Dai fondamenti agli oggetti
Corso di programmazione Java
Terza edizione
# Sommario

## 1 Computer, algoritmi e linguaggi
1.1 Algoritmi ......................................................... 1
1.2 Programmi e linguaggi di programmazione .................. 2
1.3 Informazione ..................................................... 3
1.4 Computer e programmazione, hardware e software ....... 3
1.5 La macchina di von Neumann ................................ 5
1.6 Dal linguaggio macchina ai linguaggi ad alto livello .... 9
1.7 Strumenti per la stesura dei programmi .................. 12
1.8 Java Virtual Machine ............................................ 13
1.9 Programmazione strutturata .................................... 15
1.10 Variabili e assegnamenti ....................................... 19
1.11 Strutture di controllo fondamentali: esempi ............. 23
1.12 Entità interagenti e oggetti: alcune idee ................. 30
1.13 Grammatiche ...................................................... 32
1.14 La grammatica del linguaggio Java: il lessico .......... 34
1.15 Il primo programma Java ....................................... 36

## 1 Uso degli oggetti

## 2 Protocolli e contratti
2.1 Primi esempi di oggetti e classi in Java .................. 44
2.2 Prototipi e segnature ............................................ 52
2.3 La classe String .................................................. 56
2.4 Variabili e tipi .................................................... 63

## 3 Basi del linguaggio
3.1 La classe Frazione ................................................ 73
3.2 L’istruzione if-else .............................................. 77
Sommario

3.3 Il tipo primitivo boolean

3.4 Operatori booleani e condizioni

3.5 I cicli while e do...while

3.6 L’istruzione for

3.7 Le istruzioni break e continue

4 Tipi primitivi e tipi enumerativi

4.1 Epressioni

4.2 Riepilogo degli operatori

4.3 Tipi numerici interi

4.4 Tipi numerici in virgola mobile

4.5 Conversioni implicite ed esplicite di tipo

4.6 Il tipo char

4.7 I tipi enumerativi

4.8 L’istruzione switch

4.9 I metodi statici

4.10 Le classi involucro

5 Array e collezioni

5.1 Array di oggetti

5.2 Array e cicli for

5.3 Il parametro del metodo main

5.4 Array di tipo primitivo

5.5 Array e tipi enumerativi

5.6 Array di array

5.7 La classe Sequenza: introduzione ai tipi generici

6 Uso della gerarchia

6.1 La classe Rettangolo

6.2 UML: rappresentazione di classi e oggetti

6.3 La classe Quadrato

6.4 Ereditarietà e polimorfismo

6.5 Le classi astratte

6.6 La gerarchia delle classi

6.7 Gerarchia e uso dei riferimenti

6.8 Scelta del metodo da eseguire

6.9 Esempio: gestione di un elenco di figure

6.10 I file di testo

6.11 Esempio: la tavola delle occorrenze

6.12 Le interfacce Java

6.13 L’interfaccia Iterable e il ciclo for–each

6.14 La gerarchia dei tipi

6.15 Gerarchia dei tipi e tipi generici
Sommario vii

6.16 Vincoli sui segnaposto ........................................ 279
6.17 Tipi generici e vincoli sugli argomenti ........................ 284
6.18 Tipi generici e compilazione ................................. 285

II Implementazione degli oggetti 287

7 Implementazione delle classi 289
7.1 Classi e oggetti .................................................. 289
7.2 La classe Frazione: alcuni miglioramenti .................... 300
7.3 Una nuova implementazione della classe Frazione .......... 303
7.4 Esempio: la classe Orario ................................. 309
7.5 I campi e i metodi statici ...................................... 316
7.6 Riepilogo della struttura delle classi ...................... 322
7.7 Implementazione di un’interfaccia ............................ 327
7.8 Documentazione delle classi ................................... 329
7.9 I package ......................................................... 337
7.10 I modificatori di visibilità public e “amichevole” ......... 342
7.11 Documentazione dei package .................................. 346
7.12 UML: membri di una classe .................................. 347

8 Estensione delle classi 349
8.1 Ereditarietà e implementazione di sottoclassi .............. 349
8.2 Costruttori e gerarchia delle classi .......................... 352
8.3 Il riferimento super ............................................ 355
8.4 Ereditarietà e stato degli oggetti ............................. 356
8.5 Overloading dei metodi .......................................... 362
8.6 Overriding, overloading e scelta del metodo da eseguire 363
8.7 Il metodo equals ................................................ 370
8.8 Variabili e adombramento ..................................... 375
8.9 Esempio .......................................................... 378
8.10 Implementazione della classe Figura ......................... 382
8.11 Implementazione della classe Rettangolo ................. 383
8.12 Nuovi metodi per le classi Rettangolo e Quadrato ....... 386
8.13 Il modificatore di visibilità protected ...................... 391
8.14 Il modificatore final ........................................... 394

9 Tipi enumerativi, tipi generici e interfacce 401
9.1 Definizione di tipi enumerativi ................................ 401
9.2 Definizione di classi generiche .............................. 409
9.3 Metodi generici .................................................. 413
9.4 Definizione di interfaccia .................................... 416
9.5 Uso del supertipo definito dall’interfaccia .................. 417
10 Organizzazione della memoria e ricorsione 427
  10.1 Invocazione di metodi, passaggio di parametri e rientro .................. 427
  10.2 Organizzazione della memoria .............................................. 431
  10.3 Metodi ricorsivi ............................................................. 438
  10.4 Metodi con un numero variabile di argomenti ............................... 450

11 Eccezioni 453
  11.1 Le eccezioni ................................................................. 453
  11.2 Come intercettare le eccezioni: l’istruzione try-catch ..................... 457
  11.3 Rientro dai metodi ed eccezioni .......................................... 460
  11.4 Esempio: una calcolatrice in notazione postfissa ........................... 461
  11.5 Alcune eccezioni ............................................................ 479
  11.6 Come sollevare le eccezioni: l’istruzione throw ............................ 485
  11.7 Come definire un’eccezione .................................................. 490
  11.8 Eccezioni controllate e non controllate .................................... 500
  11.9 Perché le eccezioni controllate? ............................................ 511
  11.10 Gli errori ........................................................................... 512
  11.11 La clausola finally .............................................................. 513
  11.12 Ridefinizione di metodi ed eccezioni ....................................... 514
  11.13 Metodi astratti ed eccezioni ................................................... 517

III Argomenti avanzati 519

12 Strutture dati dinamiche 521
  12.1 Implementazione di strutture a pila .......................................... 521
  12.2 Le code ............................................................................. 530
  12.3 Le liste ordinate ................................................................. 539
  12.4 Alberi binari ...................................................................... 554
  12.5 Alberi di ricerca ................................................................. 556
  12.6 Implementazione degli alberi di ricerca in Java ............................... 558
  12.7 Esempio: elenco alfabetico di stringhe ....................................... 565
  12.8 Ulteriori esempi sulle liste ..................................................... 571
  12.9 Ulteriori esempi sugli alberi ..................................................... 574

13 Gli stream 579
  13.1 Stream di caratteri ................................................................ 580
  13.2 Le gerarchie Reader e Writer .................................................. 584
  13.3 La classe File ..................................................................... 590
  13.4 Stream di byte ................................................................... 595
  13.5 Lettura e scrittura di dati primitivi .......................................... 596
  13.6 La classe PrintStream ........................................................... 598
  13.7 I flussi di input/output standard .............................................. 598
13.8 Lettura e scrittura di oggetti su stream di byte ........................................ 601

IV Appendici .................................................................................................................. 609

A Tipi primitivi .................................................................................................................. 611
  A.1 Tipi interi .................................................................................................................. 612
  A.2 Il tipo char ................................................................................................................ 613
  A.3 Operatori .................................................................................................................... 614

B Strutture di controllo ..................................................................................................... 615
  B.1 L’istruzione if-else .................................................................................................... 615
  B.2 L’istruzione while ...................................................................................................... 616
  B.3 L’istruzione do...while ............................................................................................. 617
  B.4 L’istruzione for ......................................................................................................... 617
  B.5 L’istruzione switch ................................................................................................... 619
  B.6 Le istruzioni break e continue .................................................................................. 620
  B.7 L’istruzione return .................................................................................................... 621
  B.8 L’istruzione try-catch ............................................................................................... 622

C Installazione di JDK 6 e del package prog .................................................................. 625
  C.1 Installazione .............................................................................................................. 625
  C.2 Prima esecuzione ........................................................................................................ 631
  C.3 I package .................................................................................................................... 634
  C.4 Installazione del package prog .................................................................................. 637

Note bibliografiche ............................................................................................................ 639

Indice analitico ................................................................................................................... 641
Prefazione

La programmazione consiste fondamentalmente nel codificare il procedimento risolutivo di un problema (cioè un algoritmo) in un insieme di istruzioni destinate a un esecutore, creando appunto il programma.

Agli inizi i programmi venivano scritti direttamente nel linguaggio comprensibile al processore cui erano destinati; con l’introduzione dei linguaggi di programmazione si è passati, via via, alla scrittura di programmi destinati a esecutori più evoluti, raggiungendo livelli di astrazione sempre più elevati. Tutto ciò ha modificato profundamente il modo di scrivere i programmi, al punto che, attualmente, gran parte dell’attività di programmazione consiste nel combinare tra loro “mattoni” già pronti. Questo, sebbene a prima vista possa apparire banale, richiede un maggiore livello di astrazione e di flessibilità. Il programmatore deve essere in grado di adattarsi a ciò che ha a disposizione: deve capire come sono fatti i mattoni che può utilizzare e deve avere l’abilità di combinarli in modo adeguato per ottenere la soluzione desiderata. Di volta in volta i mattoni possono cambiare: oltre che dalle librerie standard, ampiamente disponibili, essi possono provenire, ad esempio, da codice specializzato per il控制 di particolari dispositivi o per la soluzione di una determinata classe di problemi, oppure da codice scritto da colleghi nell’ambito di uno stesso progetto.

Un altro aspetto importante, strettamente legato all’evoluzione dei mezzi di calcolo, è che, in molti casi, lo svolgimento di un compito non dipende da un unico esecutore, ma è il risultato delle interazioni di diversi agenti, come del resto avviene nel mondo reale. Ad esempio, nella visualizzazione di una pagina web intervengono almeno due agenti: il browser, cioè il programma utilizzato per “navigare” nella rete, e il server su cui risiede la pagina. La visualizzazione della pagina è frutto della loro interazione (è facile osservare che, in realtà, gli agenti coinvolti in questa operazione sono molti di più).

Questo testo è un vero e proprio corso di programmazione, sviluppato facendo riferimento alle idee appena esposte. Il corso è accessibile anche a chi non ha alcuna conoscenza della materia, e nel contempo tocca e approfondisce le tematiche e gli aspetti fondamentali della programmazione.

Il paradigma a oggetti rappresenta l’ambiente ideale per avvicinarsi alla programmazione nell’ottica descritta sopra. Gli oggetti non sono altro che i mattoni base con cui costruire i programmi. La computazione è affidata a vari oggetti che possono lavorare autonomamente e cooperare tra loro. Ogni oggetto è in grado di fornire un insieme di servizi. Compito principale
del programmatore è quello di coordinare tra loro gli oggetti al fine di ottenere il comportamento desiderato. Pertanto, il testo fa ampio riferimento a questo paradigma, senza tuttavia trascurare i tradizionali argomenti relativi alla programazione con linguaggi imperativi.

Per varie ragioni, tra cui la relativa semplicità e la vasta diffusione, abbiamo scelto di concretizzare i concetti presentati nel libro tramite il linguaggio Java. Tali concetti costituiscono comunque un bagaglio sicuramente utile al lettore per affrontare lo studio successivo di linguaggi differenti, sia imperativi sia a oggetti. Infatti, lo scopo del testo non è insegnare un linguaggio di programmazione, ma piuttosto insegnare la programmazione utilizzando come ausilio un linguaggio adeguato, nel nostro caso Java.

 Tradizionalmente l’introduzione della programmazione a oggetti avviene senza insistere troppo sulla differenza tra l’uso delle classi e la loro implementazione. Quasi tutti i testi, fin dai primi capitoli, trattano subito l’implementazione delle classi. Questo approccio, che abbiamo sperimentato anche noi in passato, concentra l’attenzione principalmente sui dettagli implementativi, facendo perdere di vista i differenti livelli di astrazione. Di fronte a esercizi in cui si richiede l’uso di classi già esistenti, molti studenti tendono a riscrivere tutto ex novo, un po’ come se per ascoltare la musica da una radio la si aprisse con un cacciavite. Da queste considerazioni è nata la scelta, che caratterizza e differenzia notevolmente questo testo dai numerosi altri passati per le nostre mani, di suddividere i contenuti in due parti fondamentali, la prima relativa all’uso, la seconda all’implementazione delle classi. Riteniamo che in questo modo, tenendo ciò separati i due aspetti e chiarrendone i diversi livelli di astrazione, si riesca a enfatizzare la parte orientata agli oggetti senza trascurare le parti imperative.

Nella prima parte del testo, il lettore impara a individuare classi e oggetti utili alla soluzione di determinati problemi, e a combinarli tra loro usando quanto è già a disposizione nelle librerie. In pratica, il lettore imparerà a risolvere problemi, anche complessi, scrivendo brevi e semplici programmi che sfruttino il più possibile codice già esistente. Come evidenziato nel Sommario, vengono introdotti in questa parte concetti fondamentali, sia legati alla programmazione tradizionale sia a quella a oggetti, tra cui i concetti di variabile e tipo e le strutture di controllo fondamentali. L’aspetto particolarmente innovativo e originale del testo, cioè il livello di astrazione, è subito evidente: anziché limitarsi a mostrare come combinare le istruzioni base del linguaggio mediante le strutture di controllo, insistiamo fin dai primi esempi su come combinare i servizi forniti dalle classi e dagli oggetti. Questo ha il vantaggio di consentire, da subito, la codifica di programmi in grado di svolgere compiti complessi, ma soprattutto abituar immediatamente il lettore a pensare in astratto al comportamento degli oggetti, cioè a ciò che essi fanno, senza soffermarsi sui dettagli implementativi, rimandati alla parte successiva del testo. In questa prima parte introduciamo dal punto di vista dell’uso anche i concetti fondamentali della programmazione a oggetti, come l’ereditarietà, la gerarchia delle classi, le classi astratte e il polimorfismo. La loro presentazione risulta così assai semplificata.

La seconda parte del testo è destinata invece al progetto e all’implementazione delle classi e degli oggetti. Dopo avere acquisito, nella prima parte, l’abilità di utilizzare classi già pronte, il lettore imparerà a implementarne di nuove, che potranno essere utilizzate nei suoi programmi o in programmi scritti da altri. In questa parte vengono sviluppati in dettaglio svariati argomenti, tra cui i concetti di overloading e overriding e le interfacce. Un intero capitolo è inoltre dedicato
al meccanismo delle eccezioni, alla loro definizione, trattamento e uso per la stesura di codice semplice ed elegante che gestisca le situazioni anomale. Abbiamo scelto di inserire un capitolo relativo all’organizzazione della memoria durante l’esecuzione, argomento spesso trascurato, ma estremamente formativo in un corso di programmazione.

Il testo contiene anche una breve introduzione alla notazione UML (Unified Modeling Language) e diversi approfondimenti. In particolare, abbiamo riservato un intero capitolo alle strutture dati dinamiche e alla loro implementazione in Java. In un capitolo preliminare trattiamo l’evoluzione della programmazione dai linguaggi macchina ai linguaggi a oggetti passando per la programmazione strutturata.

I concetti sono illustrati per mezzo di numerosi esempi. Molto spesso l’estensione di un esempio è lo spunto per l’introduzione di un nuovo concetto. In ogni caso, dopo una prima esemplificazione, ogni concetto viene approfondito e studiato nei dettagli, ricorrendo, quando è necessario sottolineare particolari aspetti, a ulteriori esemplificazioni.

Poiché l’apprendimento della programmazione non è un’attività mnemonica, ma richiede lo sviluppo di abilità acquisibili solo con esperienza e applicazione, il testo comprende numerosi esercizi, indispensabili e insostituibili strumenti di apprendimento.

Il CD-ROM distribuito con il libro contiene alcune librerie da affiancare a quelle standard di Java, utilizzate principalmente per gli esempi e per gli esercizi, oltre al codice sorgente dei principali esempi. Il CD-ROM include anche l’ambiente Java Standard Edition Development Kit 6 di Sun Microsystems, per i sistemi operativi Windows e Linux.

Il testo è stato ideato per un corso universitario di programmazione accessibile a chi non conosce l’argomento. In particolare, i contenuti si sono evoluti e consolidati negli anni sulla base di un uso sperimentale di versioni preliminari nell’ambito dei corsi integrati di “Programmazione” e “Laboratorio di Programmazione” che teniamo presso i corsi di Laurea in Informatica dell’Università degli Studi di Milano e dell’Università degli Studi dell’Insubria.

Novità del linguaggio e novità del testo

Dalla sua nascita, alla metà degli anni ’90, l’ambiente Java ha subito una continua evoluzione che ha riguardato sia il linguaggio stesso, sia l’infrastruttura associata (librerie, Java Virtual Machine, ecc.).

Sicuramente, le novità più rilevanti per il linguaggio sono quelle introdotte a partire dalla versione 5, rilasciata da Sun Microsystems il 30 settembre 2004. Tra queste vi sono in particolare, i tipi enumerativi, l’autoboxing e unboxing, i tipi generici e il ciclo for-each. Java 6 (ultima versione disponibile nel momento in cui scriviamo), al contrario, non ha visto cambiamenti nel linguaggio, ma aggiornamenti nell’infrastruttura.

La seconda edizione del testo era nata dalla necessità di presentare le nuove importanti caratteristiche introdotte con Java 5. L’organizzazione generale e, soprattutto, la metodologia di esposizione degli argomenti sono rimaste quelle della prima edizione. I contenuti sono stati aggiornati integrandoli con le nuove caratteristiche del linguaggio. Nei casi più semplici, l’integrazione è avvenuta aggiungendo paragrafi ad hoc, più spesso abbiamo dovuto riscrivere alcune parti che, alla luce delle novità introdotte, risultavano obsolete. Nella terza edizione, allineata a
Java 6, il materiale è stato ulteriormente rivisto e raffinato: sono stati inseriti ulteriori esercizi, rielaborate e aggiornate le librerie utilizzate negli esempi e fornite nel CD-ROM allegato al testo.

Anche se i tipi generici non sono più una novità, come lo erano per l’edizione precedente, riteniamo utile sottolineare la loro importanza, non solo all’interno del linguaggio stesso, ma, e soprattutto, a livello concettuale, con le loro contrastanti caratteristiche di facilità e di difficoltà.

I tipi generici permettono di realizzare, con estrema facilità e con una conoscenza elementare del linguaggio, programmi che manipolano strutture dati complesse. Pensiamo ad esempio a una struttura a pila. Nelle versioni di Java precedenti era disponibile la classe Stack per rappresentare pile di oggetti (istanze della classe Object). Per scrivere applicazioni che utilizzassero pile contenenti dati di un tipo particolare, ad esempio pile di Integer, era necessario conoscere i concetti fondamentali relativi all’ereditarietà e alla gerarchia e ricorrere all’operazione, molto spesso “mal digerita”, di cast tra tipi riferimento.

Dalla versione 5 del linguaggio (al prezzo di una sintassi piuttosto barocca) è possibile costruire direttamente pile di Integer, scrivendo il codice che le utilizza senza necessità di conoscere i concetti relativi alla gerarchia delle classi e dei tipi. Questa caratteristica, che fornisce al linguaggio un ulteriore meccanismo di astrazione, costituisce una notevole semplificazione nell’uso del linguaggio e si accorda proprio alla filosofia di riuscire a svolgere compiti complessi con pochi strumenti. In quest’ottica abbiamo introdotto i tipi generici nel Capitolo 5.

Come tutti gli altri tipi riferimento, anche i tipi generici si collocano all’interno della gerarchia dei tipi. Inoltre, per alcuni di essi, il livello di genericità deve essere necessariamente limitato. Mentre, ad esempio, è facile pensare a una pila di oggetti di un qualunque tipo, per le sequenze ordinate dovremmo limitarci a considerare oggetti tra loro confrontabili secondo una relazione d’ordine. Per modellare queste caratteristiche sono stati introdotti i segnaposto (wildcard) e i vincoli su di essi. Questi aspetti legati ai tipi generici risultano piuttosto complessi e sicuramente difficili per chi si avvicini per la prima volta al linguaggio. Per completezza e poiché ricorrono spesso nella documentazione, abbiamo deciso di includere nel testo (anche se non in maniera del tutto esaustiva) queste parti. Chi si avvicina per la prima volta al linguaggio potrebbe ometterle e prenderle in esame per approfondimenti successivi.

A scopo di completezza e per non essere legati allo specifico linguaggio, abbiamo ritenuto opportuno trattare anche gli array. Di fatto, con le nuove caratteristiche di Java, questo argomento, che appare sempre più legato alla programmazione imperativa, avrebbe potuto essere ommesso.

**Supporto web**

L’home page del testo si trova all’indirizzo

http://pighizzini.dico.unimi.it/jb

Qui potrete trovare eventuale materiale integrativo o di aggiornamento, sviluppato successivamente alla pubblicazione del libro. Da questa pagina potete inoltre raggiungere le nostre home page e quelle dei nostri corsi, nelle quali è possibile reperire altro materiale.
Ringraziamenti

Questa pubblicazione è frutto della rielaborazione attraverso gli anni del materiale didattico preparato per gli studenti del nostro corso. Vogliamo quindi ringraziare i nostri studenti che, anche inconsapevolmente, hanno contribuito alla realizzazione di questo lavoro; il feedback positivo di molti è stato di grande incoraggiamento; le loro incertezze sono state uno stimolo a rivedere criticamente e a riscrivere alcune parti. Ringraziamo in particolare tutti coloro che hanno segnalato errori e imprecisioni.

Un ringraziamento a Pearson per avere accettato di pubblicare il nostro lavoro, rinnovando la fiducia nei nostri confronti, anche per questa terza edizione. In particolare, vogliamo ringraziare Alessandra Piccardo, academic editor, per l’entusiasmo con cui ha sostenuto fin dall’inizio e ha continuato a sostenere questo progetto, Micaela Guerra, per il costante e preciso supporto, e Bruno Lanata, copy-editor, per il paziente e puntuale lavoro di revisione del manoscritto grazie al quale idee e concetti hanno trovato la corretta espressione in un linguaggio semplice e scorrevole.

Giovanni Pighizzini
Mauro Ferrari
Milano, gennaio 2008