaticamente.

Il programmatore deve scrivere, tra parentesi graffe, le istruzioni per calcolare la somma e inserirla nella casella del totale:

```
private void btnCalcolaActionPerformed(java.awt.event.ActionEvent evt) {
    double num1, num2, totale;
```

```

num1 = Double.parseDouble(txtAddendo1.getText());
num2 = Double.parseDouble(txtAddendo2.getText());

totale = num1+num2;

txtTotale.setText(" " + totale);
}

```

Si noti che la variabile numerica *totale*, per essere visualizzata nella casella di testo, viene trasformata in stringa tramite concatenazione con la stringa vuota ("").

A questo punto occorre salvare il progetto: nel menu **File**, scegliere **Save All** oppure fare clic sull'icona **Save All**

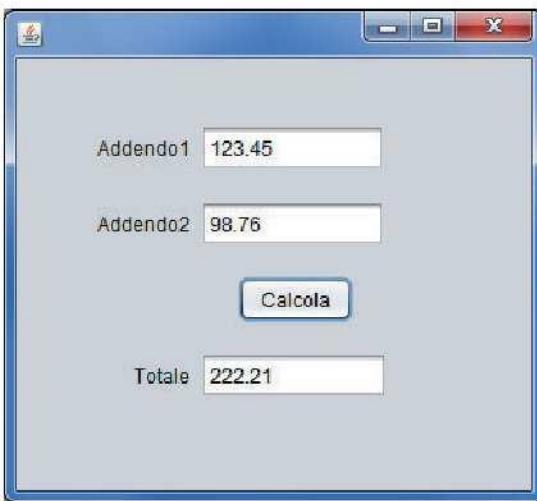


nella Barra degli strumenti.

6. Effettuare il test dell'applicazione

Premere l'icona per avviare l'esecuzione dell'applicazione. Se il programma non contiene errori si apre il *JFrame* come una normale finestra di Windows.

Per il debug e l'esecuzione del programma in ambiente *NetBeans*, vale quanto già visto nell'inserto dopo il Capitolo 3.



Nei paragrafi seguenti vengono presentate le componenti grafiche principali del package *javax.swing*. I progetti e i frammenti di codice riportati illustrano come si possono costruire le interfacce utente tramite il codice Java, a partire dalle classi della libreria. Avendo chiari questi aspetti, anche la programmazione visuale in *NetBeans* e il codice generato automaticamente risulteranno più comprensibili.

AUTOVERIFICA



Problemi da 5 a 10 pag. 347



2. La palette delle componenti Swing e AWT

3. Il codice generato automaticamente da NetBeans

7 Etichette e pulsanti

L'**etichetta** è rappresentata dalla classe **JLabel** e può contenere solo una riga di testo. Il contenuto di un'etichetta viene normalmente stabilito durante la progettazione dell'interfaccia e può essere modificato solo dal programma, ma non dall'utente.

Un'etichetta può essere definita attraverso tre diversi costruttori:

- **JLabel()**

costruttore senza parametri che crea un'etichetta vuota.

- **JLabel(String)**

crea un'etichetta contenente come testo la stringa passata tramite il parametro; la stringa viene allineata a sinistra all'interno dell'etichetta.

- **JLabel(String, int)**

il secondo parametro di tipo intero indica la modalità di allineamento della stringa, usando come valori le **costanti** statiche definite all'interno della classe *JLabel* e precisamente: *JLabel.LEFT* per allineare la stringa a sinistra, *JLabel.RIGHT* per l'allineamento a destra e *JLabel.CENTER* per l'allineamento al centro.

Per esempio:

```
JLabel lblVuota = new JLabel();
JLabel lblNome = new JLabel("Mario");
JLabel lblCompleta = new JLabel("Risultato del calcolo", JLabel.CENTER);
```

Per aggiungere queste etichette al pannello *p* si deve usare il metodo **add**:

```
p.add(lblVuota);
p.add(lblNome);
p.add(lblCompleta);
```

Se non si vuole creare esplicitamente l'oggetto, perché non deve essere successivamente manipolato, si può usare anche un modo più diretto per aggiungere una componente. Per esempio:

```
p.add(new JLabel("..."));
```

Dopo aver creato un'etichetta, si possono invocare i suoi metodi.

Il metodo **setText(String)** consente di modificare il testo dell'etichetta.

Il metodo **setForeground(Color)** modifica il colore del testo, mentre il metodo **setBackground(Color)** modifica il colore di sfondo dell'etichetta.

I seguenti esempi usano i metodi per modificare le caratteristiche di un'etichetta:

```
lblNome.setText("-- J A V A --");
lblNome.setForeground(Color.red);
lblNome.setBackground(Color.white);
```

Gli ultimi due metodi ricevono come parametro un oggetto della classe **Color**. In questa classe sono presenti alcune costanti statiche che possono essere usate per indicare un particolare colore.

Esempi di **costanti** per il colore sono:

Color.black	Color.magenta
Color.blue	Color.orange
Color.cyan	Color.pink
Color.darkGray	Color.red
Color.gray	Color.white
Color.green	Color.yellow
Color.lightGray	

La classe *JLabel*, come tutte le classi che rappresentano le componenti, oltre ai propri metodi eredita tutti quelli della classe *JComponent* e della classe *Object*. Tra i metodi visti, *setText* è un metodo proprio della classe *JLabel*, mentre *setBackground* e *setForeground* sono ereditati dalla classe *JComponent*.

I **pulsanti** vengono creati usando la classe **JButton**. Solitamente contengono una stringa e sono usati per invocare un'azione quando vengono premuti dall'utente. Il costruttore dei pulsanti può non avere parametri oppure contiene la stringa che viene visualizzata sopra il pulsante.

Per esempio:

```
 JButton btnOk = new JButton("OK");
```

Il modo con cui gestire l'evento generato della pressione del pulsante sarà spiegato in un paragrafo successivo dedicato alla gestione degli eventi.

Ogni pulsante creato è attivo, cioè può essere cliccato. È possibile disattivare un pulsante richiamando il metodo **setEnabled(boolean)**: esso riceve come parametro il valore booleano *false* per disattivare il pulsante oppure il valore *true* per attivarlo.

Il seguente esempio rende il pulsante *OK* non cliccabile:

```
btnOk.setEnabled(false);
```

8 Caselle e aree di testo

Le **caselle di testo** sono realizzate dalla classe **JTextField**. Questa classe è una sottoclasse della classe *JTextComponent*, la quale ha come sottoclasse anche la classe *JTextArea*. Vedremo come entrambe le sottoclassi usino i metodi che vengono ereditati dalla classe *JTextComponent*.

La classe **JTextField** crea una casella di testo composta da una sola riga che può essere usata per l'input o l'output di stringhe. I costruttori di questa classe consentono di impostare la stringa da visualizzare e il numero di colonne della casella, cioè la sua dimensione.

Esistono quattro costruttori che permettono di impostare i due parametri in diversi modi:

- **JTextField()**

crea una casella senza specificare il contenuto e la dimensione.

- **JTextField(String)**

crea una casella inizializzata con un certo valore e con dimensione uguale alla lunghezza della stringa.

- **JTextField(int)**

crea una casella vuota di dimensione specificata dal parametro; una casella creata in questo modo è usata per permettere all'utente di inserire i valori.

- **JTextField(String, int)**

consente di impostare sia il contenuto della casella che la sua dimensione espressa dal parametro intero.

PROGETTO 3

Creare una maschera di input per inserire il nome di un oggetto e il suo prezzo.

Nel programma sono state inserite le etichette per indicare all'utente i tipi di dato che devono essere inseriti nelle caselle di testo. Sia le etichette che le caselle di testo vengono aggiunte a un pannello, che a sua volta viene aggiunto alla finestra del programma.

PROGRAMMA JAVA (*Input.java*)

```
import javax.swing.*;
import java.awt.*;

class Input
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame("Input");
        JPanel p = new JPanel();

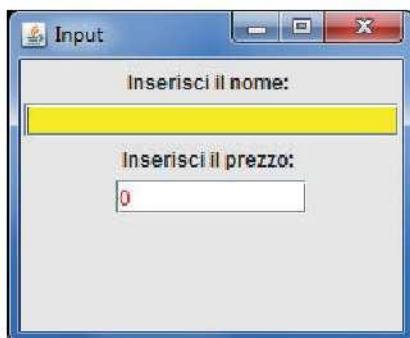
        JTextField nome = new JTextField(20);
        JTextField prezzo = new JTextField("0", 10);

        p.add(new JLabel("Inserisci il nome: ", JLabel.RIGHT));
        nome.setBackground(Color.yellow);
        p.add(nome);
        p.add(new JLabel("Inserisci il prezzo: ", JLabel.RIGHT));
        prezzo.setForeground(Color.red);
        p.add(prezzo);

        f.getContentPane().add(p);

        f.setSize(200, 200);
        f.setLocation(100,100);

        f.setVisible(true);
    }
}
```



La classe *JTextField* eredita tre metodi molto utili dalla classe *JTextComponent*. Questi metodi potranno essere usati anche con la classe *JTextArea*.

Il metodo **setText(String)** consente, dopo che la casella di testo è stata creata, di modificarne il contenuto.

Per esempio si può modificare l'oggetto *prezzo* del progetto precedente nel seguente modo:

```
prezzo.setText("0.00");
```

Il metodo **getText()** consente di leggere il contenuto di una casella di testo. Non ha parametri e restituisce un oggetto di classe *String*. Questo metodo è usato per leggere il valore inserito dall'utente. La lettura dell'input dalla casella di testo sarà spiegata successivamente all'interno della gestione degli eventi.

Il metodo **setEditable(boolean)** rende una casella di testo editabile o non editabile.

Una casella è editabile se l'utente, posizionandosi con il mouse, può digitare dei caratteri. Usando il metodo *setEditable*, si può decidere se permettere all'utente la modifica del contenuto della casella (valore *true*) oppure se impedirne le modifiche (valore *false*).

Una casella non editabile è simile a un'etichetta e può essere usata dal programma per mostrare i messaggi inviati all'utente o i risultati di un'elaborazione.

Per rendere non modificabile la casella *prezzo*, bisogna invocare il metodo *setEditable* nel seguente modo:

```
prezzo.setEditable(false);
```

La classe **JTextArea** consente di creare un'area di testo formata da più righe. I costruttori sono simili a quelli della casella di testo con l'aggiunta di un parametro che indica il numero di righe. I costruttori della classe *JTextArea* consentono di impostare la stringa da visualizzare al suo interno, la dimensione dell'area di testo espressa in numero di righe e numero di colonne e la visualizzazione delle barre di scorrimento. Esistono quattro costruttori che permettono di impostare i parametri in vari modi:

- **JTextArea()**

crea un'area di testo vuota.

- **JTextArea(int, int)**

crea un'area di testo vuota, avente come numero di righe e di colonne i due parametri interi.

- **JTextArea(String)**

crea un'area di testo con uno specifico valore di testo.

- **JTextArea(String, int, int, int)**

crea un'area di testo con uno specifico valore di testo, avente come numero di righe e di colonne i due parametri interi; il quarto parametro è una **costante** intera che specifica la visualizzazione delle barre di scorrimento: *SCROLLBARS_BOTH* (valore di *default*), *SCROLLBARS_VERTICAL_ONLY*, *SCROLLBARS_HORIZONTAL_ONLY*, *SCROLLBARS_NONE*.

Per esempio, la seguente riga di codice definisce un'area di testo composta da 10 righe e larga 30 caratteri:

```
JTextArea messaggi = new JTextArea(10, 30);
```

Agli oggetti di questa classe si possono applicare gli stessi metodi usati per *JTextField*.

Con il metodo **setText(String)** si modifica il contenuto dell'area di testo: viene cancellato il testo precedente e viene aggiunto il nuovo testo.

Con il metodo **getText()** si legge tutto il contenuto dell'area di testo.

Si può rendere modificabile o meno l'area usando il metodo **setEditable(boolean)**.

Le aree di testo hanno un metodo aggiuntivo **append(String)** per inserire nuovo testo alla fine dell'area senza cancellare il contenuto già esistente.

PROGETTO 4

Aggiungere un'area di testo di 10 righe e 30 colonne a una finestra.

PROGRAMMA JAVA (*Area.java*)

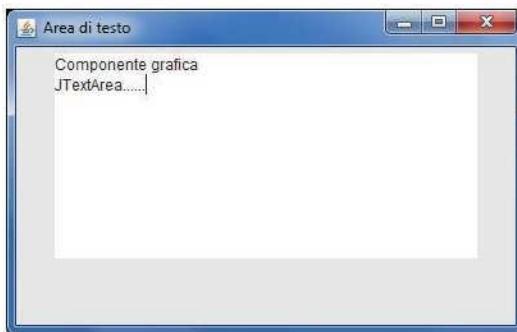
```
import javax.swing.*;
import java.awt.*;

class Area
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame("Area di testo");
        JPanel p = new JPanel();

        JTextArea messaggi = new JTextArea(10, 30);
        p.add(messaggi);

        f.getContentPane().add(p);
        f.setSize(500, 300);
        f.setLocation(100, 0);

        f.setVisible(true);
    }
}
```



9 Caselle combinate e caselle di controllo

Le **caselle combinate** (*combo box*) raggruppano un elenco di voci e consentono, tramite un menu a tendina, la scelta di una singola voce. Questo elemento grafico viene implementato in Java usando la classe **JComboBox**. La costruzione di una combo box avviene in due fasi:

- creazione dell'oggetto di classe *JComboBox*,
- aggiunta delle voci all'oggetto.

La creazione viene eseguita attraverso un costruttore che non possiede parametri. Successivamente le voci vengono aggiunte, una ad una, usando il metodo **addItem(String)**. Il parametro di questo metodo specifica il testo che deve essere visualizzato all'interno della combo box.

Il seguente frammento di codice crea una casella combinata che contiene l'elenco di quattro colori:

```
JComboBox colori = new JComboBox();
colori.addItem("rosso");
colori.addItem("verde");
colori.addItem("bianco");
colori.addItem("arancione");
```



Si può aggiungere questa componente a una finestra con il metodo **add**. L'utente può modificare il valore visualizzato in una combo box agendo con il mouse sul corrispondente menu a tendina.

La **casella di controllo** (*check box*) è un elemento grafico che gestisce solo due stati, corrispondenti ai valori booleani *true* e *false*. Se la casella ha il segno di spunta, ad essa si associa il valore booleano *true*, altrimenti si associa il valore *false*.

Questo elemento grafico viene implementato in Java con la classe **JCheckBox** che mette a disposizione i pulsanti a due stati; se l'utente fa un clic sulla casella, ne causa il cambio di stato. Ogni casella di controllo è fornita di un'etichetta che viene usata per spiegare all'utente il significato della casella stessa.

La seguente riga di codice crea una casella di controllo come oggetto di classe *JCheckBox*:

```
JCheckBox c = new JCheckBox("prima scelta");
```

Per *default*, le caselle di controllo sono disattivate e il loro valore è impostato a *false*. Una casella può essere impostata a *true* usando il costruttore che contiene, come secondo parametro, il valore booleano da attribuire ad essa: **JCheckBox(String, boolean)**.

PROGETTO 5

Presentare un menu di piatti per la scelta dell'utente.

Nel programma viene utilizzata la classe **GridLayout** per facilitare la disposizione (*layout*) delle componenti in modo ordinato all'interno di una griglia (*Grid*).

In particolare, la seguente istruzione imposta per il pannello *p* un layout con una griglia formata da 6 righe e 1 colonna:

```
p.setLayout(new GridLayout(6,1));
```

La spiegazione in dettaglio di questo e di altri *layout* sarà illustrata alla fine di questo paragrafo.

PROGRAMMA JAVA (*Scelte.java*)

```

import java.awt.*;
import javax.swing.*;

class Scelte
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame();
        JPanel p = new JPanel();
        JCheckBox c1 = new JCheckBox("antipasto", true);
        JCheckBox c2 = new JCheckBox("primo piatto");
        JCheckBox c3 = new JCheckBox("secondo piatto", true);
        JCheckBox c4 = new JCheckBox("contorno");
        JCheckBox c5 = new JCheckBox("dessert");

        // dispone gli elementi in una griglia
        p.setLayout(new GridLayout(6,1));

        p.add(new Label("Scegli il tuo pranzo"));
        p.add(c1);
        p.add(c2);
        p.add(c3);
        p.add(c4);
        p.add(c5);

        f.getContentPane().add(p);

        // esce dal programma quando la finestra viene chiusa
        f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        f.setTitle("Scelta menu");
        f.setSize(250, 200);
        f.setVisible(true);
    }
}

```



Si osservi che il codice contiene l'istruzione che termina il programma quando l'utente fa clic sul pulsante di chiusura della finestra:

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



LAYOUT DEGLI ELEMENTI GRAFICI

In Java, il modo con cui le componenti vengono disposte e posizionate (**layout**) dipende dal contenitore. A ogni contenitore può essere associato un gestore che si preoccupa della disposizione degli elementi. Questo gestore è chiamato **Layout Manager**. Tutti i contenitori hanno un *Layout Manager* di *default*, che può essere comunque cambiato usando un metodo e definendo il nuovo gestore. Il gestore interviene la prima volta che viene visualizzato un contenitore, per disporre le componenti, e successivamente tutte le volte che ne vengono modificate le dimensioni, per riposizionare le componenti. Ogni volta che vengono modificate le dimensioni di una finestra, le componenti vengono riadattate, ingrandendole o riducendole, ma mantenendo lo schema definito dal gestore che è stato scelto. Java permette di utilizzare diversi gestori. Ognuno dispone gli elementi in un modo particolare e richiede l'uso di un metodo **add** adeguato per aggiungerne di nuovi.

Per assegnare un *Layout Manager* a un contenitore si deve utilizzare il metodo **setLayout**, che riceve come parametro un oggetto che rappresenta il gestore.

Per assegnare a un Frame *f* il gestore **FlowLayout** si usa la seguente istruzione:

```
f.setLayout(new FlowLayout());
```

Tra parentesi viene creato un nuovo oggetto che rappresenta il gestore e viene passato al metodo *setLayout*. L'istruzione precedente è un'abbreviazione che consente di creare l'oggetto e di passarlo contemporaneamente come parametro. Si evita così di creare esplicitamente un oggetto *FlowLayout*. La precedente istruzione corrisponde alle seguenti:

```
FlowLayout gestore = new FlowLayout();  
f.setLayout(gestore);
```

Osservando le istruzioni precedenti si vede che il gestore è un oggetto della classe *FlowLayout*.

Di seguito vengono presentati alcuni dei gestori che si possono usare in Java, specificando sia le modalità con cui dispongono gli elementi grafici, sia il metodo *add* da utilizzare per aggiungere le componenti.

• **FlowLayout**

La classe *FlowLayout* viene usata per disporre le componenti su una stessa riga, una dopo l'altra, nello stesso ordine con cui vengono aggiunte. Quando la riga viene completata, si passa alla riga successiva. Le componenti aggiunte vengono centrate nella riga e sono separate con uno spazio di 5 pixel.

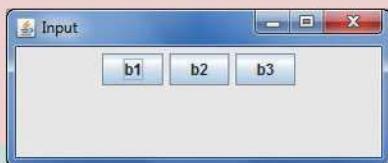
Questo gestore è associato per *default* a tutti i pannelli.

Il metodo **add** che viene usato per aggiungere le componenti a un contenitore di tipo *FlowLayout*, riceve come unico parametro la componente da inserire.

Per esempio, le seguenti istruzioni creano un pannello formato da tre bottoni:

```
JPanel p = new JPanel();  
JButton b1 = new JButton("b1");  
JButton b2 = new JButton("b2");  
JButton b3 = new JButton("b3");
```

```
p.setLayout(new FlowLayout());
p.add(b1);
p.add(b2);
p.add(b3);
```



Nella prima parte vengono creati il contenitore *p* e i pulsanti. Viene poi impostato il tipo di *Layout Manager* del pannello e vengono aggiunti, uno ad uno, i pulsanti.

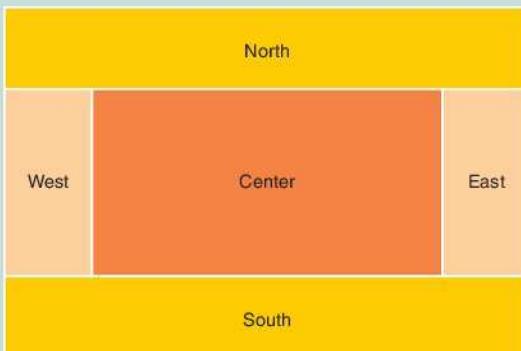
Il costruttore può anche impostare per il *FlowLayout* l'allineamento delle componenti attraverso le **costanti** statiche: *CENTER*, *RIGHT* e *LEFT*.

Per allineare a sinistra le componenti del precedente pannello si deve impostare il layout nel seguente modo:

```
p.setLayout(new FlowLayout(FlowLayout.LEFT));
```

• BorderLayout

Usando la classe *BorderLayout*, il contenitore viene diviso in cinque regioni, in ognuna delle quali può essere inserita una componente. Le regioni sono indicate con i seguenti nomi: *North*, *South*, *Center*, *East*, *West*, secondo la disposizione mostrata nella figura seguente.



La zona centrale tende a occupare più spazio possibile riducendo le altre al minimo indispensabile per visualizzare il loro contenuto. Si possono aggiungere le componenti solo ad alcune zone, lasciandone altre vuote. In questo caso il loro spazio sarà utilizzato dalle componenti presenti. Il *BorderLayout* è associato per *default* a tutte le finestre.

Se si vuole applicare questo gestore ai pannelli, occorre usare il metodo *setLayout* nel seguente modo:

```
JPanel p = new JPanel();
p.setLayout(new BorderLayout());
```

Per aggiungere gli elementi al contenitore si usa un metodo **add** diverso da quello utilizzato con il *FlowLayout*.

Il metodo *add* per il *BorderLayout* riceve due parametri, di cui il primo indica la componente che deve essere aggiunta, mentre il secondo indica la zona in cui va aggiunta. La zona è indicata usando una stringa che ne specifica il nome.

Per esempio, l'aggiunta di un bottone *b* nella parte inferiore di una finestra *f* che usa il *BorderLayout* viene eseguita richiamando il metodo *add* nel seguente modo:

```
f.add(b, "South");
```

Questo tipo di disposizione viene usato quando c'è una componente che deve occupare molto spazio, per esempio un'area di testo o un'area di disegno, e che quindi può essere opportunamente aggiunta al contenitore nella zona centrale. Le zone laterali vengono usate per inserire i pannelli che solitamente contengono i comandi dell'applicazione.

• **GridLayout**

La classe *GridLayout* permette di posizionare le componenti in una griglia, come visto nel progetto precedente. Il contenitore viene diviso in tante celle, tutte della stessa dimensione, come una tabella, organizzata in righe e colonne: ogni componente può essere inserita in una cella.

Esistono due costruttori per questo gestore che differiscono per il numero di parametri che ricevono.

GridLayout(int, int)

consente di specificare il numero di righe e il numero di colonne che formano la griglia. Per esempio:

GridLayout(5,3) definisce una griglia con 5 righe e 3 colonne.

GridLayout(int, int, int, int)

oltre alle dimensioni della griglia, permette di indicare quanto spazio lasciare tra le componenti. Ha quattro parametri: i primi due hanno lo stesso significato del costruttore precedente; il terzo parametro indica quanti pixel separano le componenti orizzontalmente, mentre il quarto parametro indica la spaziatura verticale.

Le componenti sono aggiunte usando un metodo **add** che possiede come unico parametro la componente. Le celle vengono riempite usando le componenti nell'ordine con cui vengono inserite. Prima si riempie la prima riga, da sinistra a destra e poi si passa alle celle sulla seconda riga e così via.

PROGETTO 6

Predisporre un'interfaccia grafica per l'inserimento di un nome, di un cognome e di un numero di telefono.

Questa interfaccia è realizzata usando una tabella avente nella prima colonna le etichette e nella seconda i campi di testo. Essendo tre i valori di input, la griglia è composta da 3 righe e 2 colonne.

PROGRAMMA JAVA (*Tabella.java*)

```
import javax.swing.*;
import java.awt.*;

class Tabella
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame("Tabella");
        JPanel p = new JPanel();
```

```

JTextField txtNome      = new JTextField(30);
JTextField txtCognome  = new JTextField(30);
JTextField txtTelefono = new JTextField(30);

p.setLayout(new GridLayout(3,2,10,10));
p.add(new JLabel("Nome", JLabel.RIGHT));
p.add(txtNome);
p.add(new JLabel("Cognome", JLabel.RIGHT));
p.add(txtCognome);
p.add(new JLabel("Telefono", JLabel.RIGHT));
p.add(txtTelefono);

f.getContentPane().add(p);
f.setSize(200,120);
f.setLocation(100,100);
f.setVisible(true);
}
}

```



AUTOVERIFICA

Problemi da 11 a 14 pag. 347-348



4. La componente per gestire un'area di disegno

5. I contenitori ScrollPane e TabbedPane

10 Gestione degli eventi

I due aspetti principali dell'interazione con l'utente sono:

- riconoscere quando l'utente compie le azioni
- predisporre le operazioni da eseguire in corrispondenza delle azioni compiute dall'utente.

Il primo è un compito affidato al sistema che gestisce gli eventi. Il secondo è affidato al programmatore che, in base all'applicazione, scrive il codice da eseguire per ogni possibile azione dell'utente. Queste due fasi sono naturalmente collegate: durante l'esecuzione, il gestore degli eventi, in base all'evento che rileva, manda in esecuzione il codice appropriato.

L'obiettivo di chi gestisce un'interfaccia utente è la rapidità di risposta alle azioni dell'utente: nel linguaggio Java gli eventi vengono gestiti solo da chi si registra come **gestore**. Tutti gli eventi che non hanno un gestore vengono ignorati, mentre se un evento si accorge di avere un gestore, ne affida ad esso la gestione.

Questo modello viene chiamato **a delegazione**, perché il programmatore stabilisce, con una delega, chi può essere abilitato a gestire un evento.

Gli eventi sono generati a partire da un particolare oggetto, chiamato **origine**. Per esempio, un pulsante genera un evento quando si fa clic su di esso, una finestra genera un evento quando viene chiusa. Tutte le componenti sono possibili oggetti di origine.

Uno o più **ascoltatori** (*listener*) possono registrarsi nell'oggetto origine per essere avvisati della generazione di un particolare tipo di evento. Gli ascoltatori sono in pratica i gestori degli eventi. L'origine, in presenza di un evento, avverte solo gli ascoltatori che si sono registrati.

Un ascoltatore è un oggetto la cui classe è costruita in un modo particolare. Deve implementare un'interfaccia di tipo **listener**. Un'interfaccia è una classe astratta in cui tutti i metodi sono astratti.

Una classe che implementa un'interfaccia, eredita dall'interfaccia tutti i metodi astratti e, per ognuno, deve fornire un'implementazione.

Per realizzare la gestione degli eventi si deve procedere nel seguente modo:

- 1) Creare uno o più ascoltatori in base agli eventi che si vogliono gestire.
- 2) Registrare l'ascoltatore in un oggetto origine che si vuole controllare.
- 3) Gestire l'evento eseguendo il metodo associato.

1. Creazione degli ascoltatori

Esistono diversi tipi di ascoltatori, ognuno rappresentato da un'interfaccia: l'interfaccia per creare l'ascoltatore delle finestre, dei pulsanti, del mouse e della tastiera. Una classe diventa ascoltatore se implementa un'interfaccia di tipo *listener*.

Per esempio:

```
import java.awt.event.*;

class GestoreFinestra implements WindowListener
{
}
```

La parola chiave **implements** specifica che la classe implementa una particolare interfaccia. Il funzionamento è simile all'ereditarietà. In questo caso l'interfaccia **WindowListener** contiene i metodi astratti che vengono ereditati dalla classe *GestoreFinestra*. Questa, non essendo astratta, deve fornire un'implementazione di tutti i metodi ereditati.

Le interfacce che gestiscono gli eventi sono contenute nel package **java.awt.event** che quindi deve essere importato.

Per completare la creazione dell'ascoltatore bisogna conoscere quali sono i metodi astratti da implementare. In questo caso, l'ascoltatore per la finestra deve possedere i seguenti metodi:

```
public void windowIconified(WindowEvent e) {}
public void windowDeiconified(WindowEvent e) {}
public void windowActivated(WindowEvent e) {}
public void windowDeactivated(WindowEvent e) {}
public void windowOpened(WindowEvent e) {}
public void windowClosed(WindowEvent e) {}
public void windowClosing(WindowEvent e) {}
```

È obbligatorio che tutti i metodi di un ascoltatore siano presenti anche se non contengono alcuna istruzione. All'interno di questi metodi vanno inserite le istruzioni che devono essere eseguite in risposta a un particolare evento. Ogni metodo degli ascoltatori riceve come parametro l'evento che è stato generato.

Per esempio, se il programma deve terminare quando la finestra viene chiusa, si deve sovrscrivere il metodo *windowClosing* nel seguente modo.

```
public void windowClosing(WindowEvent e)
{
    System.exit(0);
}
```

2. Registrazione presso l'origine

Dopo aver predisposto gli ascoltatori, si devono scegliere gli oggetti origine. La scelta ricade sugli oggetti che possono generare un evento e ai quali si vuole associare una risposta. Un oggetto origine è una componente, per esempio una finestra o un pulsante.

Ogni componente possiede i metodi per registrare un particolare ascoltatore.

Il metodo **addWindowListener** è usato dalle finestre per registrare gli ascoltatori di tipo *WindowListener*, che gestiscono gli eventi legati alle finestre.

Il metodo **addActionListener** è usato dai pulsanti per registrare gli ascoltatori che gestiscono gli eventi sui pulsanti.

Supponiamo di aver dichiarato la finestra *f* nel seguente modo:

```
Frame f = new Frame();
```

Se utilizziamo un ascoltatore di classe *GestoreFinestra*, la sua registrazione avviene invocando il metodo *addWindowListener* sull'oggetto finestra:

```
f.addWindowListener(new GestoreFinestra());
```

3. Modalità di esecuzione

Dopo aver spiegato il modo con cui creare un gestore, vediamo in pratica quello che si verifica quando viene generato un evento.

L'utente, interagendo con il programma, può compiere le azioni che generano gli eventi. Ogni evento nasce da una particolare componente, chiamata *origine*. A seguito della nascita di un evento, l'origine cerca un *ascoltatore*, tra quelli che si sono registrati, in grado di gestire quel tipo di evento. Se non ci sono ascoltatori, l'evento viene abbandonato. Se invece l'origine trova un ascoltatore, invoca il metodo associato all'evento.

Dopo aver completato la gestione di un evento, il sistema passa all'evento successivo tra quelli in coda, oppure resta in attesa.

Il seguente progetto mostra come gestire l'evento di **chiusura di una finestra**.

PROGETTO 7

Creare una finestra e registrare il gestore per gli eventi legati alla finestra.

Nell'esecuzione dei programmi presentati in precedenza, le applicazioni grafiche dovevano essere chiuse forzosamente usando la combinazione di tasti *Ctrl + C*. Con la gestione degli eventi si può trattare questa situazione in modo più corretto.

L'evento da gestire è la chiusura di una finestra: quando l'utente preme il tasto di chiusura, genera un evento che deve essere gestito per terminare l'applicazione.

Il *GestoreFinestra* è impostato per stampare un messaggio e poi terminare il programma.

IMPLEMENTAZIONE DELLA CLASSE (*GestoreFinestra.java*)

```
import java.awt.event.*;

class GestoreFinestra implements WindowListener
{
    public void windowIconified(WindowEvent e) {}
    public void windowDeiconified(WindowEvent e) {}
    public void windowActivated(WindowEvent e) {}
    public void windowDeactivated(WindowEvent e) {}
    public void windowOpened(WindowEvent e) {}
    public void windowClosed(WindowEvent e) {}
    public void windowClosing(WindowEvent e)
    {
        System.out.println("Programma terminato.");
        System.exit(0);
    }
}
```

PROGRAMMA JAVA (*Vuota.java*)

```
import javax.swing.*;
import java.awt.*;

class Vuota
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame("Finestra vuota");

        f.setSize(200,200);
        f.addWindowListener(new GestoreFinestra());

        f.setVisible(true);
    }
}
```

L'interfaccia *WindowListener* contiene i metodi che vengono attivati a seguito di particolari eventi. Nel progetto precedente ci sono metodi che non contengono alcuna istruzione. Nella seguente tabella sono indicati, per ogni evento, i metodi che vengono eseguiti.

Evento	Metodo eseguito
Quando la finestra viene ridotta ad icona	WindowIconified
Quando la finestra viene ingrandita	WindowDeiconified
Quando la finestra riceve lo stato attivo	WindowActivated
Quando la finestra perde lo stato attivo	WindowDeactivated
Quando la finestra viene aperta	WindowOpened
Quando la finestra è stata chiusa	WindowClosed
Quando la finestra si sta chiudendo	WindowClosing

Il seguente progetto mostra come gestire l'evento di **clic su un pulsante**.

PROGETTO 8

Creare un'interfaccia con due pulsanti e un'area di testo, gestendo gli eventi della finestra e dei pulsanti.

Disegno dell'interfaccia grafica

La finestra è organizzata con un *BorderLayout*. Nella zona in alto e in basso vengono posizionati due pulsanti che rappresentano rispettivamente il pulsante superiore e quello inferiore. Nella zona centrale viene collocata l'area di testo, impostata come non editabile, dentro la quale sono visualizzati i messaggi.

Gestione degli eventi

L'unico evento disponibile per i pulsanti è la pressione su di essi con il tasto del mouse. Un ascoltatore, per questo evento, deve implementare l'interfaccia **ActionListener**. Questa interfaccia contiene un solo metodo che deve essere ridefinito e che si chiama **actionPerformed**.

Quando un pulsante viene premuto, genera un evento. Se è stato registrato un ascoltatore, viene invocato il metodo *actionPerformed* contenuto nell'ascoltatore.

Lo stesso ascoltatore può essere registrato da più pulsanti. Questo significa che viene eseguito lo stesso metodo ogni volta che un pulsante viene premuto. Per discriminare il pulsante e quindi le diverse operazioni da eseguire, all'interno del metodo *actionPerformed* viene eseguito un controllo per stabilire chi ha generato l'evento.

Il metodo *actionPerformed*, riceve come parametro un oggetto di classe **ActionEvent**. Per acquisire l'indicazione dell'oggetto che ha generato l'evento si usa il metodo **getActionCommand()**. Questo metodo restituisce una stringa che rappresenta il testo mostrato dal pulsante. Confrontando questa scritta con i pulsanti che si vogliono gestire, si può stabilire chi ha generato l'evento e quindi eseguire le opportune operazioni.

IMPLEMENTAZIONE DELLA CLASSE (*GestorePulsante.java*)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class GestorePulsante implements ActionListener
{
    private JTextArea a;

    public GestorePulsante(JTextArea a)
    {
        this.a = a;
    }

    public void actionPerformed(ActionEvent e)
    {
        String pulsante = e.getActionCommand();

        if (pulsante.equals("Superiore"))
        {
            a.append("È stato premuto il pulsante *superiore*.\n");
        }
    }
}
```

```

if (pulsante.equals("Inferiore"))
{
    a.append("È stato premuto il pulsante *inferiore*.\n");
}
}
}

```

PROGRAMMA JAVA (*Pulsanti.java*)

```

import javax.swing.*;
import java.awt.*;

class Pulsanti
{
    public static void main(String argv[])
    {
        JFrame f = new JFrame("Pulsanti");
        JPanel p = new JPanel();
        JButton sup = new JButton("Superiore");
        JButton inf = new JButton("Inferiore");
        JTextArea a = new JTextArea(50,10);

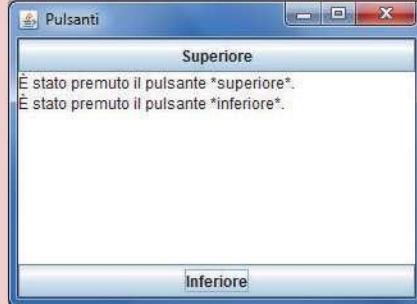
        p.setLayout(new BorderLayout());
        p.add(sup, "North");
        p.add(a, "Center");
        p.add(inf, "South");

        f.addWindowListener(new GestoreFinestra());
        sup.addActionListener(new GestorePulsante (a));
        inf.addActionListener(new GestorePulsante(a));

        a.setEditable(false);

        f.getContentPane().add(p);
        f.setSize(450, 300);
        f.setVisible(true);
    }
}

```



La classe *GestoreBottone* ha un costruttore che riceve come parametro una *JTextArea*. Questa è un riferimento all'area di testo che deve essere gestita dalla pressione dei pulsanti. Essa è passata come parametro al gestore dei pulsanti, perché altrimenti non saprebbe come scrivere all'interno dell'area di testo.

Il messaggio all'interno dell'area di testo viene stampato usando il metodo **append(String)**. Per la gestione della finestra, viene usata la medesima classe *GestoreFinestra* definita nel progetto precedente.

Il seguente progetto mostra come gestire la **lettura da una casella di testo** a seguito di un evento.

PROGETTO 9**Realizzare, con un'interfaccia grafica, un convertitore tra gradi centigradi e fahrenheit.**

La formula di conversione è: fahrenheit = $32 + (\text{centigradi}/100) * 180$.

Disegno dell'interfaccia grafica

La finestra è organizzata con un *GridLayout*: la griglia è composta da tre celle disposte verticalmente. Nella prima si inserisce un pannello per i gradi centigradi contenente un'etichetta e una casella di testo. Nella seconda viene posizionato un pulsante per eseguire la conversione. Nella terza si inserisce un altro pannello per la misura in fahrenheit con un'etichetta e una casella di testo.

La tabella seguente riassume i nomi degli oggetti utilizzati nell'interfaccia grafica con la relativa classe.

Classe	Nome oggetto
JFrame	f
JPanel	p1
JPanel	p2
JTextField	txtCentigradi
JTextField	txtFahrenheit
JButton	btnConverti

Gestione degli eventi

La lettura del contenuto di una casella di testo, come già detto, è realizzata dal metodo **getText()**. La lettura, però, deve essere fatta dopo che l'utente ha inserito il valore. Si può usare un pulsante con il quale l'utente possa confermare l'avvenuto inserimento dei dati. In questo modo la lettura della casella di testo avviene all'interno del gestore del pulsante.

IMPLEMENTAZIONE DELLA CLASSE (*ConvertFrame.java*)

```

import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class ConvertFrame extends JFrame implements ActionListener
{
    private JPanel p1 = new JPanel();
    private JPanel p2 = new JPanel();
    private JTextField txtCentigradi = new JTextField(15);
    private JTextField txtFahrenheit = new JTextField(15);
    private JButton btnConverti = new JButton("Converti");

    public ConvertFrame()
    {
        super("Convertitore Centigradi->Fahrenheit");

        addWindowListener(new GestoreFinestra());
    }
}

```

```

// inserisce le componenti nei pannelli
p1.add(new JLabel("Gradi Centigradi: "));
p1.add(txtCentigradi);
p2.add(new JLabel("Gradi Fahrenheit: "));
p2.add(txtFahrenheit);

// inserisce le componenti nella finestra disponendole con una griglia
setLayout(new GridLayout(3,1,5,10));
add(p1);
add(btnConverti);
add(p2);
btnConverti.addActionListener(this);
}

public void actionPerformed(ActionEvent e)
{
    String pulsante = e.getActionCommand();
    double cent, fahr;

    if (pulsante.equals("Converti"))
    {
        try
        {
            String numeroLetto = txtCentigradi.getText();
            cent = Double.valueOf(numeroLetto).doubleValue();
            fahr = 32+(cent/100)*180;
            txtFahrenheit.setText(""+fahr);
        }
        catch(Exception exc)
        {
            txtFahrenheit.setText("");
        }
    }
}
}

```



PROGRAMMA JAVA (*Convertitore.java*)

```

class Convertitore
{
    public static void main(String argv[])
    {
        ConvertFrame f = new ConvertFrame();

        f.pack();
        f.setVisible(true);
    }
}

```

La classe *Convertitore* è una classe semplice contenente solo il *main*. Il suo compito è creare una finestra di classe *ConvertFrame* e di visualizzarla. Il metodo **pack()** impone che le dimensioni della finestra siano le più piccole possibili, sufficienti a contenere tutte le componenti.

La classe *ConvertFrame* è una sottoclassificazione di un *JFrame* e quindi è una finestra. Contiene come attributi privati tutti gli elementi grafici che vengono usati all'interno della finestra. Il suo costruttore posiziona le componenti e registra il gestore della finestra e quello del pulsante. Essendo una finestra, possono essere usati i metodi ereditati dalla classe *JFrame* senza usare la notazione con il punto. Ne sono un esempio i metodi *addWindowListener*, *setLayout* e *add*. Il gestore del pulsante è implementato dalla stessa classe *ConvertFrame*. Questo viene fatto usando la parola chiave *implements* nella definizione della classe.

Non è quindi necessario che un ascoltatore sia una classe a sé stante, ma può essere inserito in un'altra classe, basta che vengano ridefiniti tutti i metodi di quella interfaccia. In questo caso, l'interfaccia è la *ActionListener* e l'unico metodo che viene ridefinito è *actionPerformed*.

Per registrare questo gestore nella componente *btnConverti*, corrispondente al pulsante, è stata usata la seguente istruzione:

```
btnConverti.addActionListener(this);
```

Questo significa che l'ascoltatore è **this**, cioè l'oggetto stesso che contiene questa istruzione. Raggruppare un ascoltatore all'interno della classe in cui viene usato è utile, perché l'ascoltatore può accedere liberamente a tutte le componenti dell'interfaccia.

Nel programma precedente, l'ascoltatore accede in lettura alla casella di testo *txtCentigradi* e in scrittura alla casella di testo *txtFahrenheit*. Anche in questo programma viene usata la classe *GestoreFinestra* definita nei progetti precedenti.

Il seguente progetto mostra come gestire la **lettura da una combo box** a seguito di un evento.

PROGETTO 10

Offrire all'utente la possibilità, tramite una casella combinata, di modificare il colore di sfondo di un'area.

Disegno dell'interfaccia grafica

La finestra è organizzata con un *BorderLayout*. Nella zona in alto si dispone un pannello contenente un'etichetta, una combo box per la scelta e un pulsante per effettuare il cambio di colore. La zona centrale, la più ampia della finestra, è occupata da un pannello impostato inizialmente con lo sfondo bianco.

La tabella seguente riassume i nomi degli oggetti utilizzati nell'interfaccia grafica con la relativa classe.

Classe	Nome oggetto
JFrame	f
JPanel	p
JPanel	panArea
JComboBox	cbColori
JButton	btnCambia

Gestione degli eventi

Per la lettura da una casella combinata si adotta lo stesso procedimento usato con le caselle di testo. Un utente può cambiare il valore della casella combinata e inviare al programma questo valore facendo clic su un pulsante. Perciò la lettura del valore della combo box è inserito nel gestore del pulsante. Viene utilizzato il metodo **getSelectedIndex()** che restituisce un numero corrispondente all'indice della voce selezionata. Le voci sono numerate in base all'ordine con cui sono state aggiunte alla combo box in fase di creazione. La prima voce ha indice 0, la seconda indice 1 e così via.

La classe *GestoreFinestra* è la stessa utilizzata nei progetti precedenti.

IMPLEMENTAZIONE DELLA CLASSE (*CambiaFrame.java*)

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

class CambiaFrame extends Frame implements ActionListener
{
    private Color sfondo      = Color.white;
    private JPanel p           = new JPanel();
    private JPanel panArea     = new JPanel ();
    private JComboBox cbColori = new JComboBox();
    private JButton btnCambia = new JButton("Cambia");

    public CambiaFrame()
    {
        super("Cambia sfondo!");
        setSize(400,250);
        addWindowListener(new GestoreFinestra());
        inizializzaCombo();
        panArea.setBackground(sfondo);
        // inserisce le componenti nel pannello in alto
        p.add(new Label("Colore di sfondo: "));
        p.add(cbColori);
        p.add(btnCambia);

        // inserisce le componenti nella finestra
        add(p, "North");
        add(panArea, "Center");
        btnCambia.addActionListener(this);
    }

    private void inizializzaCombo()
    {
        // aggiunge le voci alla combo-box
        cbColori.addItem("bianco");
        cbColori.addItem("rosso");
        cbColori.addItem("arancione");
        cbColori.addItem("giallo");
        cbColori.addItem("verde");
        cbColori.addItem("blu");
        cbColori.addItem("nero");
    }
}
```

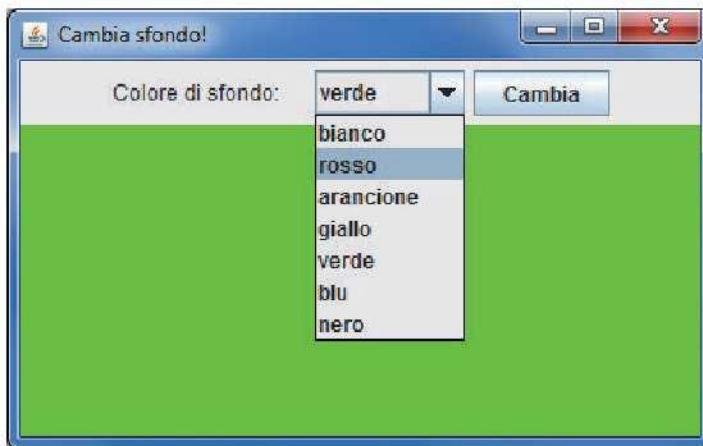
```

public void actionPerformed(ActionEvent e)
{
    String pulsante = e.getActionCommand();

    if (pulsante.equals("Cambia"))
    {
        switch (cbColori.getSelectedIndex())
        {
            case 0: sfondo = Color.white; break;
            case 1: sfondo = Color.red; break;
            case 2: sfondo = Color.orange; break;
            case 3: sfondo = Color.yellow; break;
            case 4: sfondo = Color.green; break;
            case 5: sfondo = Color.blue; break;
            case 6: sfondo = Color.black; break;
        }

        panArea.setBackground(sfondo);
    }
}
}

```



PROGRAMMA JAVA (*Cambia.java*)

```

class Cambia
{
    public static void main(String argv[])
    {
        CambiaFrame f = new CambiaFrame();

        f.setVisible(true);
    }
}

```

11 Finestre di dialogo

Le interfacce grafiche utilizzano le **finestre di dialogo** per inviare all'utente messaggi di avvertimento o di errore oppure la richiesta di inserimento dati o di conferma all'esecuzione delle operazioni. Ci sono poi le finestre di dialogo tipiche per l'apertura e il salvataggio dei file, per la scelta dei *font* di caratteri oppure per la selezione dei colori.

La componente Swing, che rappresenta le finestre di dialogo standard, si chiama **JOptionPane**: essa produce una finestra di *popup* sul video per comunicare un messaggio all'utente o chiedere un dato. I metodi utilizzati da questa classe sono:

Metodo	Descrizione
showConfirmDialog	Chiede una conferma all'utente con possibilità di risposta <i>Sì</i> , <i>No</i> , <i>Annulla</i>
showInputDialog	Fornisce un prompt per l'inserimento di dati
showMessageDialog	Comunica un messaggio all'utente
showOptionDialog	Visualizza una finestra di carattere generale che unifica le funzionalità delle tre precedenti

Per esempio l'istruzione seguente produce sul video la finestra di dialogo in figura:

```
JOptionPane.showMessageDialog(null,
    "Dati registrati",
    "Inserimento",
    JOptionPane.INFORMATION_MESSAGE);
```



Il primo parametro indica il frame principale che può essere *this* o *null*:

- con **this** il frame principale rimane bloccato fino alla risposta dell'utente alla finestra di dialogo
- con **null** la finestra di dialogo viene visualizzata al centro dello schermo ed è indipendente dalla finestra principale.

Il secondo parametro è il messaggio che compare al centro della finestra di dialogo, mentre il terzo parametro è il titolo della finestra.

L'ultimo parametro specifica il tipo di icona da visualizzare nella parte sinistra della finestra. Le icone standard sono rappresentate con i seguenti valori per il parametro:



JOptionPane.ERROR_MESSAGE



JOptionPane.INFORMATION_MESSAGE



JOptionPane.WARNING_MESSAGE



JOptionPane.QUESTION_MESSAGE

JOptionPane.PLAIN_MESSAGE (non visualizza alcuna icona)

Si osservi che i nomi delle icone sono scritti tutto in maiuscolo.

Come si vede dalla figura della finestra precedente, viene aggiunto automaticamente il pulsante *OK*.

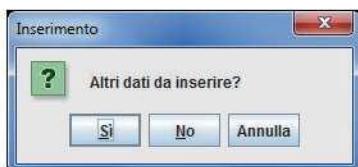
I pulsanti possono essere impostati in modo diverso da questo di *default* con un ulteriore parametro, da inserire dopo il titolo della finestra, che può assumere i seguenti valori:

JOptionPane.DEFAULT_OPTION (pulsante *OK*)

JOptionPane.YES_NO_OPTION (pulsanti *Si*, *No*)

JOptionPane.YES_NO_CANCEL_OPTION (pulsanti *Si*, *No*, *Annulla*)

JOptionPane.OK_CANCEL_OPTION (pulsanti *OK*, *Annulla*).



Per esempio per visualizzare la finestra di dialogo della figura, occorre scrivere la seguente istruzione:

```
JOptionPane.showConfirmDialog(null,
    "Altri dati da inserire?",
    "Inserimento",
    JOptionPane.YES_NO_CANCEL_OPTION,
    JOptionPane.QUESTION_MESSAGE);
```

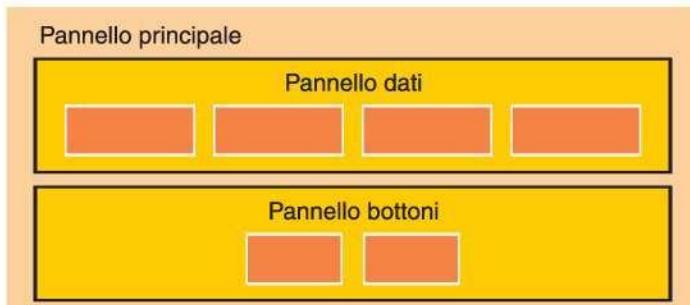
PROGETTO 11

Scrivere il programma per richiedere all'utente il cognome e il nome.

I dati da inserire devono essere considerati obbligatori, per cui se l'utente lascia una delle caselle di input vuote, il programma visualizza una finestra con un messaggio di avvertimento, altrimenti visualizza una finestra di informazione che riassume i dati inseriti.

Disegno dell'interfaccia grafica

La finestra dell'interfaccia utente contiene un pannello principale che a sua volta organizza le componenti in due pannelli secondari: il primo contiene due etichette e due caselle di testo per il cognome e il nome, il secondo contiene i pulsanti *Annulla* e *OK*.



La tabella seguente riassume i nomi degli oggetti utilizzati nell'interfaccia grafica con la relativa classe.

Classe	Nome oggetto
JFrame	f
JPanel	pan
JPanel	panDati
JLabel	lblCognome
JTextField	txtCognome
JLabel	lblNome
JTextField	txtNome
JPanel	panPulsanti
JButton	btnAnnulla
JButton	btnOk

Gestione degli eventi

All'inizio dell'esecuzione le caselle di testo sono vuote. L'utente inserisce i dati di una persona. Quando fa clic sul pulsante *OK*, il programma controlla che le due caselle di testo non siano vuote. In base al risultato del controllo mostra una diversa finestra di dialogo. Se l'utente fa clic sul pulsante *Annulla*, le caselle di testo vengono svuotate.

IMPLEMENTAZIONE DELLA CLASSE (*Anagrafe.java*)

```

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class Anagrafe extends JFrame implements ActionListener
{
    private JPanel pan = new JPanel();

    private JPanel panDati = new JPanel();
    private JLabel lblCognome = new JLabel ("Cognome ", JLabel.RIGHT);
    private JTextField txtCognome = new JTextField(10);
    private JLabel lblNome = new JLabel ("Nome ", JLabel.RIGHT);
    private JTextField txtNome = new JTextField(10);

    private JPanel panPulsanti = new JPanel();
    private JButton btnAnnulla = new JButton("Annulla");
    private JButton btnOk = new JButton("OK");

    public Anagrafe()
    {
        pan.setLayout(new BorderLayout());

        // imposta le stringhe di comando e gli ascoltatori per gli eventi
        // sui pulsanti di comando
        btnAnnulla.setActionCommand("annulla");
        btnOk.setActionCommand("ok");
        btnAnnulla.addActionListener(this);
        btnOk.addActionListener(this);
    }
}

```

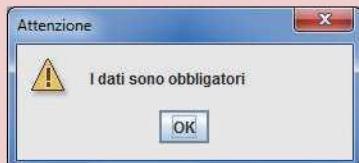
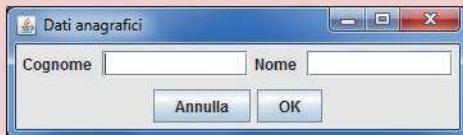
```

// inserisce le componenti nei pannelli
panDati.add(lblCognome);
panDati.add(txtCognome);
panDati.add(lblNome);
panDati.add(txtNome);
panPulsanti.add(btnAnnulla);
panPulsanti.add(btnOk);
pan.add(panDati, "Center");
pan.add(panPulsanti,"South");

// aggiunge il pannello principale al frame
this.getContentPane().add(pan, BorderLayout.CENTER);
// controlla l'uscita dal programma quando la finestra viene chiusa
this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

public void actionPerformed(ActionEvent e)
{
    if ("annulla".equals(e.getActionCommand()))
    {
        txtCognome.setText("");
        txtNome.setText("");
    }
    if ("ok".equals(e.getActionCommand()))
    {
        String cognomeLetto = txtCognome.getText();
        String nomeLetto = txtNome.getText();
        if ((cognomeLetto.equals("")) || (nomeLetto.equals("")))
        {
            JOptionPane.showMessageDialog(this,
                "I dati sono obbligatori",
                "Attenzione",
                JOptionPane.WARNING_MESSAGE);
        }
        else
        {
            String messaggio = "Dati inseriti: " + cognomeLetto + " " + nomeLetto;
            JOptionPane.showMessageDialog(this,
                messaggio,
                "Conferma",
                JOptionPane.INFORMATION_MESSAGE);
        }
    }
}
}

```



PROGRAMMA JAVA (*DatiAnagrafici.java*)

```

import javax.swing.*;

class DatiAnagrafici
{
    public static void main(String argv[])
    {
        Anagrafe f = new Anagrafe();

        f.setTitle("Dati anagrafici");
        f.pack();
        f.setVisible(true);
    }
}

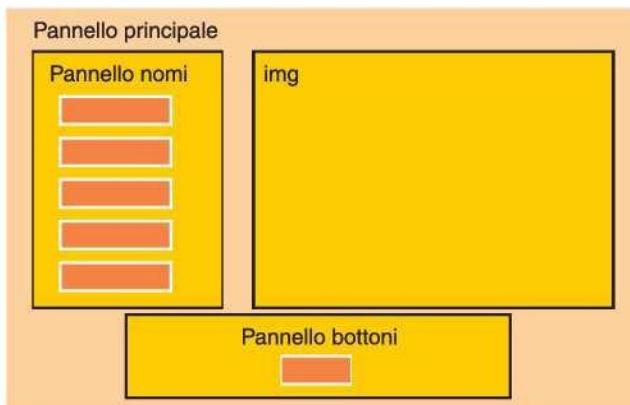
```

PROGETTO 12

Scrivere il programma per consentire all'utente la scelta in un elenco di città italiane. In corrispondenza della città selezionata, viene visualizzata una fotografia della città. Si deve anche controllare l'uscita dal programma chiedendone conferma all'utente.

Disegno dell'interfaccia grafica

La finestra dell'interfaccia utente contiene un pannello principale contenente, a sua volta, un pannello per i nomi delle città nella parte sinistra, l'immagine della città al centro e il pannello per il pulsante *Uscita* in basso. L'elenco delle città è organizzato con un gruppo di pulsanti di opzione (*JRadioButton*), in modo tale che si possa selezionare una sola città per volta. Le immagini delle città sono registrate in una sottodirectory *images* della directory dove si trova l'applicazione: hanno come nome di file il nome della città ed estensione *.jpg*.



Il layout assegnato al pannello principale è un *BorderLayout*, mentre il pannello dei nomi ha un *GridLayout* con 5 righe e 1 colonna.

La componente **JRadioButton** visualizza un piccolo cerchio, con accanto una descrizione: esso viene annerito quando l'utente fa clic su di esso. Per consentire una sola scelta possibile, i diversi *JRadioButton* devono essere assegnati ad un **ButtonGroup** (gruppo di pulsanti) con il metodo **add**.

Per una migliore gestione dei dati, ma anche per una maggiore chiarezza del codice, i nomi delle città sono organizzati in un array: il nome della città, infatti, coincide con il nome del pulsante, con il nome dell'immagine e con il nome assegnato al comando di azione per la gestione degli eventi sul *JRadioButton*.

L'immagine della città è contenuta in una componente **JLabel**: l'immagine viene associata alla *JLabel* attraverso il metodo **setIcon**.

La tabella seguente riassume i nomi degli oggetti utilizzati nell'interfaccia grafica con la relativa classe.

Classe	Nome oggetto
JFrame	f
JPanel	pan
JPanel	panNomi
JRadioButton	rbCitta
ButtonGroup	bg
JLabel	imgCitta
JPanel	panPulsanti
JButton	btnUscita

Gestione degli eventi

L'utente interagisce con il programma facendo clic sui pulsanti di opzione. A seguito di un clic, viene mostrata una fotografia della città e al centro del video compare una finestra di dialogo che visualizza il nome della città scelta. L'utente può uscire dal programma facendo clic sul pulsante di *Uscita* che attiva una finestra di dialogo per chiedere la conferma.

L'utente ha dunque due possibilità per uscire dal programma: con il pulsante di chiusura in alto a destra della finestra oppure con il pulsante *Uscita*.

IMPLEMENTAZIONE DELLA CLASSE (*Immagini.java*)

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class Immagini extends JFrame implements ActionListener
{
    static String arrCitta[] = new String[5];
    private JPanel pan = new JPanel();
    private JPanel panNomi = new JPanel();
    private JRadioButton rbCitta[] = new JRadioButton[5];
    private ButtonGroup bg = new ButtonGroup();
    private JLabel imgCitta = new JLabel();
    private JPanel panPulsanti = new JPanel();
    private JButton btnUscita = new JButton("Uscita");
}
```

```
public Immagini()
{
    arrCitta[0] = "Milano";
    arrCitta[1] = "Roma";
    arrCitta[2] = "Napoli";
    arrCitta[3] = "Venezia";
    arrCitta[4] = "Firenze";

    pan.setLayout(new BorderLayout());
    panNomi.setLayout(new GridLayout(5,1));

    // crea i pulsanti di opzione, li aggiunge al gruppo
    // e imposta la stringa di comando per l'evento sul pulsante
    for (int i=0; i<5; i++)
    {
        rbCitta[i] = new JRadioButton(arrCitta[i]);
        bg.add(rbCitta[i]);
        rbCitta[i].setActionCommand(arrCitta[i]);
    }

    // imposta la prima opzione come preselezionata
    rbCitta[0]. setSelected(true);
    imgCitta.setIcon(new ImageIcon("images/" + arrCitta[0] + ".jpg"));

    // imposta la stringa di comando per l'evento sul pulsante di uscita
    btnUscita.setActionCommand("uscita");

    // registra gli ascoltatori per i pulsanti
    for (int i=0; i<5; i++) rbCitta[i].addActionListener(this);
    btnUscita.addActionListener(this);

    // inserisce le componenti nei pannelli
    for (int i=0; i<5; i++) panNomi.add(rbCitta[i]);
    panPulsanti.add(btnUscita);
    pan.add(panNomi, "West");
    pan.add(imgCitta, "Center");
    pan.add(panPulsanti,"South");

    // aggiunge il pannello al frame
    this.getContentPane().add(pan, BorderLayout.CENTER);
    // controlla l'uscita dal programma quando la finestra viene chiusa
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}

// gestione degli eventi
public void actionPerformed(ActionEvent e)
{
    if ("uscita".equals(e.getActionCommand()))
    {
```

```

int n = JOptionPane.showConfirmDialog(null,
    "Sei sicuro di voler uscire dal programma?",
    "Uscita dal programma",
    JOptionPane.YES_NO_OPTION);
if (n == JOptionPane.YES_OPTION)
{
    System.exit(0);
}
else
{
    imgCitta.setIcon(new ImageIcon("images/" + e.getActionCommand() + ".jpg"));
    String messaggio = "Città scelta: " + e.getActionCommand();
    JOptionPane.showMessageDialog(null,
        messaggio,
        "Conferma",
        JOptionPane.INFORMATION_MESSAGE);
}
}

```

PROGRAMMA JAVA (*Cittaitaliane.java*)

```

class CittaItaliane
{
    public static void main(String argv[])
    {
        Immagini f = new Immagini();

        f.setTitle("Città Italiane");
        f.setSize(250, 200);
        f.setVisible(true);
    }
}

```



AUTOVERIFICA

Problemi da 15 a 21 pag. 348



6. Evento sulla modifica delle caselle di testo

12 I menu

I menu con Swing sono realizzati dalle componenti:

- **JMenuBar** per la barra dei menu,
- **JMenu** per i menu a tendina inseriti nella barra,
- **JMenuItem** per le voci contenute nei menu.

La barra dei menu, per convenzione, si trova all'esterno del *content pane* che contiene le componenti grafiche della finestra ed è posizionata in alto nella finestra stessa.

Ogni barra può contenere uno o più menu a tendina e ogni menu può contenere una o più voci, oltre alle linee di separazione tra gruppi di voci. Il menu viene aggiunto alla barra con il metodo **add**. Anche la voce viene aggiunta al menu con il metodo **add**.

La barra dei menu viene assegnata al frame con il metodo **setJMenuBar**.

Le seguenti linee di codice illustrano le modalità per costruire il menu *File* e per inserire in esso la voce di menu *Nuovo*:

```
JMenuBar barra;
JMenu menu;
JMenuItem voce;

menu = new JMenu("File");
barra.add(menu);
voce = new JMenuItem("Nuovo")
menu.add(voce);
```

Per aggiungere una linea di separazione tra gruppi di voci, nello stesso menu, si usa il metodo **addSeparator**.

```
menu.addSeparator();
```

Nelle interfacce dei software spesso ai menu e alle voci di menu vengono associati caratteri alfabetici, mnemonici, che corrispondono al tasto da premere insieme al tasto **Alt** per selezionare il menu o la voce con una scorciatoia da tastiera (*shortcut*). La lettera può essere una tra quelle che compongono la descrizione del menu o della voce, ma deve essere univoca all'interno dello stesso menu. La lettera scelta risulta sottolineata nella visualizzazione del menu.

Per esempio le istruzioni seguenti impostano la lettera F come mnemonica per il menu e la lettera N per la voce di menu:

```
menu.setMnemonic(KeyEvent.VK_F);
voce.setMnemonic(KeyEvent.VK_N);
```

Le lettere sono rappresentate con le costanti **VK_***lettera*. Non viene fatta alcuna distinzione tra maiuscole e minuscole.

Per le componenti di tipo *JMenuItem* si può usare una forma più compatta, inserendo la specificazione della lettera come secondo parametro, direttamente nell'istruzione di creazione della voce:

```
voce = new JMenuItem("Nuovo", KeyEvent.VK_N);
```

PROGETTO 13

Costruire un'applicazione per l'editing di testi che consenta all'utente di scegliere le funzioni del programma all'interno della barra dei menu.

Il programma simula un editor di testi con le funzionalità per creare un nuovo testo, per salvare su disco il testo e per aprire un testo già esistente. Sono implementate anche le funzioni di *Taglia*, *Copia* e *Incolla*, tipiche nell'editing dei testi.

Disegno dell'interfaccia grafica

La finestra dell'applicazione contiene la barra dei menu e un pannello di tipo **JScrollPane** che fornisce un riquadro il cui contenuto può essere letto attraverso le barre di scorrimento orizzontali e verticali. Le barre di scorrimento sono visualizzate automaticamente quando le dimensioni del contenuto superano le dimensioni del riquadro. Il testo è rappresentato con una componente **JTextArea**, che è un'area di testo multilinea per visualizzare testo senza evidenziazioni.

La struttura della barra dei menu è illustrata dalla seguente tabella:

Menu	Voce	Tasto mnemonico
File		F
	Nuovo	N
	Apri	R
	Salva	S
	separatore	
	Esci	E
Modifica		M
	Taglia	L
	Copia	C
	Incolla	O

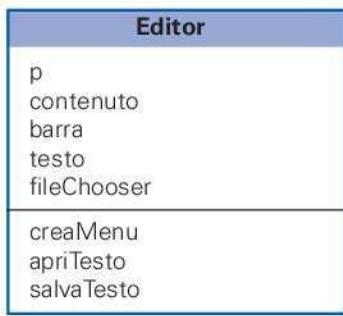


Per le operazioni di *Apri* e *Salva*, si utilizza una componente **JFileChooser**: esso è lo strumento che consente di navigare nelle directory e sottodirectory del disco per selezionare un file. I metodi usati nel programma sono:

- **showOpenDialog** per l'apertura di un file esistente
- **showSaveDialog** per il salvataggio su disco.

Questi metodi provocano la visualizzazione delle finestre di dialogo *Apri* e *Salva*, tipiche dei software con interfaccia grafica.

L'applicazione si basa sulla classe *Editor* che viene creata come sottoclasse di *JFrame* e possiede gli attributi e i metodi indicati nel seguente diagramma:



Gestione degli eventi

L'utente interagisce con questo programma azionando le voci della barra dei menu. Il controllo sulla voce di menu selezionata è realizzato con il metodo **getSource** per determinare la sorgente dell'evento generato:

```
JMenuItem source = (JMenuItem)(e.getSource());  
String s = source.getText();
```

Il metodo **getText** restituisce la descrizione della sorgente dell'evento, che corrisponde alla descrizione della voce di menu. In base alla voce di menu scelta, viene attivata l'esecuzione del metodo delegato alla sua gestione.

IMPLEMENTAZIONE DELLA CLASSE (*Editor.java*)

```
import javax.swing.*;  
import javax.swing.text.*;  
import java.awt.*;  
import java.awt.event.*;  
import java.io.*;  
  
public class Editor extends JFrame implements ActionListener  
{  
    private JPanel p = new JPanel();  
    private JScrollPane contenuto = new JScrollPane();  
    private JMenuBar barra = new JMenuBar();  
    private JTextArea testo = new JTextArea();  
    private JFileChooser fileChooser = new JFileChooser();  
  
    public Editor()  
    {  
        // crea la barra dei menu  
        creaMenu();  
        this.setJMenuBar(barra);  
  
        // imposta le caratteristiche dell'area di testo  
        testo = new JTextArea(25, 60);  
        testo.setEditable(true);  
        testo.setFont(new Font("Arial", Font.PLAIN, 12));  
        testo.setLineWrap(true);  
        testo.setWrapStyleWord(true);  
  
        // aggiunge le componenti del frame  
        contenuto = new JScrollPane(testo);  
        p.add(contenuto);  
        this.getContentPane().add(p, BorderLayout.CENTER);  
  
        // controlla l'uscita dal programma quando la finestra viene chiusa  
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);  
    }
```

```
// creazione delle voci di menu
public JMenuBar creaMenu()
{
    JMenu menu;
    JMenuItem voce;

    // menu File
    menu = new JMenu("File");
    menu.setMnemonic(KeyEvent.VK_F);
    barra.add(menu);

    // voci
    voce = new JMenuItem("Nuovo", KeyEvent.VK_N);
    voce.addActionListener(this);
    menu.add(voce);

    voce = new JMenuItem("Apri", KeyEvent.VK_R);
    voce.addActionListener(this);
    voce.setActionCommand("apri");
    menu.add(voce);

    voce = new JMenuItem("Salva", KeyEvent.VK_S);
    voce.addActionListener(this);
    menu.add(voce);

    menu.addSeparator();

    voce = new JMenuItem("Esci", KeyEvent.VK_E);
    voce.addActionListener(this);
    voce.setActionCommand("esci");
    menu.add(voce);

    // menu Modifica
    menu = new JMenu("Modifica");
    menu.setMnemonic(KeyEvent.VK_M);
    barra.add(menu);

    // voci
    voce = new JMenuItem("Taglia", KeyEvent.VK_L);
    voce.addActionListener(this);
    menu.add(voce);
    voce = new JMenuItem("Copia", KeyEvent.VK_C);
    voce.addActionListener(this);
    menu.add(voce);
    voce = new JMenuItem("Incolla", KeyEvent.VK_O);
    voce.addActionListener(this);
    menu.add(voce);

    return barra;
}
```

```
// funzioni del menu
public void apriTesto()
{
    int status = fileChooser.showOpenDialog(this);
    if (status==JFileChooser.APPROVE_OPTION)
    {
        try
        {
            File f = fileChooser.getSelectedFile();
            Reader fIN = new FileReader(f);
            testo.read(fIN,null);
            setTitle(f.getName());
        }
        catch(Exception e) {}
    }
}

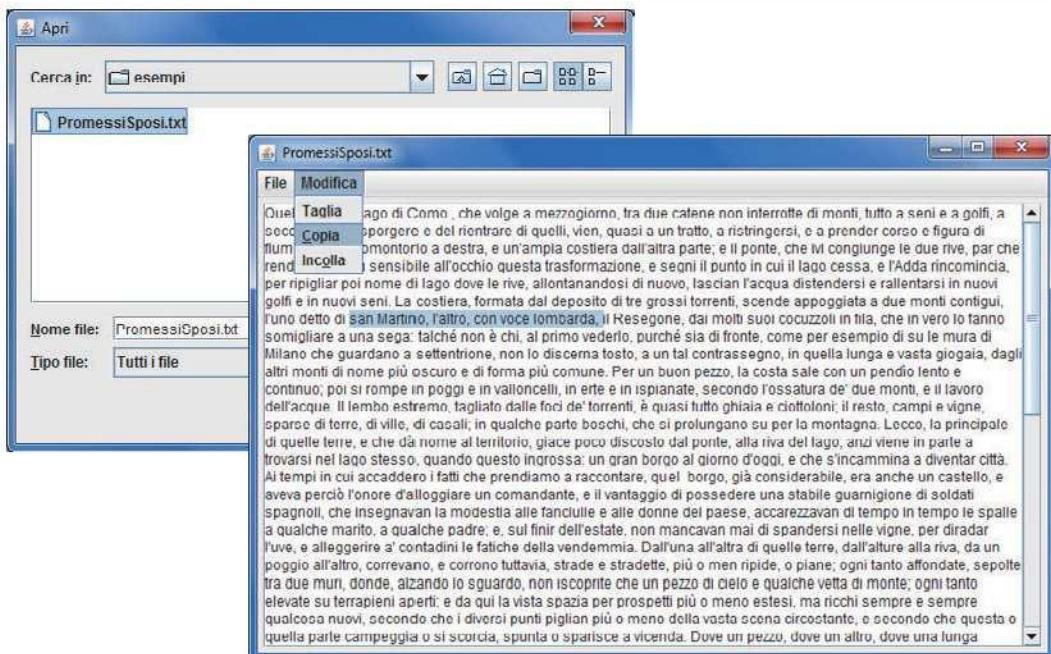
public void salvaTesto()
{
    int status = fileChooser.showSaveDialog(this);
    if (status==JFileChooser.APPROVE_OPTION)
    {
        try
        {
            File f = fileChooser.getSelectedFile();
            Writer fOUT = new FileWriter(f);
            testo.write(fOUT);
            setTitle(f.getName());
        }
        catch(Exception e) {}
    }
}

// gestione degli eventi
public void actionPerformed(ActionEvent e)
{
    JMenuItem source = (JMenuItem)(e.getSource());
    String s = source.getText();
    if (s.equals("Nuovo"))
    {
        testo.setText("");
        setTitle("Nuovo");
    }
    if (s.equals("Apri")) apriTesto();
    if (s.equals("Salva")) salvaTesto();
    if (s.equals("Esci")) System.exit(0);
    if (s.equals("Taglia")) testo.cut();
    if (s.equals("Copia")) testo.copy();
    if (s.equals("Incolla")) testo.paste();
}
}
```

PROGRAMMA JAVA (*EditorTesto.java*)

```
class EditorTesto
{
    public static void main(String argv[])
    {
        Editor f = new Editor();

        f.setTitle("Editor di testo");
        f.pack();
        f.setVisible(true);
    }
}
```



Le caratteristiche dell'area di testo sono impostate con le seguenti istruzioni:

```
testo.setEditable(true);
```

rende editabile il contenuto dell'area di testo,

```
testo.setFont(new Font("Arial", Font.PLAIN, 12));
```

imposta il font di visualizzazione,

```
testo.setLineWrap(true);
```

porta a capo una linea quando si raggiunge l'estremo destro dell'area di testo,

```
testo.setWrapStyleWord(true);
```

porta a capo la parola intera (determinata dallo spazio bianco tra i caratteri) quando si raggiunge l'estremo destro dell'area di testo.

L'esecuzione del metodo **showOpenDialog** sulla componente **FileChooser** restituisce un valore di stato che è uguale a **APPROVE_OPTION** se l'operazione è stata eseguita correttamente.

Il metodo **getSelectedFile** poi restituisce il file selezionato. A questo punto il flusso associato al file viene copiato nell'area di testo con il metodo **read** e il titolo della finestra è sostituito con il nome del file (**setTitle**):

```
int status = fileChooser.showOpenDialog(this);
if (status==JFileChooser.APPROVE_OPTION)
{
    try
    {
        File f = fileChooser.getSelectedFile();
        Reader fIN = new FileReader(f);
        testo.read(fIN,null);
        setTitle(f.getName());
    }
    catch(Exception e) {}
}
```

Considerazioni analoghe possono essere fatte per l'operazione di scrittura sul file con la scelta *Salva*.

Le operazioni del menu *Modifica* (*Taglia*, *Copia* e *Incolla*) sono gestite con i metodi predefiniti della componente *JTextArea*:

- **cut**
- **copy**
- **paste**.

AUTOVERIFICA

Problemi da 22 a 24 pag. 348



-
- 7. Simulatore di un miscelatore di acqua**
8. Rilevazione di dati in un stazione meteorologica
-

PROBLEMI

Elementi base dell'interfaccia utente

- 1 Descrivere a parole e schematizzare con un disegno l'interfaccia grafica utilizzabile nelle seguenti situazioni, specificando quali sono gli elementi grafici usati: una finestra per gestire l'input di due numeri qualunque; una maschera per inserire una data composta da giorno, mese e anno; un programma per disegnare (per esempio il programma *Paint* di Windows); un programma che gestisce la spedizione di messaggi di posta elettronica.
- 2 Basandosi sui software conosciuti, elencare alcuni esempi di eventi che vengono generati quando si utilizza il mouse.
- 3 Definire, con il linguaggio Java, le seguenti finestre sia con la libreria Swing che con AWT:
 - a) una finestra di dimensione 320x240
 - b) una finestra di dimensione 100x100 contenente due pulsanti, *Ok* e *Annulla*
 - c) una finestra con due etichette contenenti due nomi di persona.
- 4 Scrivere un programma Java che acquisisca da riga di comando l'anno di nascita dell'utente (interfaccia a caratteri) e, successivamente, apra una finestra grafica per visualizzare l'età in un'etichetta (interfaccia grafica).

Programmazione visuale con NetBeans

- 5 Creare un progetto e inserire nella finestra di progettazione gli oggetti necessari per implementare le componenti seguenti:
 - a) un'etichetta con la scritta *Inserisci il nome*,
 - b) un'etichetta con la scritta *1.234,5* allineata a destra,
 - c) un'etichetta con la scritta *Programma 8.0* di colore blu su sfondo giallo,
 - d) un pulsante con la scritta *OK*,
 - e) un pulsante con la scritta *Annulla* disabilitato,
 - f) una casella di testo larga 200 pixel e alta 30 pixel,
 - g) una casella di testo contenente la scritta *Msg fisso* e resa non modificabile.
- 6 Modificare il Progetto 2 *CalcolatriceProject* presentato nell'inserto, aggiungendo un pulsante *Azzera*: quando l'utente fa clic su di esso, il programma pulisce le tre caselle di testo impostando una stringa vuota.
- 7 Creare un'applicazione con interfaccia grafica per calcolare il prodotto tra due numeri.
- 8 Creare un'applicazione con interfaccia grafica in cui l'utente può inserire la velocità in chilometri orari e, dopo aver fatto clic su un pulsante di conversione, ottiene il valore espresso in metri al secondo.
- 9 Creare un'applicazione grafica per effettuare calcoli con la legge di Ohm. Dati V (differenza di potenziale) e I (intensità della corrente), calcolare la resistenza R con la formula $R=V/I$. Descrivere anche quali componenti grafiche si intendono utilizzare per l'interfaccia utente.
- 10 Scrivere l'applicazione per la conversione di una misura da pollici a centimetri e per la conversione inversa, presentando all'utente due pulsanti per scegliere il verso della conversione.

Componenti grafiche

- 11 Scrivere il codice Java per inserire i seguenti oggetti in una finestra grafica:
 - a) un'area di testo di dimensioni 5x10
 - b) un'area di testo contenente tre numeri su ogni riga
 - c) una casella combinata per scegliere un giudizio tra i seguenti valori: ottimo, buono, sufficiente e insufficiente
 - d) una casella combinata per scegliere un numero tra 1 e 31
 - e) una casella di controllo per selezionare uno o più dei dodici mesi dell'anno.

- 12** Ripetere i problemi precedenti, usando l'ambiente di sviluppo visuale *NetBeans*.
- 13** Scrivere il programma Java per ciascuna delle seguenti interfacce, disponendo gli elementi grafici nei contenitori con un opportuno layout:
- un pulsante *OK* e *Annulla* all'interno di un pannello
 - una casella di testo preceduta da una scritta
 - in una finestra con disposizione *BorderLayout*, un pulsante per ogni zona
 - nella parte alta di una finestra una casella di testo, nella parte bassa un'area di testo
 - tre bottoni in un pannello disposti verticalmente
 - una finestra contenente tre pannelli di colore verde, bianco e rosso.
- 14** Disegnare all'interno di una finestra una griglia 4x4 contenente 16 caselle di testo. Colorare le celle usando due colori diversi.

Gestione di eventi

- 15** Creare un'interfaccia grafica con un'area di testo formata da 60 colonne e 10 righe in cui l'utente può scrivere del testo. Aggiungere anche un pulsante: quando l'utente fa clic su di esso, il programma pulisce l'area di testo cancellando tutto il suo contenuto.
- 16** Creare un'interfaccia grafica composta da una casella di testo, un pulsante e un'area di testo. L'utente utilizza la casella di testo per inserire dei nomi. Quando fa clic sul pulsante, i nomi vengono ricopiatati nell'area di testo.
- 17** Scrivere il programma per richiedere all'utente un numero e controllare che il numero inserito sia maggiore di 500; se il valore non è maggiore, il programma visualizza una finestra con un messaggio di avvertimento, altrimenti apre una finestra per visualizzare il valore inserito.
- 18** Scrivere il programma per consentire all'utente la scelta in un elenco di animali. In corrispondenza della scelta effettuata, viene visualizzata l'immagine dell'animale.
- 19** Scrivere il programma per consentire all'utente la scelta in un elenco di colori. Il colore scelto diventa lo sfondo di una casella di testo posizionata al centro della finestra. Controllare anche l'uscita dal programma chiedendone conferma all'utente.
- 20** Simulare la parte della finestra di dialogo per le stampe tipica delle applicazioni in Windows con la quale si può scegliere l'orientamento (orizzontale o verticale) della pagina da stampare: usare i pulsanti di opzione.
- 21** Creare una finestra che resta aperta finché l'utente non inserisce in una casella di testo la parola *FINE*.

Menu

- 22** Aprire un applicativo in ambiente Windows: osservare le voci presenti nei primi due menu e riprodurre in una finestra la struttura dei menu con le voci di ciascuno.
- 23** Costruire un programma che consente di scegliere un nome di nazione all'interno di un elenco organizzato come menu a tendina. Il nome scelto viene scritto con un'etichetta all'interno della finestra.
- 24** In una finestra sono inseriti un pulsante e un'area di testo: facendo clic sul pulsante si apre la finestra di dialogo che consente di selezionare un file presente sul disco; il nome del file selezionato viene poi aggiunto all'area di testo.