

Prefazione

Questo libro è un testo introduttivo di informatica, che focalizza l'attenzione sui principi della programmazione e sulla relativa pratica. Perché dovrete scegliere questo libro per il vostro primo corso di informatica? Eccovi le ragioni principali:

- Utilizzo un punto di vista che va al di là della sintassi del linguaggio e si concentra sui concetti dell'informatica.
- Insisto in modo particolare sul paradigma orientato agli oggetti, fin dal primo esempio, una versione orientata agli oggetti del tradizionale programma "Hello, World".
- Motivo gli studenti ad apprendere gli aspetti pratici della programmazione, con molti suggerimenti utili e un capitolo sul collaudo e la correzione degli errori.
- Presento un sottoinsieme accuratamente selezionato della libreria Java che sia comprensibile per i principianti ma abbastanza ricco da consentire la creazione di programmi interessanti.
- Uso il linguaggio, la libreria e gli strumenti Java standard, e non un ambiente specificamente dedicato al solo addestramento.
- Nei capitoli finali del libro, descrivo importanti tecniche per lo sviluppo di programmi di tipo client/server, come la programmazione di basi di dati, XML e Java Server Pages.

L'uso di Java

Questo libro si basa sul linguaggio di programmazione Java, che ho scelto per quattro motivi:

- Orientato agli oggetti
- Sicuro
- Semplice
- Dotato di un'ampia libreria standard

A tutt'oggi, il punto di vista orientato agli oggetti è il paradigma dominante nella progettazione del software. Io credo con grande convinzione che la programmazione orientata agli oggetti consenta agli studenti di trascorrere più tempo nella progettazione dei propri programmi e meno tempo nella scrittura e verifica del codice. In questo libro inizio a presentare oggetti e classi molto presto: gli studenti imparano a manipolare oggetti e a costruire semplici classi fin dal Capitolo 2.

Uso molto raramente metodi statici, oltre al metodo `main`. Come conseguenza, gli studenti pensano in termini di oggetti fin dall'inizio, senza dover trascorrere la seconda metà del corso a disimparare le cattive abitudini sviluppate nella prima metà.

Nella progettazione delle classi, tengo rigidamente separate le classi dai loro programmi di collaudo. (In realtà, se usate un ambiente di sviluppo come BlueJ, non avrete nemmeno bisogno dei programmi di collaudo. Questo libro non richiede che usiate BlueJ o alcun altro ambiente particolare, ma gli esempi funzionano particolarmente bene con BlueJ. Provatelo e potreste convertirvi anche voi: i miei studenti sono rimasti affascinati dalla possibilità di interagire in modo intuitivo con i loro oggetti.)

Un altro aspetto significativo di questo libro consiste nella trattazione delle interfacce prima delle sottoclassi. Questo ha un grande vantaggio: gli studenti vedono la potenza del polimorfismo prima di doversi preoccupare degli aspetti tecnici connessi all'ereditarietà fra le classi.

Ovviamente, ci sono molti linguaggi di programmazione orientati agli oggetti oltre a Java. In linea di principio, si potrebbe insegnare la programmazione orientata agli oggetti anche con il linguaggio C++, ma Java ha un vantaggio fondamentale rispetto al C++, la sua sicurezza. Gli studenti possono compiere (e in realtà compiono) un sorprendente numero di errori quando usano C++, molti dei quali portano a comportamenti del programma misteriosi e difficilmente riproducibili. Usando C++, un docente deve trascorrere molto tempo delle sue lezioni per spiegare le abitudini di programmazione sicura, oppure gli studenti svilupperanno una ben radicata scarsa confidenza nelle loro possibilità di programmazione, una situazione assai poco consona a un corso introduttivo.

Un altro grande vantaggio di Java è la sua semplicità. Sebbene non sia un obiettivo ragionevole insegnare tutti i costrutti di Java in un corso introduttivo, i docenti sono in grado di comprendere appieno tutti gli aspetti sintattici e semantici del linguaggio Java e possono rispondere alle domande degli studenti con piena consapevolezza. Al contrario, il linguaggio C++ è così complesso che ben poche persone sono in grado di capirne tutte le caratteristiche. Sebbene io abbia utilizzato C++ in modo intenso per più di una dozzina d'anni, rimango regolarmente di stucco di fronte a un principiante che mi mostra un messaggio d'errore del compilatore particolarmente criptico. La semplicità è importante, specialmente per un corso di base. Non è una buona idea scegliere come strumento di base un linguaggio di programmazione che gli studenti e i docenti non siano in grado di apprendere con totale sicurezza.

Infine, la libreria standard di Java ha una dimensione sufficiente per poter essere usata nella maggioranza dei corsi di un curriculum in informatica. La grafica, la costruzione di interfacce utente, l'accesso a basi di dati, la programmazione *multi-threading* e la programmazione di rete fanno parte della libreria standard, per cui le abilità di programmazione che gli studenti apprendono nel corso di base serviranno loro per tutto il corso di studi. Di nuovo, C++ fallisce in modo evidente a questo proposito.

Non esistono strumenti standard per nessuno degli aspetti di programmazione appena citati. Il sottoinsieme della libreria Java trattato in questo libro consente agli studenti di gestire un'ampia varietà di problemi di programmazione.

Una panoramica del libro

Il libro può essere suddiviso in quattro parti in modo molto naturale. La Figura 1 mostra le propedeuticità fra i capitoli.

Parte A

I Capitoli da 1 a 7 (unitamente al primo dei capitoli disponibili gratuitamente nel sito Internet dedicato al testo, <http://www.apogeeonline.com/libri/02024/allegati/>) trattano i fondamenti della programmazione orientata agli oggetti: oggetti, metodi, classi, variabili, tipi numerici, stringhe e strutture di controllo. Gli studenti imparano a costruire classi molto semplici nel Capitolo 2. Il Capitolo 7 affronta, invece, l'argomento della progettazione di classi in un modo più sistematico.

Iniziando dal Capitolo 7 uso un piccolissimo sottoinsieme della notazione UML, soltanto i diagrammi di classi e i quattro tipi di frecce per la dipendenza, l'implementazione di interfacce, l'ereditarietà e l'associazione diretta. Tale piccolo sottoinsieme è utile per visualizzare le relazioni tra le classi, ma minimizza quegli aspetti che i principianti ritengono essere complessi, come la scelta corretta tra associazione, aggregazione e attributi.

Tratto la grafica molto presto, nel Capitolo 4, perché molti studenti si divertono a scrivere programmi che tracciano disegni, e anche perché rettangoli, ellissi e linee sono buoni esempi di oggetti. Uso intensamente le classi per la "grafica 2D" del pacchetto `java.awt.geom`, invece degli obsoleti metodi procedurali della classe `Graphics`. Invocare `g.drawRect(x,y,w,h)` non è orientato agli oggetti. Manipolare oggetti geometrici è sia orientato agli oggetti sia divertente per gli studenti. Uso gli applet perché gli studenti li possono programmare con una piccola conoscenza tecnica.

Nonostante ciò, la trattazione della grafica è completamente facoltativa: tutto il materiale è stato accuratamente presentato in modo che si possano evitare tutti i capitoli che trattano di grafica e di interfacce utente grafiche.

Il capitolo integrativo disponibile online si occupa di collaudo e debugging, un argomento a cui viene, sfortunatamente, dato scarso respiro nella maggior parte dei libri di testo.

Come spiegato nel Capitolo 3, per leggere dati inseriti dall'utente potete usare la classe `JOptionPane` e la sua finestra di dialogo (anche in programmi eseguiti in una finestra di console), oppure potete usare un oggetto di tipo `BufferedReader`. Quest'ultima scelta obbliga a contrassegnare il metodo `main` con la clausola `throws IOException`, cosa che ritenevo inaccettabile finché non ho deciso di riorganizzare tutti i programmi in modo che il metodo `main` sia soltanto una classe di prova "usa e getta". Non mi sembra un problema se i metodi di una classe di prova lanciano eccezioni, in ogni caso al termine della seconda parte del libro gli studenti impareranno a gestire le eccezioni.

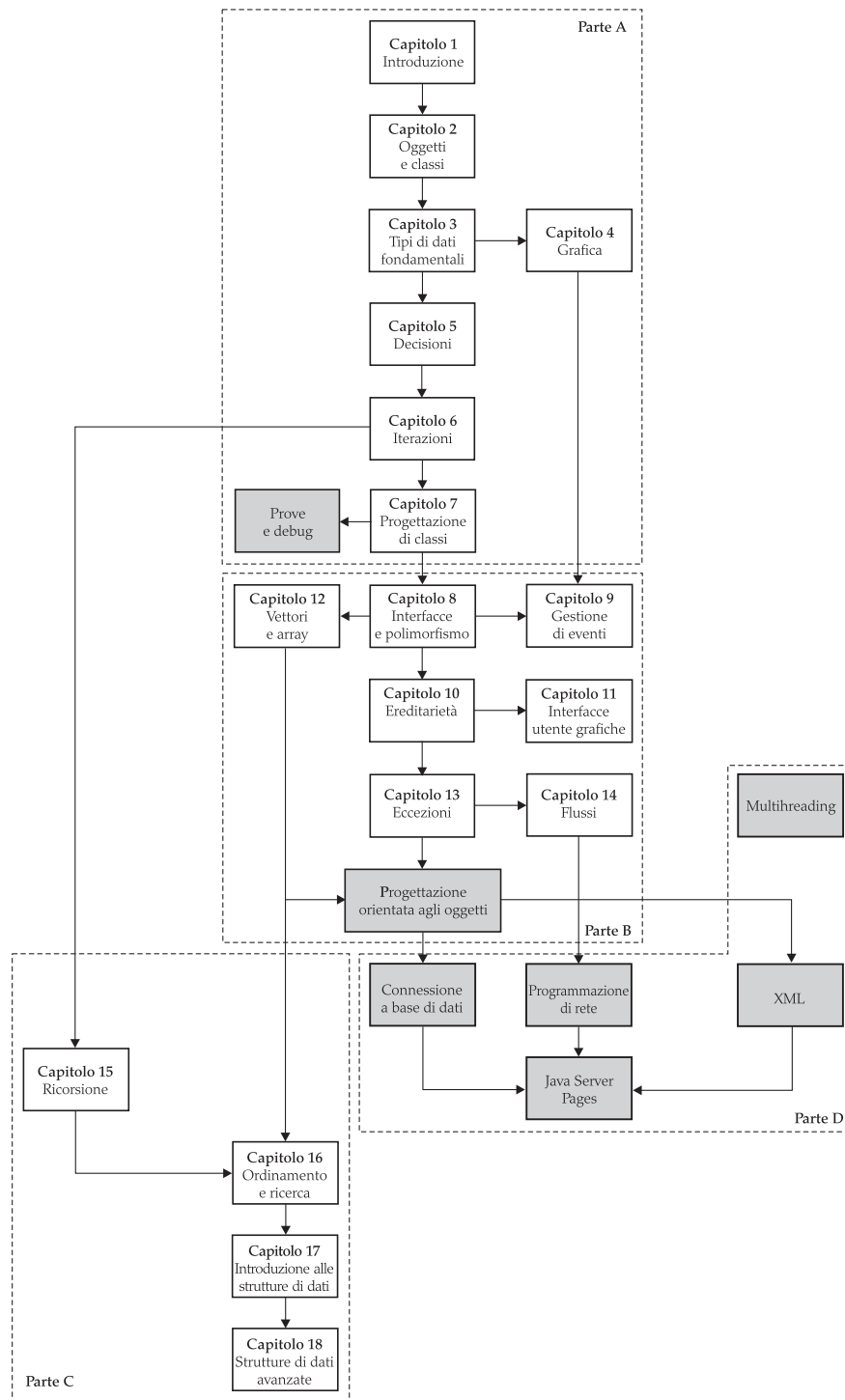


Figura 1
Dipendenze fra i capitoli

Parte B

I Capitoli da 8 a 14 (unitamente al secondo dei capitoli disponibili gratuitamente nel sito Internet dedicato al testo, <http://www.apogeeonline.com/libri/02024/allegati/>) trattano l'ereditarietà, gli array, le eccezioni, i flussi e, facoltativamente, la progettazione di interfacce utente grafiche (GUI).

La discussione sull'ereditarietà è suddivisa in due capitoli. Il Capitolo 8 tratta le interfacce e il polimorfismo, mentre il Capitolo 10 si occupa dell'ereditarietà vera e propria. L'introduzione delle interfacce prima dell'ereditarietà è utile per parecchie ragioni. Gli studenti vedono immediatamente il polimorfismo prima di dover affrontare la costruzione della superclasse, per cui diventa possibile presentare la programmazione e la gestione degli eventi fin da subito. In questo modo gli studenti sono condotti in modo naturale all'utilizzo di classi interne locali per i gestori di eventi, una tecnica più robusta della realizzazione di interfacce *ad hoc* per gli eventi che si trovano ancora in vecchi libri di testo.

La programmazione GUI è anch'essa suddivisa in due capitoli. Il Capitolo 9 si occupa della programmazione guidata dagli eventi, basandosi soltanto sulla nozione di interfaccia presentata nel Capitolo 8. Il Capitolo 11 tratta, invece, dei componenti GUI e della loro disposizione. Questo capitolo richiede alcune conoscenze di ereditarietà (l'estensione di `frame` e pannelli e l'invocazione di `super.paintComponent`), in ogni caso è possibile presentare questi due capitoli unitariamente, prima o dopo il Capitolo 10.

Voglio nuovamente puntualizzare che la trattazione di grafica e di interfacce utente grafiche è completamente facoltativa. Una possibile alternativa è la presentazione di grafica e applet (che sono abbastanza semplici da programmare), evitando GUI e gestione degli eventi.

In questo libro presento gli array e i flussi dopo l'ereditarietà. Da un punto di vista orientato agli oggetti, l'ereditarietà è un concetto cruciale e trovo che presentarlo il più presto possibile sia utile. Tuttavia, se preferite trattare prima gli array e i flussi, potete semplicemente scambiare i capitoli senza incorrere in alcun problema.

Ho preferito presentare le liste sequenziali (`ArrayList`) prima degli array, perché secondo la mia esperienza gli studenti trovano che i metodi `get` e `set` siano assai naturali, mentre sono sorprendentemente diffidenti nei confronti dell'operatore `[]`. Non mi sembra che siano preoccupati neppure dal cast che è necessario quando si usa il metodo `get`. Usando le liste sequenziali eviterete completamente gli array riempiti solo in parte: non ci si deve meravigliare se molti programmatori professionisti usano sempre le liste sequenziali (o i vettori) e ricorrono agli array solo di rado. Ovviamente, c'è bisogno degli array per i numeri, ma sequenze di numeri non sono poi così comuni nei programmi orientati agli oggetti.

Raccomando caldamente di presentare i flussi di oggetti e la serializzazione, soprattutto se il corso prevede esercitazioni di progettazione significative. Nella mia esperienza, gli studenti sono piacevolmente sorpresi dalla scoperta che è possibile memorizzare l'intero stato di una loro applicazione con una sola invocazione di `writeObject`, per poi recuperarlo in seguito altrettanto agevolmente.

Parte C

I Capitoli da 15 a 18 contengono un'introduzione agli algoritmi e alle strutture di dati, con la trattazione di ricorsione, ordinamento e ricerca, liste concatenate, alberi binari e tabelle hash.

Presentando la ricorsione, trovo che sia molto utile un punto di vista orientato agli oggetti. Nei miei esempi introduttivi, un oggetto che risolve un problema ricorsivamente costruisce un altro oggetto della stessa classe che risolve un problema più semplice. Fare in modo che questo nuovo oggetto svolga il compito più semplice è molto più plausibile per gli studenti rispetto a una funzione che invoca se stessa.

Ho inserito le strutture di dati nel contesto delle collezioni nella libreria standard di Java. Gli studenti impareranno le astrazioni essenziali della libreria standard (come: iteratori, insiemi, mappe) così come le prestazioni caratteristiche delle diverse collezioni. Tuttavia, una discussione dettagliata della realizzazione di strutture di dati è ben al di là degli scopi di questo libro.

Parte D

Questi cinque capitoli trattano tecniche avanzate di programmazione Java che spaziano senza alcun dubbio oltre il programma di un corso introduttivo di Java e sono consultabili *a pagamento* nel sito Internet dell'editore: <http://www.apogeeonline.com/Ebook>. Sebbene, come già detto, uno studio completo della libreria Java richiederebbe parecchi volumi, molti docenti ritengono che un libro di testo dovrebbe offrire agli studenti materiale di riferimento addizionale rispetto a quanto richiesto per il corso. Alcune strutture didattiche prevedono anche un secondo corso che tratta di aspetti di programmazione più tecnici, come la programmazione di rete e di basi di dati, piuttosto del più tradizionale materiale di approfondimento sulle strutture di dati e gli algoritmi. Personalmente, ritengo che questa sia una strada percorribile, per cui questo testo può essere usato per corsi prolungati che diano agli studenti una formazione preliminare sui fondamenti della programmazione, *unitamente* a un'ampia copertura di applicazioni. In alternativa, tale materiale può essere utile per i progetti degli studenti.

Selezionando gli argomenti per quest'ultima parte del libro, piuttosto di inserire materiale più approfondito sulla progettazione di interfacce utente, ho volutamente incluso quelle tecnologie che ritengo di particolare interesse per la *programmazione dei server*, come la programmazione di rete, le basi di dati, il linguaggio XML e Java Server Pages, nonché i servlet. La rete Internet ha reso possibile l'installazione di molte utili applicazioni su server, a cui spesso si accede semplicemente con un navigatore. Questo approccio incentrato sui server per lo sviluppo di applicazioni è stato in parte reso possibile dal linguaggio Java e dalle sue librerie, e oggi la maggior parte dell'uso industriale di Java riguarda la programmazione di server.

Appendici

L'appendice A1 contiene una guida allo stile usato per i programmi di questo libro: ho trovato molto utile richiedere uno stile coerente nelle esercitazioni. Altre appendici contengono interessante materiale di riferimento.

La struttura didattica

L'inizio di ciascun capitolo ha la consueta panoramica sugli obiettivi del capitolo stesso e le motivazioni introduttive. Nel corso del capitolo, note a margine mostrano i punti in cui vengono introdotti nuovi concetti. Tali note sono riassunte alla fine di ogni capitolo.

Esistono, poi, sei tipi di note per aiutare gli studenti, intitolate "Errori comuni", "Consigli per la produttività", "Consigli per la qualità", "Argomenti avanzati", "Note di cronaca" e "Consigli pratici". Queste note sono evidenziate in modo particolare per non interrompere il flusso del materiale principale. Mi aspetto che la maggior parte dei docenti trattino soltanto poche di queste note in aula, lasciando le altre per una lettura a casa. Alcune note sono piuttosto brevi, altre occupano più di una pagina: ho deciso di assegnare a ciascuna nota lo spazio necessario per una piena e convincente spiegazione, piuttosto che tentare di farle rientrare in "consigli" di un unico paragrafo.



■ Gli Errori comuni descrivono i tipi di errori compiuti spesso dagli studenti, con una spiegazione del motivo dell'errore e di come rimediare. La maggior parte degli studenti scopre rapidamente queste note e le legge per conto proprio.



■ I Consigli per la qualità illustrano buone abitudini di programmazione. Dato che molti di essi richiedono uno sforzo iniziale per l'apprendimento, queste note danno attente motivazioni per i consigli stessi e spiegano come lo sforzo verrà ripagato in seguito.



■ I Consigli per la produttività insegnano agli studenti come usare in modo più efficace gli strumenti che hanno a disposizione. Molti studenti principianti pongono poca attenzione al proprio modo di utilizzare i computer e il relativo software e spesso non hanno dimestichezza con i trucchi del mestiere, come le scorciatoie da tastiera, le ricerche e le sostituzioni globali o l'automazione di compiti frequenti mediante file di comandi.



■ Gli Argomenti avanzati trattano materiale non essenziale o più complesso. Alcuni di questi argomenti propongono costrutti sintattici alternativi che non sono necessariamente tecniche avanzate: in molti casi il libro usa un particolare costrutto del linguaggio, ma presenta eventuali alternative come Argomenti avanzati. Docenti e studenti dovrebbero sentirsi liberi di poter usare tali costrutti nei propri programmi, se li preferiscono. Tuttavia, la mia esperienza è che molti studenti gradiscono l'approccio più semplice, perché in tal modo si riduce il numero di decisioni che devono prendere.



■ Le Note di cronaca forniscono informazioni storiche e sociali sull'informatica, oltre a "pillole" di argomenti scientifici avanzati. Molti studenti leggeranno tali note per conto proprio, mentre fingeranno di seguire la lezione.



■ Come novità, in questa edizione sono state introdotte delle sezioni di Consigli pratici, ispirati dalle guide "HOWTO" di Linux. Queste note hanno lo scopo di rispondere a domande frequenti degli studenti, come "Cosa devo fare ora?", fornendo istruzioni sull'esecuzione dei compiti più comuni passo dopo passo.

