

Prefazione

Questo libro presenta un'introduzione all'informatica di tipo tradizionale utilizzando strumenti moderni. In quanto informatici, abbiamo la fortuna di poter iniziare gli studenti a un'attività che è accessibile, dà soddisfazione ed è profonda piuttosto che ampia, vale a dire l'attività della *programmazione*. Come la maggioranza degli informatici, sono convinto che la programmazione sia un tema centrale dell'informatica. Di conseguenza, questo corso insegna agli studenti come si programma.

Anche se questo libro ha una struttura tradizionale, si serve di tecniche moderne in tre modi diversi.

In primo luogo, utilizza un sottoinsieme strategico del linguaggio Java. Java ha molti vantaggi come linguaggio per l'insegnamento: ha una sintassi più semplice e più coerente di quella del C o del C++ e l'ambiente run-time è molto esigente per quanto riguarda il rispetto dei limiti degli array e la correttezza dei riferimenti. Tuttavia, Java ha molte più funzionalità del C o del Pascal e un corso di introduzione alla programmazione non può coprirle tutte. In questo libro non si accenna neppure al multithreading, ai riferimenti deboli o alla riflessione e per necessità si tratta soltanto una piccola parte della libreria standard, che ormai è arrivata a contenere migliaia di classi.

Il secondo aspetto moderno è l'uso degli oggetti fin dall'inizio. Gli oggetti vengono presentati per passi successivi. Nei capitoli 1 e 2, gli studenti imparano a *usare* gli oggetti, prendendo confidenza con i concetti di creazione degli oggetti e di chiamata dei loro metodi. Nel Capitolo 3, gli studenti imparano a implementare classi con metodi *molto semplici*. Il Capitolo 3 presenta giusto quel minimo di concetti sugli oggetti che è necessario per mettere gli studenti in grado di scrivere, nel Capitolo 4, propri programmi di grafica. Il Capitolo 7 è dedicato agli aspetti più tecnici della sintassi di Java, quali le variabili e i metodi statici. Il Capitolo 9 si occupa dell'ereditarietà e – in vista del capitolo sulla gestione degli eventi – delle interfacce. Infine, il Capitolo 14 insegna come *scoprire* le classi in una maniera sistematica. Qualche docente potrebbe chiedersi se “cominciare subito con gli oggetti” sia proprio la cosa migliore. Nel C++, la goffaggine della sintassi e delle regole del linguaggio rende poco attraente un approccio del genere, ma in Java le classi sono sufficientemente semplici da poter essere presentate fin dall'inizio. Inoltre, programmi Java che usano soltanto metodi statici hanno un aspetto ben strano e non sarebbe sensato avviare gli studenti a uno stile di

programmazione procedurale per poi insegnar loro ad abbandonarlo. Ciò nondimeno, nessuno si aspetta che gli studenti siano capaci di progettare fin dall'inizio complesse raccolte di classi. Questo libro procede secondo un approccio articolato in quattro passaggi:

1. Usare le classi.
2. Leggere implementazioni di classi semplici.
3. Implementare classi semplici.
4. Implementare raccolte di classi.

Questo approccio costruisce in modo graduale la progettazione orientata agli oggetti.

Per finire, questo libro presenta la grafica e le interfacce utente grafiche. Agli studenti piace programmare la grafica. A partire dalla release 2, Java dispone di funzionalità orientate agli oggetti per rendere la grafica; questi oggetti vengono introdotti fin dall'inizio. (Per esempio, invece di usare il metodo procedurale `drawRect`, gli studenti manipolano oggetti `Rectangle`.) Gli studenti imparano anche a costruire semplici interfacce utente usando il pacchetto Swing, che è disponibile in Java 2. L'accento è sulla semplicità, non sulla completezza. Per esempio, non vedo alcun senso nello sprecare tempo prezioso in aula insegnando le complessità del `GridBagLayout`, quando sono già tanti i concetti importanti da insegnare nel primo corso di informatica. I layout definiti in base al bordo, al flusso e alla griglia vanno altrettanto bene per creare semplici programmi. Il primo capitolo sulla grafica comincia con le applet, per aggirare le complessità della gestione degli eventi finestra. Le applicazioni di grafica sono presentate nel Capitolo 10 e la costruzione dell'interfaccia utente è l'argomento del Capitolo 12.

Tra i problemi che devono affrontare gli autori di libri di testo su Java, gli insegnanti e gli studenti c'è quello di come gestire la complessità della libreria Java. Sebbene il linguaggio di per sé sia semplice e pulito, la libreria delle classi è fastidiosamente piena di goffaggini per quel che riguarda l'input di testi, la gestione delle date, la chiusura dei programmi di grafica e così via. Il fatto è che Java non è stato concepito come linguaggio per l'insegnamento e nessuno si è mai preoccupato di nascondere certe minuzie di implementazione. Inoltre, alcuni pacchetti di librerie sono stati messi insieme in gran fretta, senza tener conto in alcun modo dei principi di una buona progettazione. Per la prima edizione di questo libro avevo preparato una libreria a "scatola nera", che non faceva vedere agli studenti i particolari di minore importanza, in modo che si potesse dedicare più tempo in aula a insegnare concetti fondamentali, piuttosto che perdersi nelle minuzie di Java. Quell'approccio piacque meno di quanto sperassi. Studenti e docenti avevano l'impressione di star sprecando del tempo prezioso imparando una libreria che in seguito non avrebbero mai più usato. Ancora più importante, temevano che si sarebbero trovati completamente persi una volta passati al corso successivo, per il quale quella libreria non era più disponibile. Alcuni dei problemi più fastidiosi della libreria Java standard sono stati eliminati in Java 2. Questi miglioramenti mi hanno consentito di abbandonare la libreria di grafica e di affidarmi ai pacchetti Java standard per tutti i programmi di grafica. Hanno reso inoltre possibile incapsulare le complessità dell'input da console in una classe semplice, che gli studenti possono capire. Gli studenti possono liberamente portare quella classe – o anche semplicemente la conoscenza di come implementarla – al loro prossimo corso di programmazione.

Nel programmare un corso, dovrete prendere le seguenti decisioni:

- ◆ Volete insegnare la grafica e le interfacce utente grafiche? Se non è vostra intenzione farlo, omettete i capitoli 4, 10 e 12. Questo non ha alcun riflesso sul resto del corso.
- ◆ Volete insegnare come prima cosa le classi oppure preferite trattare le diramazioni e i cicli prima delle classi? In questo secondo caso, presentate i capitoli 5 e 6 prima del Capitolo 3, con qualche piccola messa a punto in un paio di programmi esemplificativi.
- ◆ Volete insegnare i particolari della classe del lettore di console? Se non volete farlo, saltate i paragrafi 2.8 e 3.10.
- ◆ Quali capitoli facoltativi volete presentare? I capitoli, 8, 14, 15 e 16 su prove e debug, progettazione orientata agli oggetti, algoritmi e strutture di dati sono autonomi e potete includerli o escluderli in base al tempo che avrete a disposizione.

La Figura 1 mostra le correlazioni fra i capitoli.

Il materiale di ciascun capitolo è suddiviso in tre parti: le cose essenziali, quelle utili e quelle facoltative. Per presentare il materiale essenziale, basta saltar via tutte le note a margine (i paragrafi evidenziati con un motivo grafico sul bordo sinistro). È del tutto ragionevole ignorare completamente le note a margine durante le lezioni e assegnarle come compito di lettura a casa.

Tre insiemi di note a margine sono utili per gli studenti, in particolare quelle intitolate “Errori comuni”, “Consigli per la produttività” e “Suggerimenti per la qualità”. Gli studenti scoprono subito gli Errori comuni e li leggono per conto loro. Dovreste incoraggiare gli studenti a leggere i Suggerimenti per la qualità. Quanto ai Consigli per la produttività, potrebbero essere un po’ ardui per gli studenti più deboli, ma quelli che hanno maggior esperienza di computer di solito li trovano molto utili.

Le Note di cronaca e gli Argomenti avanzati sono facoltativi. Le Note di cronaca forniscono informazioni di carattere storico e sociale sull’informatica, come vengono richieste per soddisfare i requisiti di “contesto storico e sociale” stabiliti nelle linee guida della Association for Computing Machinery per i programmi di studio. La maggior parte degli studenti finirà per leggere di sua iniziativa le Note di cronaca, facendo finta di seguire le lezioni. Gli Argomenti avanzati trattano per lo più temi non essenziali o più difficili, come per esempio costrutti sintattici alternativi.

In molti casi, il libro usa un solo specifico costrutto del linguaggio, ma spiega quelli alternativi negli Argomenti avanzati. Insegnanti e studenti possono liberamente usare questi costrutti nei loro programmi, se lo preferiscono. L’esperienza, però, mi ha insegnato che molti studenti apprezzano l’approccio più semplice, perché riduce notevolmente il numero delle decisioni che devono prendere.

L’Appendice A1 contiene una guida per lo stile da usare con questo libro. Ho trovato molto vantaggioso esigere uno stile coerente per tutti i compiti in classe e a casa. Se questa guida per lo stile dovesse essere in contrasto con l’orientamento dell’insegnante o con le abitudini locali, può essere modificata. Proprio a questo scopo, la guida per lo stile è disponibile in formato elettronico nel booksite abbinato a questo libro.

L’Appendice A2 contiene un riepilogo del sottoinsieme API Java usato in questo libro.

L’Appendice A3 contiene le tabelle dei sottoinsiemi Basic Latin (noto anche come ASCII) e Latin-1 dell’Unicode.

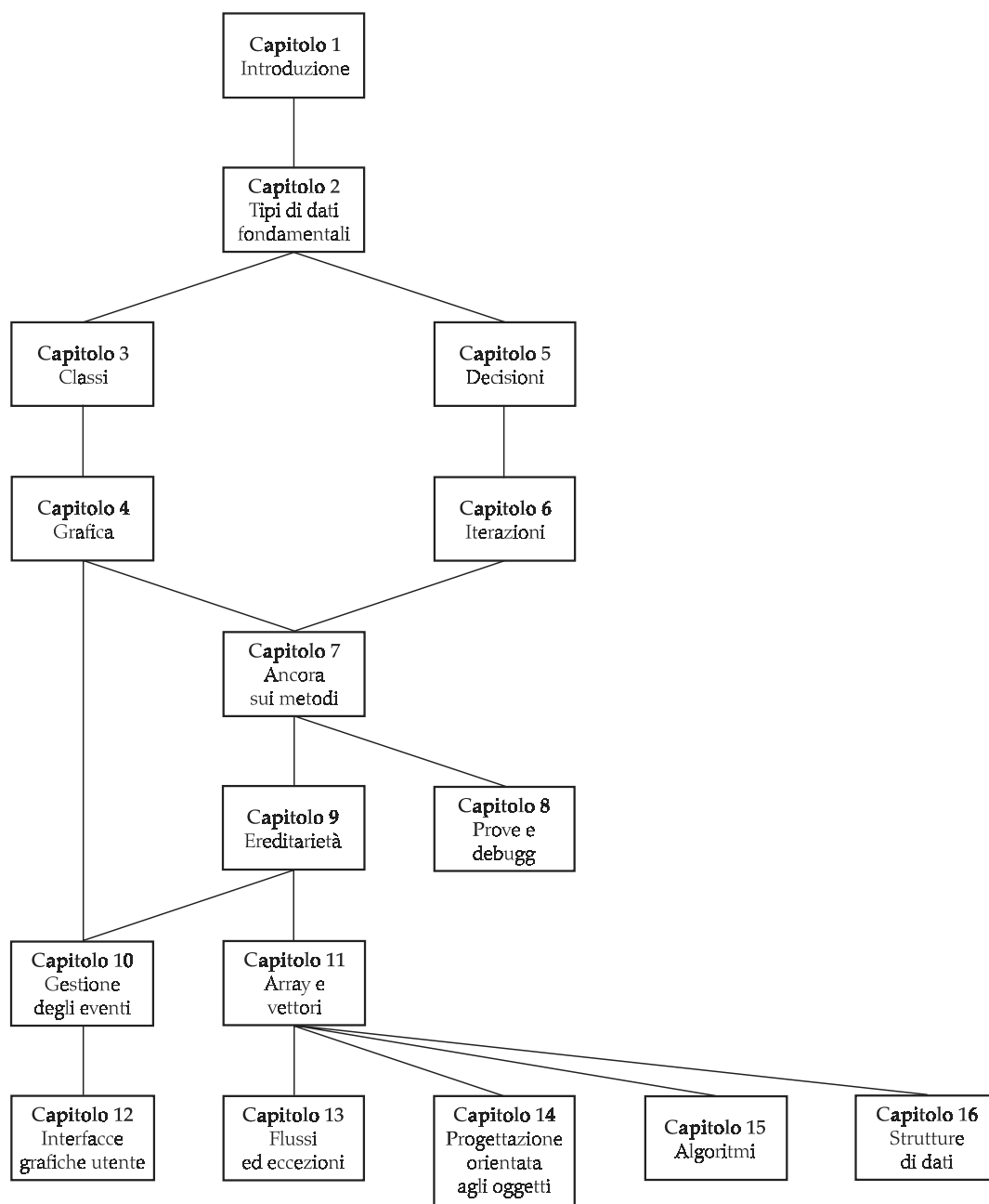


Figura 1
Correlazioni
fra i capitoli

Il libro tratta le seguenti unità conoscitive previste dalla guida per i programmi di studio della Association for Computing Machinery:

AL1: Elementi di strutture dei dati (6 ore su 13)

AL2: Tipi di dati astratti (2 ore su 3)

AL3: Ricorsione (2 ore su 3)

AL6: Ordinamento e ricerca (2 ore su 6)

PL3: Rappresentazione dei tipi di dati (2 ore su 2)

PL4: Controllo della sequenza (2 ore su 4)

PL5: Controllo dei dati, condivisione e verifica dei tipi (2 ore su 4)

PR: Introduzione a un linguaggio di programmazione (12 ore su 12)

SE1: Concetti fondamentali della risoluzione dei problemi (16 ore su 16)

SP1: Contesto storico e sociale dell'informatica (3 ore su 3)

Ulteriori risorse per docenti e studenti sono disponibili nel bokksite abbinato a questo libro, all'indirizzo URL:

www.apogeonline.com/education/booksite

Proprietà intellettuale delle immagini

Capitolo 1

Figura 1 e 2: Concessa da Intel. *Figura 3:* Concessa da Lisa Passmore. *Figura 4:* Concessa da Seagate. *Figura 5:* Concessa da Iomega. *Figura 6:* Concessa da Toshiba/The Benjamin Group. *Figura 7:* Concessa da Maynard Electronics. *Figura 8:* Concessa da International Business Machines Corporation. *Figura 9:* Concessa da Intel. *Figura 11:* Sperry Univac, Divisione della Sperry Corporation.

Capitolo 3

Figura 4: Concessa da International Business Machines Corporation.

Capitolo 4

Figura 13: Concessa da SAS Institute, Inc. *Figura 14:* Concessa da Autodesk, Inc. *Figura 15:* M. Tchervkf/The Image Bank.

Capitolo 5

Figura 5: Concessa da Digital Equipment Corporation, Corporate Photo Library. *Figura 6:* © Sun Microsystems.

Capitolo 8

Figura 2: Naval Surface Weapons Center.