



## Prefazione

---

Questo libro è una guida graduale all'apprendimento della programmazione con il linguaggio Java, senza che sia richiesto alcun prerequisito in termini di conoscenze ed esperienze di programmazione. L'obiettivo è far conoscere e sperimentare Java nella sua completezza, illustrando le basi della programmazione ad oggetti e applicandone i principi, in modo da mettere in grado il lettore non soltanto di programmare, ma anche di comprendere il perché delle scelte adottate nel linguaggio e far suo un metodo di lavoro.

Gli argomenti trattati vengono presentati in modo graduale e sistematico e sono corredati da numerosi esempi illustrativi. Lo sforzo è stato quello di introdurre ogni proposizione sottolineando l'aspetto pratico e nondimeno mantenendone una descrizione rigorosa e priva di ambiguità. La stessa teoria degli oggetti viene presentata come una naturale evoluzione della programmazione strutturata mostrando a quali esigenze pratiche risponde.

Il libro tratta sostanzialmente del linguaggio in sé, ma approfondisce anche alcuni argomenti correlati, come le librerie per l'interfaccia grafica e per l'utilizzo su Internet, che sono necessari per comprendere come utilizzare il linguaggio in applicazioni complesse. In ogni caso, dato che le librerie sono destinate a mutare nel tempo, non è mai stato dato peso al nozionismo, ma sempre ai principi di base e alle tecniche usate, in modo tale da mettere il lettore in grado di apprendere semplicemente l'utilizzo di nuove librerie.

L'obiettivo non è solo illustrare teoricamente il linguaggio, ma anche fornire indicazioni pratiche d'utilizzo, per tale ragione sono considerate ed evidenziate anche le nuove caratteristiche dell'ultima versione Java 1.6 o, per usare il nome ufficiale "Java™ Platform, Standard Edition 6" o Java™ SE 6. Il motivo delle due differenti numerazioni è brevemente illustrato nel prossimo paragrafo.

## Java

Java è un linguaggio sviluppato da Sun Microsystems Inc. che ne ha registrato il nome e ne ha controllato l'evoluzione almeno fino al 13 novembre 2006 quando ha deciso di rilasciare il linguaggio e tutti i suoi componenti con licenza Open Source. Esistevano comunque già da tempo implementazioni di fornitori diversi sia del compilatore che della virtual machine.

Nato nei primi anni '90 come linguaggio per la programmazione di apparecchi per la navigazione su Internet collegati al televisore, ben presto ha dimostrato delle potenzialità così notevoli da diventare un prodotto a se stante destinato a essere impiegato per la programmazione a ogni livello.

La prima versione ufficiale è uscita nel 1996 col nome "JDK 1.0" (Java Development Kit versione 1.0), ma è stata presto seguita dal JDK 1.1 (1.1.0, 1.1.1 e così via fino alla versione 1.1.8) che ha introdotto importanti novità, come una nuova gestione degli eventi e la possibilità di creare classi interne ad altre classi. All'uscita del "JDK 1.2", Sun ha deciso di modificare il nome, per evidenziare che Java non è solo un linguaggio, ma una piattaforma applicativa già matura di cui il linguaggio è solo un componente. È uscito così il "Java 2 Standard Edition Development Kit" (J2SE). Il termine "Standard Edition" è stato aggiunto perché Sun ha reso disponibile anche il "Java 2 Enterprise Edition" (J2EE) che affianca al linguaggio una serie di strumenti per la realizzazione di applicazioni per la grande azienda, e il "Java 2 Micro Edition" destinato a dispositivi con limitate risorse. In pratica quindi da quel momento esiste una doppia numerazione, una per lo sviluppo (1.2, 1.3, 1.4) che tiene conto dell'evoluzione dell'ambiente di programmazione e una a livello di prodotto/piattaforma (Java 2). In ogni versione sono stati aggiunti miglioramenti senza però incidere sul linguaggio che in ogni caso ha sempre mantenuto la compatibilità all'indietro. La 1.4 ha inserito una nuova parola chiave, `assert`, ma le innovazioni vere sono arrivate con la versione 1.5, che ha cambiato anche il nome della piattaforma in "Java™ 2 Platform, Standard Edition 5" (J2SE™ 5.0): in essa sono state aggiunte tra l'altro, le classi generiche, i tipi enumerari e i cicli estesi. La versione attuale "Java™ Platform, Standard Edition 6" (Java™ SE 6) conserva però anche la vecchia numerazione 1.6. In questo testo può capitare di usare entrambe le numerazioni poiché non esiste ambiguità.

Java è oggi il linguaggio di riferimento per ogni genere di applicazione su qualsiasi tipo di macchina, dai telefonini al mainframe.

Le sue caratteristiche più interessanti dal punto di vista tecnico possono essere riassunte nei seguenti punti.

- *Java è un linguaggio orientato agli oggetti*

I progettisti hanno cercato di implementare le caratteristiche più interessanti dei linguaggi a oggetti preesistenti evitando però quelle particolarità ritenute inutilmente complicate. Il risultato è un ambiente semplice, compatto ed efficiente anche se per ottenere questi risultati si è dovuto in certi casi rinunciare a un disegno rigoroso.

- *Java è un linguaggio interpretato*

Lo scopo principale di questa scelta è l'indipendenza dei programmi dalla piattaforma hardware/software, ma ne consegue anche un più rapido ciclo di sviluppo. Un programma Java sorgente viene in effetti compilato, ma il codice prodotto, anziché essere legato alla macchina, è un codice intermedio che per essere eseguito ha bisogno di un interprete o macchina virtuale (Virtual Machine). Ciò permette una compilazione molto rapida senza necessità della fase di link. A differenza di altri linguaggi interpretati, Java ha una forte tipizzazione dei dati, che consente di individuare un gran numero di errori già in fase di compilazione. Ovviamente le prestazioni in fase di esecuzione sono inferiori a quelle di un linguaggio compilato, ma questo non costituisce un problema nella maggior parte dei casi, grazie anche a nuove tecniche messe a punto recentemente, come la compilazione in fase di esecuzione (Just In Time Compilation o JIT).

- *Java è un linguaggio moderno*

In fase di progettazione sono state previste diverse funzionalità che lo rendono robusto (come la gestione della memoria automatica con *garbage collector* e la gestione delle eccezioni) e adatto a sfruttare in modo semplice le caratteristiche dei moderni sistemi operativi, come il multithreading e le interfacce utente grafiche.

- *Java è un linguaggio orientato alle applicazioni di rete*

Oltre ad avere funzionalità per gestire semplicemente le connessioni tra elaboratori, nell'interprete sono state aggiunte delle protezioni particolari in modo da evitare accessi indesiderati al proprio computer che possano fare danni o divulgare informazioni riservate.

- *Java ha una ricchissima libreria a disposizione*

A differenza dei predecessori da cui eredita lo stile (C e C++), Java è fornito di una libreria di oggetti che consente ai programmatori di affrontare la maggior parte dei problemi senza dover ricorrere a prodotti di terze parti. Questo, oltre a semplificare il lavoro e diminuire la curva d'apprendimento, migliora la compatibilità e la portabilità.

Da quanto detto risulta quindi evidente, per chi opera in ambito universitario e professionale, l'importanza di conoscere questo linguaggio e i principi su cui si basa.

## Struttura del testo

I due capitoli iniziali sono dedicati a introdurre, in modo organico e propedeutico allo studio di Java e delle tecniche di programmazione, argomenti di base quali: algoritmi, sistemi di elaborazione, linguaggi. Un'appendice è dedicata alla rappresentazione e al trattamento dell'informazione, con una presentazione dei sistemi di numerazione e di

codifica. In questo modo anche lo studente che si avvicina per la prima volta alla programmazione ha a portata di mano tutti gli strumenti concettuali che gli sono necessari, senza che debba ricorrere obbligatoriamente ad altre fonti da cui attingere informazioni frammentate, più difficilmente riconducibili a un quadro d'insieme.

Java non è un linguaggio a oggetti "puro" (come ad esempio Smalltalk), per cui comprende tipi di dati primitivi e costrutti simili a quelli dei linguaggi più tradizionali. I Capitoli 3-7 introducono queste nozioni di base e non hanno a che vedere direttamente con la programmazione a oggetti. Gli argomenti, come nel resto del libro, vengono adeguatamente accompagnati da esempi ed esercizi e il lettore ha immediatamente la possibilità di sperimentare programmi completi.

Nel Capitolo 8 vengono analizzati i punti deboli dei linguaggi di programmazione strutturati. La teoria degli oggetti e la corrispondente implementazione in Java vengono esposte in modo pragmatico nel Capitolo 9 come evoluzione naturale del modello funzionale; numerosi esempi aiutano il lettore a calare i nuovi concetti nella pratica di programmazione. Nell'ambito dell'analisi e modellazione *object oriented* è illustrata e utilizzata la metodologia standard di riferimento UML (Unified Modelling Language) per la rappresentazione delle relazioni di generalizzazione, associazione, aggregazione e composizione.

Il Capitolo 10 è dedicato alle *interfacce*, alle *classi generiche* e ai *package* che, pur rientrando nella teoria generale degli oggetti, rappresentano i costrutti più innovativi introdotti in Java. Il Capitolo 11 illustra alcune *classi* che corredano il linguaggio e che è indispensabile conoscere per comprendere gli argomenti successivi.

Il Capitolo 12 descrive la gestione delle eccezioni, che in Java è parte integrante del linguaggio, mentre il Capitolo 13 si occupa dei *thread* che invece sono implementati come classi. Nel Capitolo 14 vengono illustrate alcune importanti caratteristiche relative all'esecuzione dei programmi e alla loro iterazione con il sistema operativo, mentre il Capitolo 15 si occupa delle classi dedicate alla gestione degli archivi. Data l'importanza dell'argomento, un ampio spazio è dato nei Capitoli 16 e 17 alla gestione dell'interfaccia grafica utente e nel Capitolo 18 ai componenti *Swing*. Nel Capitolo 19 vengono infine illustrate le *applet* e come queste vengono eseguite da un browser HTML.

A completamento dell'opera, l'Appendice A riporta le parole riservate del linguaggio, l'Appendice B il sorgente completo e commentato di un'applet che permette di giocare a Othello contro il computer, in modo tale da fornire un esempio concreto di programmazione a oggetti. Le Appendici C e D riportano rispettivamente `javadoc`, per realizzare una documentazione automatica del software, e un'introduzione ai sistemi di numerazione e alla rappresentazione dell'informazione.

## Contenuto

**Capitolo 1 Sistemi di elaborazione.** I primi due capitoli introducono i concetti di algoritmo e computer, riflettendo sulle possibilità e sui limiti dell'elaborazione automatica. L'obiettivo non è tanto di porre, seppur in forma semplice ed esemplificativa, i fondamenti teorici della disciplina, ma quanto di proiettare una luce sullo studio



della programmazione che andremo a svolgere nel testo, rendendo il lettore più consapevole del proprio lavoro e forse più motivato. In questo capitolo incontreremo problemi e relativi algoritmi di risoluzione con l'uso implicito dei meccanismi primari del controllo del flusso; risulterà evidente la necessità della memoria e il significato di esecuzione. Osserveremo l'architettura di un sistema di elaborazione e le parti che costituiscono un computer.

**Capitoli 2 Programmazione.** Introduce i linguaggi e la programmazione e dà concretezza alla distinzione tra linguaggi di alto e di basso livello, mostrando l'interazione tra programma e parti fisiche della macchina per poi concentrarsi sulla programmazione strutturata. Ragioneremo graficamente sul problema del "salto" e sulle conseguenze negative di un uso deregolamentato. Comanderemo la soluzione offerta dalla programmazione strutturata grazie al controllo del flusso garantito da sequenza, selezione, ripetizione e dal "blocco d'istruzioni". Le strutture di controllo verranno spiegate nel dettaglio utilizzando Java in modo informale. I paragrafi conclusivi accennano agli approcci top-down, alla programmazione modulare e alla programmazione orientata agli oggetti che verranno utilizzati nel testo.

Consigliamo il lettore che non sia già familiare con gli argomenti esposti a iniziare lo studio dal primo capitolo, anche se la trattazione vera e propria della programmazione Java inizia dal Capitolo 3, da dove si può partire senza che si sia perso niente d'indispensabile.

**Capitolo 3 Introduzione al linguaggio.** L'obiettivo di questo capitolo è mettere il lettore immediatamente in grado di realizzare i primi programmi in linguaggio Java. Impareremo a visualizzare informazioni sul monitor e accettare valori da tastiera, definire e utilizzare variabili e costanti, far riferimento a classi e metodi già definiti nelle librerie di sistema. Esamineremo i tipi dati interi, in virgola mobile, booleani e carattere, le conversioni di tipo automatiche e il *casting*. Apprenderemo le varie fasi della programmazione.

Gli esempi e gli esercizi del capitolo utilizzano una sola modalità di flusso, la sequenza, in cui le istruzioni vengono eseguite una dopo l'altra, nell'ordine in cui sono state scritte. Particolare attenzione viene posta all'operazione di assegnamento, e numerose riflessioni accompagnano l'introduzione del concetto di variabile e costante. Prende così corpo un primo approccio pratico agli oggetti e alla struttura di un programma Java.

**Capitolo 4 Strutture di controllo decisionali.** Percorre due tappe fondamentali verso l'apprendimento della programmazione: costrutti di controllo decisionali e blocchi d'istruzioni. Grazie al costrutto decisionale `if` faremo eseguire un'istruzione piuttosto che un'altra in base al presentarsi di una certa condizione. Comanderemo l'utilità delle istruzioni composte e degli `if` annidati. Conosceremo e sperimenteremo il significato, la gerarchia e l'associatività degli operatori aritmetici, relazionali, logici, dell'operatore di assegnamento e dell'operatore condizionale. Effettueremo scelte multiple con il costrutto `switch-case`. Tratteremo i dati attraverso gli operatori bit a bit, come *AND*, *OR*, *XOR*, *NOT*. Vedremo l'uso delle asserzioni.

Gli esempi e gli esercizi del capitolo utilizzano due modalità di flusso: la sequenza e la selezione, in cui un'istruzione o un blocco d'istruzioni vengono eseguiti se un'espressione risulta essere vera o meno. Particolare attenzione viene posta alla corrispondenza tra le istruzioni operative e le differenti vie aperte dai costrutti annidati e alla necessità o meno dei blocchi d'istruzioni.

**Capitolo 5 Strutture di controllo iterativo.** Costituisce un ulteriore passo fondamentale verso la programmazione strutturata, attraverso i costrutti di controllo iterativi. Grazie a `while`, `do-while` e `for` faremo ripetere l'esecuzione di un'istruzione o di un blocco d'istruzioni. Vedremo come gestire a piacimento il numero di iterazioni quale risultato della valutazione di un'espressione, che potrà essere un valore determinato a priori o dipendente dal presentarsi di una certa condizione maturata all'interno del ciclo stesso. Saremo in grado di utilizzare cicli annidati, incrementi e decrementi e l'operatore virgola. Approfondiremo ed estenderemo l'uso di espressioni e operatori. Si affacciano anche le interruzioni `break` e `continue`. Vengono utilizzate tutte le modalità di flusso della programmazione strutturata: sequenza, diramazione, iterazione congiuntamente con i blocchi d'istruzioni.

**Capitolo 6 Array, ricerche, ordinamenti.** Le variabili strutturate di tipo array permettono di compiere un salto qualitativo nel trattamento dei dati. Impareremo a definire e utilizzare array mono e multidimensionali, a gestire gli indici di vettori e matrici. Approfondiremo l'uso di operatori e cicli annidati. Gli esempi e gli esercizi mostrano la gestione di sequenze di dati omogenei, in particolare si consolida quanto appreso con la ricerca di massimo, minimo, media e con il *prodotto di matrici*.

Questo capitolo è anche un "laboratorio" per sperimentare, affrontando alcuni dei problemi "classici" della programmazione, tutto quello che è stato appreso. Implementeremo algoritmi di ricerca, ordinamento e fusione e ci impadroniremo delle tecniche per la gestione dei vettori. Grande importanza rivestono i metodi di *bubblesort*, ricerca binaria e *merge*, per se stessi e per l'occasione che offrono di riflettere sulle logiche di programmazione.

**Capitolo 7 I metodi.** Partendo da sottoprogramma e funzione, strumenti della programmazione modulare per la scomposizione di un problema in parti più semplici, il programma in sottoprogrammi, il capitolo introduce per analogia il concetto di metodo. Dichiareremo, definiremo e lavoreremo con nostri *metodi*, in particolare ne realizzeremo uno per l'acquisizione di dati da tastiera. Predisporremo ed effettueremo il passaggio dei parametri, gestiremo il valore di ritorno e il tipo `void`. Saremo interpretare e utilizzare visibilità e mascheramento dei nomi.

Nella seconda parte il capitolo presenta i metodi ricorsivi, metodi che richiamano in modo diretto o indiretto se stessi. Confronteremo iterazione e ricorsione; ci eserciteremo estensivamente con il *calcolo combinatorio* (fattoriale, disposizioni, combinazioni), la *successione di Fibonacci* e la *torre di Hanoi*. L'argomento è interessante almeno per tre ragioni: costringe a seguire con attenzione flusso di esecuzione e visi-

bilità delle variabili; induce a considerazioni e contrapposizioni tra eleganza ed efficienza degli algoritmi; permette un'ulteriore interessante sperimentazione dei metodi e del passaggio di parametri.

Acquisiremo familiarità con la programmazione e il linguaggio realizzando un programma per la *gestione di una sequenza* di valori per mezzo di metodi di immisione, ordinamento e ricerca.

**Capitolo 8 Verso la programmazione a oggetti.** Tratta le ragioni che hanno portato al paradigma degli oggetti. Dal problema alla sua suddivisione in problemi più semplici, dal programma monolitico alla programmazione modulare alle *funzioni*, alla complessità a sua volta indotta dalla necessità di far interagire tra loro i diversi moduli, per arrivare alla programmazione a *maniglie* e da questa alle risposte dell'approccio Object Oriented.

**Capitolo 9 Programmazione a oggetti in Java.** Presenta i concetti fondanti della programmazione a oggetti e la loro implementazione in Java: classe, oggetto, incapsulamento, ereditarietà, polimorfismo, overriding, binding dinamico, operatori sulle istanze, costruttori, finalizzazione, *this* e *super*, variabili e metodi di classe, classi astratte. Introduce la notazione UML (Unified Modelling Language) – la metodologia standard di riferimento per modellare la realtà d'interesse usando un approccio orientato agli oggetti – delle relazioni di generalizzazione, associazione, aggregazione e composizione.

Da questo capitolo in avanti, la presentazione degli argomenti è accompagnata da un utilizzo consapevole del linguaggio Java nell'ambito vero e proprio della programmazione orientata agli oggetti.

**Capitolo 10 Oltre le classi.** Le gerarchie di classi sono costruzioni logiche che ci aiutano a semplificare la comprensione del mondo; capita però che sia vantaggioso trattare oggetti di classi del tutto differenti in modo generale e omogeneo, in considerazione di alcune caratteristiche trasversali comuni che nulla hanno a che vedere con la gerarchia di classi utilizzata. La prima parte del capitolo affronta questo tema con le classi trasversali, le interfacce, le classi contenitrici (collezioni) e le interfacce *generics*. Seguono la programmazione per componenti, lo spazio dei nomi, i *package*, i modificatori di classe, di interfaccia e di metodo variabile e le interfacce e le classi annidate.

**Capitolo 11 Classi standard.** Tratta alcune delle classi più importanti che il linguaggio rende disponibili. Presenta la metaclassa `Class` e la classe `Object`, classi e metodi per la gestione delle stringhe – costruttori, confronti, estrazione di caratteri, conversioni –, per la gestione delle collezioni – `Vector`, iteratori, `for-each`. Prosegue illustrando le classi involucro – `Number`, numeri interi, numeri in virgola mobile, `BigDecimal`, `BigInteger`, *autoboxing* – e i metodi che implementano le funzioni matematiche e gli enumerali.

**Capitolo 12 Gestione delle eccezioni.** In Java la gestione degli errori o *eccezioni* fa parte del linguaggio. Illustra la classe `Throwable`, `try` per gettare un'eccezione, `catch` per catturarla, `finally` per forzare l'esecuzione di istruzioni, le classi `Error` per la segnalazione di problemi a cui il programmatore non può porre rimedio, ed `Exception` per problemi che viceversa possono essere gestiti dal programmatore, la clausola `throws` nella dichiarazione dei metodi e il rigetto esplicito con `throw`.

**Capitolo 13 Thread.** Mostra l'attivazione asincrona e l'esecuzione indipendente di moduli di programma. Presenta la potenza e le problematiche del multithreading, la classe `Thread` – attributi, stati, istanza e attivazione dei *thread* – la sincronizzazione e le comunicazioni tra *thread* e le variabili volatili.

**Capitolo 14 Ambiente di esecuzione.** Studia le caratteristiche relative all'esecuzione dei programmi e alla loro iterazione con il sistema. A tale scopo viene premessa la trattazione di alcune classi di utilizzo generale le cui istanze sono utilizzate per scambiare informazioni tra programmi e sistema. Mostra la gestione di insiemi di oggetti, l'interfaccia `Map` e i suoi metodi, la gestione della sicurezza, le classi `Runtime` e `Process`, la classe `Date`, il metodo `main`, la classe `SecurityManager`, la classe `System` per gestire alcune risorse del sistema, l'oggetto `Properties` per la definizione dell'ambiente. Sono descritti altri metodi utili per l'ambiente di esecuzione.

**Capitolo 15 Gestione dell'I/O.** Illustra l'organizzazione dell'input/output come *flusso* da un programma fonte a un programma destinazione per scambiare dati, riceverli o inviarli da dispositivi esterni come tastiere e stampanti, per le connessioni in rete. Presenta il package `java.io` per il controllo dei flussi, la gestione di file e directory, i flussi di input e di output, le classi e le interfacce per l'accesso casuale, la classe `RandomAccessFile` per trattare i file come un array di byte memorizzati sul file system, gli oggetti persistenti, le classi `ObjectOutputStream` e `ObjectInputStream`, che implementano in modo automatico la *serializzazione* e la *deserializzazione* di oggetti.

**Capitolo 16 Interfaccia grafica.** I Capitoli 16, 17 e 18 trattano le GUI (*Graphical User Interface*) e la programmazione *guidata dagli eventi* (*event driven*). Questo capitolo introduce il package `java.awt`, gli eventi e gli oggetti *listener*, la visualizzazione di oggetti grafici, la lettura da tastiera, gli eventi dal cursore del mouse, la gestione di colori, font e la posizione del testo.

**Capitolo 17 Controlli grafici.** Approfondisce la gestione delle GUI con il package `java.awt`. Presenta la gestione di bottoni, l'inserimento e la formattazione di testi, le etichette, le caselle di controllo a scelta singola e a scelte mutuamente esclusive, le liste a selezione singola e multipla, le barre di scorrimento, i menu, l'impaginazione secondo diverse strategie, la suddivisione della finestra principale in aree logiche.

**Capitolo 18 Swing.** Nell'ambito delle GUI il capitolo tratta *Swing*, una libreria di componenti grafici totalmente scritti in Java in cui solo le finestre esterne vengono richieste al sistema operativo tramite chiamate all'*awt*, studiato nei precedenti due capitoli, il

resto dei componenti è gestito totalmente da linguaggio. Illustra i componenti grafici, le applicazioni Swing, il *Java Look and Feel*, il progetto di nuovi componenti.

**Capitolo 19 Internet e Java.** Dopo aver introdotto alcune necessarie nozioni di base sul funzionamento di Internet, sono illustrate le *applet* e il modo di impiegarle. Presenta il concetto di protocollo, porta, nome risorsa, URL, WWW, HTTP, HTML. Introduce all'uso delle applet nel browser, alla sicurezza, alla realizzazione di immagini e colori, ai metodi della classe `Applet`.

L'**Appendice A** riporta le parole riservate del linguaggio, l'**Appendice B** descrive un'applet Java con il sorgente completo e commentato che permette di giocare a Othello contro il computer in modo tale da fornire un esempio concreto di programmazione a oggetti. L'**Appendice C** presenta `javadoc` per una corretta documentazione dei programmi. L'**Appendice D** è dedicata ai sistemi di numerazione e alla rappresentazione dell'informazione.

Come per gli altri testi della collana si accede alla documentazione digitale relativa al libro tramite il sito `www.ateneonline.it`. Vi si possono scaricare i listati dei programmi di tutto il testo, le soluzioni degli esercizi e altro materiale didattico di supporto a professori e studenti.

Nel libro sono riportati numerosi esempi di programmazione. Si consiglia il lettore di procurarsi un ambiente di programmazione Java e provare gli esempi, almeno quelli che risultano meno intuitivi. Il JDK (Java Development Kit) è l'ambiente di sviluppo ufficiale ed è disponibile gratuitamente su Internet al sito Sun (`http://java.sun.com` oppure `http://www.javasoft.com`). JDK è un ambiente spartano, ma corrisponde con sicurezza all'ultima release del linguaggio. Negli stessi siti è anche possibile trovare NetBeans, un IDE (Integrated Development Environment cioè ambiente di sviluppo integrato) grafico che semplifica lo sviluppo di programmi e la documentazione aggiornata del linguaggio, delle API (Application Program Interface) e di tutti gli strumenti di contorno che completano l'architettura di Java.

## Ringraziamenti

Il testo ha beneficiato anche dei preziosi suggerimenti pervenutici da lettori degli altri nostri libri e in particolare dai docenti che li hanno adottati nei corsi d'informatica. Siamo loro grati per il contributo e la disponibilità al confronto.

Desideriamo infine ringraziare vivamente Francesca Biancalana, David Bertacca e Matilde Murgueytio per l'importante supporto fornito durante la stesura del libro; auguriamo di cuore buono studio/lavoro al lettore. Chi volesse mettersi in contatto con noi può scriverci ai seguenti indirizzi di e-mail: `marco@picosoft.it`, `andrea@softpi.it`.

*Marco Bertacca  
Andrea Guidi*