Fondamenti di informatica
un approccio a oggetti con Java

Luca Cabibbo

Interfacce grafiche

Capitolo 27

settembre 2001

Contenuti

- Interfacce grafiche
  - la classe **JFrame**
  - componenti
  - gestori di layout
- Interfacce grafiche e IDE

Introduzione

Le applicazioni che vengono normalmente utilizzate dagli utenti sono basate su una interfaccia grafica

- l'interfaccia grafica di una applicazione è dotata di componenti grafici
  - ad esempio, finestre, bottoni, menu, aree di testo
- l'utente interagisce con l'applicazione interagendo con la sua interfaccia utente
  - l'utente agisce cliccando con il mouse sui bottoni e selezionando voci dai menu, oppure operando sulla tastiera

Vengono ora studiati alcuni componenti grafici, e il loro utilizzo nella definizione di interfacce grafiche

- gli eventi, che sono il meccanismo di gestione delle interfacce grafiche, ovvero delle azioni eseguite dagli utenti, saranno studiati nel capitolo successivo

Interfacce grafiche

L'interfaccia grafica (o GUI, graphical user interface) di un programma è normalmente basata su

- un certo numero di elementi grafici, chiamati **componenti**
  - ad esempio, bottoni, etichette, immagini, aree di testo, aree dedicate al disegno, finestre
- un certo numero di componenti utilizzati per raggruppare altri componenti, chiamati **contenitori**
  - ad esempio, la finestra principale del programma, che contiene gli altri componenti dell'interfaccia grafica
- altri oggetti di supporto
  - i **gestori di layout** — oggetti che regolano il posizionamento dei componenti nei contenitori
  - i **gestori degli eventi** — oggetti che gestiscono l'interazione tra l'utente e i vari elementi dell'interfaccia grafica

Interfacce grafiche e Swing

Le API di Java supportano la definizione di interfacce grafiche mediante le tecnologie AWT (implementata principalmente dal package **java.awt**) e Swing (implementata principalmente dal package **javax.swing**)

Con riferimento alla tecnologia Swing, una interfaccia grafica è normalmente composta da

- un contenitore principale
  - ad esempio, un applet (**JApplet**), una frame (**JFrame**) o una finestra di dialogo (**JDialog**)
- un pannello (**JPanel**), usato come contenitore intermedio
  - il pannello è l'unico componente contenuto nel contenitore principale, ed è usato come contenitore di altri componenti atomici e/o di altri pannelli
- componenti "atomici", che vengono effettivamente visualizzati nell'interfaccia grafica
  - ad esempio, bottoni (**JButton**), etichette (**JLabel**), campi di testo (**JTextField**), aree di testo (**JTextArea**)

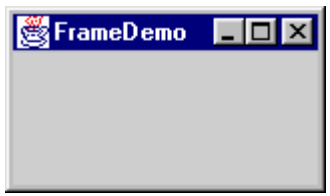
Alcuni componenti Swing

Le figure esemplificano alcuni componenti Swing

- contenitori principali

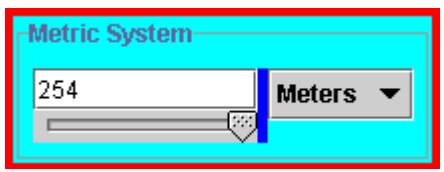
applet

finestra di dialogo



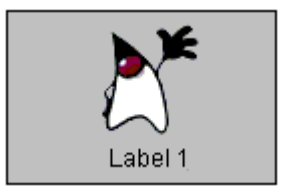
frame

- contenitori intermedi



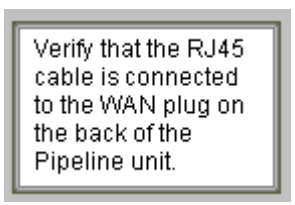
pannello (contiene altri componenti)

- alcuni componenti atomici



etichetta

campi di testo



area di testo

Una semplice interfaccia grafica e i suoi componenti

La seguente figura mostra una semplice interfaccia grafica

- il contenitore principale è una frame
  - una frame è una finestra dotata di una cornice, che comprende, tra l'altro, la barra del titolo della finestra
- la frame contiene un pannello come contenitore intermedio
  - i contenitori intermedi non sono visibili, ma sono utili per la disposizione dei componenti grafici
- il pannello contiene come componenti atomici un bottone (a sinistra) e una etichetta (a destra)
  - i componenti atomici sono invece visibili nell'interfaccia e, quindi, utilizzabili per fornire informazioni all'utente e ricevere richieste di esecuzione di operazioni

La classe JFrame

Il contenitore principale per un programma dotato di una interfaccia grafica può essere

- un applet (il programma è un applet)
- una frame (il programma è una applicazione)
  - una frame è una finestra autonoma, dotata di una cornice
- una finestra di dialogo (che non costituisce un programma a sé stante)

Nel seguito sarà considerato solo il caso di applicazioni basate su frame (la definizione di interfacce grafiche basate su applet è simile)

- in generale, una frame è un oggetto istanza che, polimorficamente, è di tipo **Frame** (tecnologia AWT) oppure **JFrame** (tecnologia Swing)
- la classe **JFrame** del package **javax.swing** è il progetto di una frame elementare, che può essere estesa per realizzare il comportamento voluto

Un primo esempio — una frame sullo schermo

```
import javax.swing.JFrame;

/** Applicazione che visualizza una frame sullo schermo. */
public class FrameSemplice extends JFrame {
    /* Crea una nuova FrameSemplice. */
    public FrameSemplice() {
        super();
        this.setTitle("Una frame semplice");
        this.setSize(300, 100);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        this.show();
    }
    /* L'applicazione FrameSemplice. */
    public static void main(String[] args) {
        /* crea una nuova FrameSemplice */
        new FrameSemplice();
    }
}
```

Una frame sullo schermo

Questa è la definizione di una semplice frame che si visualizza sullo schermo

- per realizzare una frame è necessario definire una classe che estende la classe **JFrame** (del package **javax.swing**)
- il costruttore della classe serve a inizializzare gli oggetti creati dalla classe
- la classe è anche una classe applicazione
  - la classe contiene la definizione del metodo di classe **void main(String[] args)**, che ha solitamente solo lo scopo di inizializzare l'applicazione (creando, tra l'altro, un oggetto dalla classe, che costituisce l'interfaccia grafica dell'applicazione) e di avviarne l'esecuzione

Una frame sullo schermo

```
/* Crea una nuova FrameSemplice. */
public FrameSemplice() {
    super();
    this.setTitle("Una frame semplice");
    this.setSize(300, 100);
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    this.show();
}
```

Il costruttore della frame normalmente fa quanto segue

- invoca il costruttore della superclasse (come devono fare tutti i costruttori delle classi estese) mediante l'istruzione **super();**
- assegna alcune proprietà della frame, ad esempio
  - un titolo, mediante il metodo **setTitle(...)**
  - le dimensioni, mediante il metodo **setSize(...)**
  - specifica che la chiusura della frame comporta la terminazione dell'intera applicazione, mediante l'invocazione di **setDefaultCloseOperation(...)**
- visualizza la frame, invocando il metodo **show()**

Componenti

Vengono ora descritti alcuni componenti che possono essere utilizzati nella realizzazione di interfacce grafiche

- ciascuna tipologia di componente è rappresentata da una sotto-classe della classe **JComponent** package **javax.swing**
  - ad esempio, i bottoni sono rappresentati dalla classe **JButton** (del package **javax.swing**), che estende la classe **JComponent**
- i componenti sono oggetti istanza di tali classi
  - ad esempio, per bottone si intende un oggetto istanza della classe **JButton**

Creazione e uso di componenti

Per utilizzare un componente ad una interfaccia grafica, bisogna

- creare il componente (ovvero, un oggetto dalla classe opportuna)
- inserire il componente nell'interfaccia grafica
  - di norma, i componenti non vanno inseriti direttamente nel contenitore principale per l'interfaccia grafica, ma vanno piuttosto inseriti in un pannello, usato come contenitore intermedio
  - dopo aver inserito tutti i componenti nel contenitore intermedio, il contenitore intermedio va inserito nel contenitore principale

Struttura di una interfaccia grafica

Viene ora descritta una possibile struttura su cui basare la definizione di interfacce grafiche

- è solo un possibile modo di procedere, sufficiente nei casi più elementari
- l'applicazione è basata sulla definizione di una classe che estende **JFrame**
- il metodo di classe **main** crea e avvia l'applicazione
- vengono usate variabili d'istanza per i componenti dell'interfaccia grafica
- il costruttore della classe inizializza la frame, ma delega l'inizializzazione dell'interfaccia grafica ad un metodo di supporto **void inizializzaGUI()**
- il metodo **inizializzaGUI()** crea un pannello, crea i componenti atomici dell'interfaccia, inserisce i componenti nel pannello e inserisce il pannello nel contenitore principale

Gli esempi saranno mostrati come completamento della definizione di questa classe

Struttura di una interfaccia grafica

```
import javax.swing.*;

/** Una applicazione Swing. */
public class ApplicazioneX extends JFrame {
    ... variabili d'istanza per i componenti
        dell'interfaccia ...

    ... altre variabili d'istanza ...

    ... costruttore ...

    ... metodo per l'inizializzazione dell'interfaccia ...

    ... altri metodi ...

    /* L'applicazione ApplicazioneX. */
    public static void main(String[] args) {
        /* crea ed avvia una nuova ApplicazioneX */
        new ApplicazioneX();
    }
}
```

Struttura di una interfaccia grafica

```
/* Crea una nuova ApplicazioneX. */
public ApplicazioneX() {
    /* inizializza la frame */
    super();

    /* l'esecuzione dell'applicazione termina con
     * la chiusura della frame */
    this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

    /* assegna un titolo alla frame */
    this.setTitle("Una semplice interfaccia grafica");

    /* inizializza l'interfaccia dell'applicazione */
    this.inizializzaGUI();

    /* dimensiona e visualizza la frame */
    this.pack();
    this.show();
}
```

Struttura di una interfaccia grafica

```
/* Inizializza l'interfaccia dell'applicazione. */
private void inizializzaGUI() {
    /* pannello, usato come contenitore intermedio */
    JPanel jPanel;

    /* crea il pannello */
    jPanel = new JPanel();

    ... crea i componenti dell'interfaccia ...

    ... aggiunge i componenti al pannello ...

    /* aggiunge il pannello alla frame */
    this.getContentPane().add(jPanel);
}
```

Esempio — una frame con un bottone e una etichetta

Viene ora mostrata la definizione di una interfaccia basata su frame e contenente un bottone e una etichetta

- un bottone è
  - una istanza di **JButton**
  - un'area rettangolare cliccabile, con un bordo, in cui è normalmente scritta una stringa oppure visualizzata una icona, che possono essere specificate al momento della creazione del bottone
- una etichetta è
  - una istanza di **JLabel**
  - un'area rettangolare senza bordi in cui è scritta una stringa oppure visualizzata una icona, che possono essere specificate al momento della creazione dell'etichetta

Una frame con un bottone e una etichetta

Dichiarazione delle variabili d'istanza per il bottone e l'etichetta

```
private JButton jButton;     // un bottone
private JLabel jLabel;     // una etichetta
```

Crea i componenti e li inserisce nel pannello

- istruzioni nel corpo del metodo **inizializzaGUI()**

```
/* crea i componenti dell'interfaccia */
jButton = new JButton("Un bottone");
jLabel = new JLabel("Una etichetta");

/* aggiunge i componenti al pannello */
jPanel.add(jButton);
jPanel.add(jLabel);
```

Esempio — campi di testo e aree di testo

Vengono ora mostrate interfacce contenenti, tra l'altro, campi di testo e aree di testo

- un campo di testo è
  - una istanza di **JTextField**
  - una linea di testo editabile
- una area di testo è
  - una istanza di **JTextArea**
  - un gruppo di linee di testo, editabile

Una frame con una etichetta e un campo di testo

Dichiarazione delle variabili d'istanza per l'etichetta e il campo di testo

```
private JLabel jLabel;    // una etichetta
private JTextField jTextField;    // un campo di testo
```

Crea i componenti e li inserisce nel pannello

```
/* crea i componenti dell'interfaccia */
jLabel = new JLabel("Scrivi il tuo nome");
/* crea un campo di testo lungo 30 caratteri */
jTextField = new JTextField(30);

/* aggiunge i componenti al pannello */
jPanel.add(jLabel);
jPanel.add(jTextField);
```

Una frame con una etichetta e un'area di testo

Dichiarazione delle variabili d'istanza

```
private JLabel jLabel;    // una etichetta
private JTextArea jTextArea;    // un'area di testo
```

Crea i componenti e li inserisce nel pannello

```
/* crea i componenti dell'interfaccia */
jLabel = new JLabel("I tuoi commenti");
/* crea un'area di testo di 5 linee,
 * ciascuna lunga 30 caratteri */
jTextArea = new JTextArea(5,30);

/* aggiunge i componenti al pannello */
jPanel.add(jLabel);
jPanel.add(jTextArea);
```

La classe JButton

Un oggetto **JButton** rappresenta un bottone cliccabile

- costruttori
  - **JButton(String s)** crea un nuovo bottone contenente l'etichetta **s**
  - **JButton(Icon i)** crea un nuovo bottone contenente l'immagine **i**
  - **JButton()** crea un nuovo bottone, senza etichetta né immagine
- metodi
  - **void setText(String s)** assegna l'etichetta al bottone
  - **String getText()** accede all'etichetta del bottone
  - **void setIcon(Icon i)** assegna l'immagine al bottone
  - **Icon getIcon()** accede all'immagine del bottone
  - **void setEnabled(boolean b)** abilita (**true**) o disabilita (**false**) il bottone (si veda il capitolo sugli eventi)

La classe JLabel

Un oggetto **JLabel** rappresenta un'area per visualizzare un breve stringa (non modificabile) o un'immagine

- costruttori
  - **JLabel(String s)** crea una nuova etichetta contenente il testo **s**
  - **JLabel(Icon i)** crea una nuova etichetta contenente l'immagine **i**
  - **JLabel()** crea una nuova etichetta, senza testo né immagine
- metodi
  - **void setText(String s)** assegna il testo all'etichetta
  - **String getText()** accede al testo dell'etichetta
  - **void setIcon(Icon i)** assegna l'immagine all'etichetta
  - **Icon getIcon()** accede all'immagine dell'etichetta

La classe JTextField

Un oggetto **JTextField** rappresenta un'area per visualizzare una linea di testo editabile (ovvero, modificabile)

- costruttori
  - **JTextField(String s)** crea un nuovo campo di testo contenente il testo **s**
  - **JTextField(int n)** crea un nuovo campo di testo di lunghezza **n**, inizialmente vuoto
  - **JTextField()** crea un nuovo campo di testo, inizialmente vuoto
- metodi
  - **void setText(String s)** assegna il testo al campo di testo
  - **String getText()** accede al testo del campo di testo
  - **String getSelectedText()** accede alla porzione di testo selezionata nel campo di testo
  - **void setEditable(boolean b)** abilita (**true**) o disabilita (**false**) la possibilità di modificare il campo di testo

La classe JTextArea

Un oggetto **JTextArea** rappresenta un'area per visualizzare un gruppo di linee di testo editabili

- costruttori
  - **JTextArea(String s)** crea una nuova area di testo contenente il testo **s**
  - **JTextArea(int r, int c)** crea una nuova area di testo composta da **r** righe e **c** colonne, vuota
  - **JTextArea()** crea una nuova area di testo, vuota
- metodi
  - **void append(String s)** aggiunge la stringa **s** alla fine dell'area di testo
  - **void setText(String s)** assegna il testo all'area di testo
  - **String getText()** accede al testo dell'area di testo
  - **String getSelectedText()** accede alla porzione di testo selezionata nell'area di testo
  - **void setEditable(boolean b)** abilita (**true**) o disabilita (**false**) la possibilità di modificare l'area di testo

Gestori di layout

Per inserire un componente in un contenitore è stato utilizzato il metodo **add(Component c)**

- nell'invocare questo metodo, viene specificato il componente da inserire nel contenitore, ma normalmente non viene specificata la posizione che il componente dovrà occupare nel contenitore
- in effetti, è possibile specificare la posizione in cui inserire un componente in un contenitore, ma solo in modo abbastanza indiretto
  - è possibile specificare la posizione di un componente in un contenitore, ma solitamente in modo relativo (ad esempio, dicendo se il componente va in alto oppure a destra) e non in modo assoluto (specificando le coordinate in cui andrà collocato il componente)
- la possibilità di collocare un componente in un contenitore è gestita mediante oggetti chiamati **gestori di layout** che sono associati ai contenitori

Gestori di layout

I gestori di layout sono rappresentati da sotto-classi della classe **LayoutManager** (del package **java.awt**), tra cui

- **FlowLayout**
  - i componenti sono disposti come un testo in una pagina, su linee che vanno da sinistra a destra
- **GridLayout**
  - i componenti sono disposti su una griglia rettangolare
- **BorderLayout**
  - il contenitore è organizzato in cinque aree (centro, alto, basso, sinistra e destra) e ciascun componente va esplicitamente disposto in una di queste cinque aree

Uso dei gestori di layout

I gestori di layout sono associati ai contenitori, e gestiscono l'inserimento di componenti nel contenitore

- in particolare, è possibile associare un layout manager al pannello usato come contenitore intermedio nell'interfaccia grafica
- l'associazione di un layout manager a un contenitore può avvenire con due modalità
  - al momento della creazione del contenitore, specificando il layout manager come argomento del costruttore del contenitore

```
jPanel = new JPanel( new BorderLayout() );
```

  - aggiungendo il layout manager al contenitore dopo la creazione del contenitore

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

FlowLayout

FlowLayout è il gestore di layout di default per un pannello, e dispone i componenti come un testo in una pagina, su linee che vanno da sinistra a destra

- una interfaccia con sei bottoni disposti usando **FlowLayout**

- effetti del ri-dimensionamento della frame

FlowLayout

```
/* Inizializza l'interfaccia dell'applicazione. */
private void inizializzaGUI() {
    /* pannello, usato come contenitore intermedio */
    JPanel jPanel;

    /* crea il pannello */
    jPanel = new JPanel();

    /* assegna il layout manager del pannello */
    jPanel.setLayout( new FlowLayout() );

    /* crea i componenti dell'interfaccia */
    jB1 = new JButton("Uno");
    jB2 = new JButton("Due");
    jB3 = new JButton("Tre");
    jB4 = new JButton("Quattro");
    jB5 = new JButton("Cinque");
    jB6 = new JButton("Sei");

    ... segue ...
```

FlowLayout

```
        /* aggiunge i componenti al pannello */
        jPanel.add(jB1);
        jPanel.add(jB2);
        jPanel.add(jB3);
        jPanel.add(jB4);
        jPanel.add(jB5);
        jPanel.add(jB6);

        /* aggiunge il pannello alla frame */
        this.getContentPane().add(jPanel);
    }
```

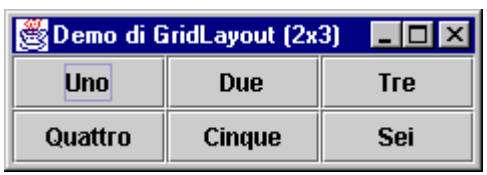
GridLayout

GridLayout dispone i componenti su una griglia rettangolare

- il numero di righe e colonne che compone la griglia va specificata al momento della creazione del **GridLayout**
- i componenti vanno inseriti, per righe, da sinistra verso destra
  - disposizione su una griglia 3 per 2

```
/* assegna il layout manager del pannello */
jPanel.setLayout( new GridLayout(3, 2) );
```

  - disposizione su una griglia 2 per 3



BorderLayout

BorderLayout dispone i componenti in cinque regioni

- le cinque regioni sono identificate dalle costanti **BorderLayout.NORTH**, **BorderLayout.SOUTH**, **BorderLayout.EAST**, **BorderLayout.WEST**, **BorderLayout.CENTER**
- la disposizione di un componente in una di queste cinque arre va specificata utilizzando una di queste cinque costanti come secondo argomento del metodo **add(...)**

- se più componenti vengono collocate nella stessa regione, possono sovrapporsi
  - nell'esempio, è il caso del bottone etichettato **"Cinque"**, coperto dal bottone etichettato **"Sei"**

BorderLayout

Codice per l'inizializzazione dell'interfaccia grafica

```
... crea il pannello ...

/* assegna il layout manager del pannello */
jPanel.setLayout( new BorderLayout() );

... crea i componenti dell'interfaccia ...

/* aggiunge i componenti al pannello */
jPanel.add(jB1, BorderLayout.NORTH);
jPanel.add(jB2, BorderLayout.EAST);
jPanel.add(jB3, BorderLayout.SOUTH);
jPanel.add(jB4, BorderLayout.WEST);
jPanel.add(jB5, BorderLayout.CENTER);
jPanel.add(jB6, BorderLayout.CENTER);
/* jB5 e jB6 nella stessa regione */

... aggiunge il pannello alla frame ...
```

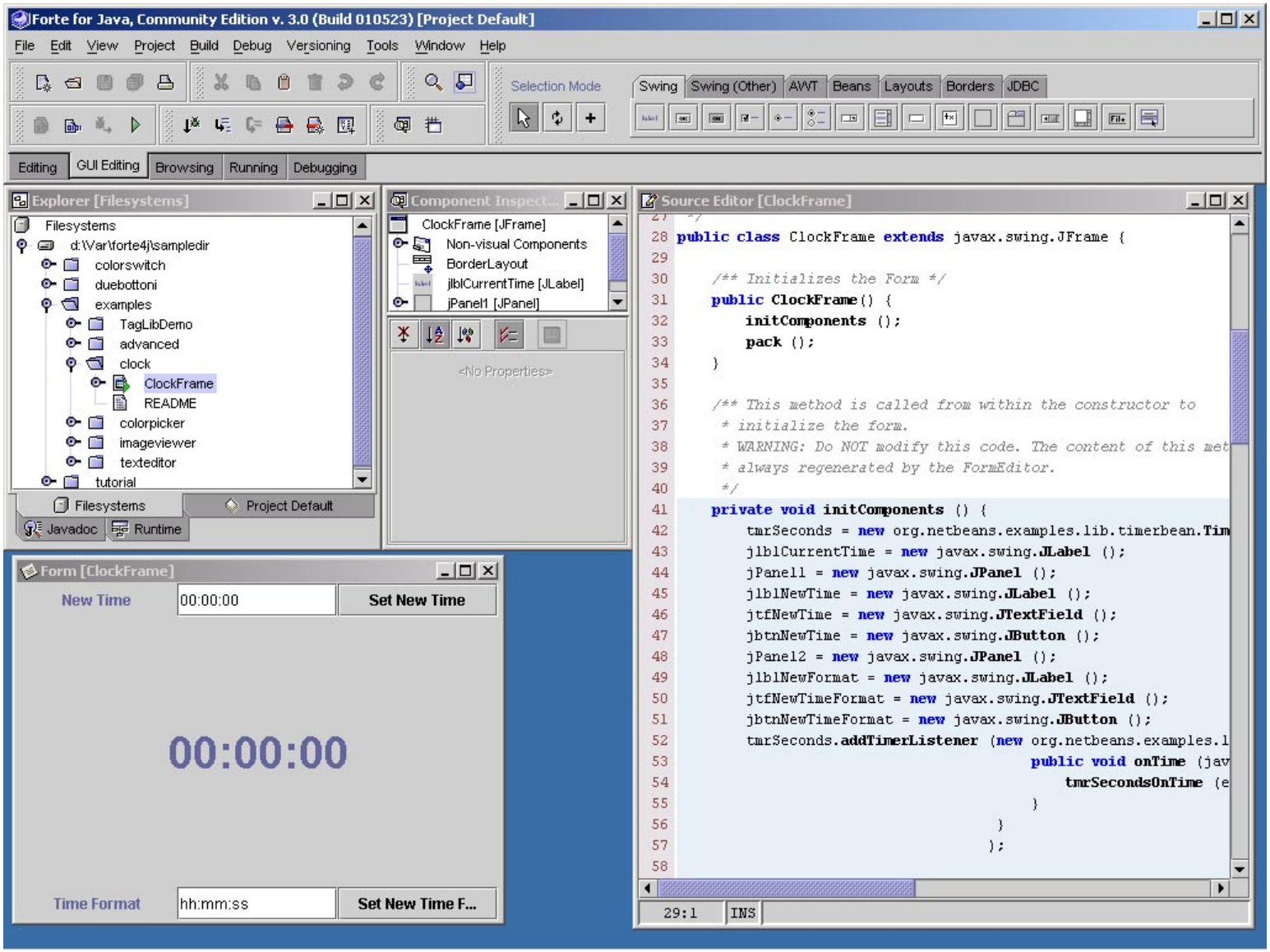
Interfacce grafiche e IDE

La realizzazione di una interfaccia grafica richiede normalmente la creazione e il posizionamento di numerosi componenti

- il programmatore dovrebbe ripetere, per ciascun componente
  - la dichiarazione della variabile per il componente
  - la creazione dell'oggetto che rappresenta il componente
  - l'inserimento del componente in un opportuno contenitore

Normalmente, questa attività non viene svolta dal programmatore mediante la scrittura diretta di codice, ma piuttosto vengono usati degli strumenti di programmazione (GUI editor) dedicati alla realizzazione di interfacce grafiche, disponibili in numerosi IDE

Esempio di GUI editor — Forte for Java



GUI editor

Un GUI editor dispone di numerose finestre, tra cui

- una finestra che visualizza l'interfaccia grafica che si sta realizzando (nella figura, in basso a sinistra)
  - il programmatore può selezionare componenti dalla barra degli strumenti del GUI editor (in alto a destra) e posizionarli in tale finestra
- una finestra che elenca i componenti dell'interfaccia grafica, con le loro proprietà (nella figura, al centro)
  - il programmatore può esaminare e modificare le proprietà dei componenti, ad esempio nome e testo di un bottone
- una finestra di editing, che mostra il codice per la realizzazione dell'interfaccia grafica (nella figura, sulla destra)
  - lo scopo del GUI editor è di generare automaticamente la parte standard del codice dell'interfaccia grafica (evidenziata da uno sfondo celeste anziché bianco)
  - il programmatore deve invece scrivere le porzioni di codice relative alla logica applicativa dell'applicazione