

---

# Prefazione

---

In questo volume completiamo l'esplorazione del linguaggio Java che abbiamo iniziato in *Java Fondamenti di programmazione*. I due testi fanno parte di un percorso didattico unitario, come testimoniano i frequenti riferimenti incrociati (soprattutto negli esercizi di fine capitolo); abbiamo fatto il possibile, comunque, per renderli utilizzabili anche separatamente.

Il volume *Fondamenti* si concentrava sugli aspetti di base della programmazione in Java (pur essendo pienamente aggiornato a Java 2 e integrando completamente, fin dai capitoli iniziali, elementi quali l'interfaccia grafica Swing). Questo volume introduce tecniche e metodologie più avanzate, dalle funzionalità grafiche e multimediali al networking in Java.

In particolare, il capitolo 1 inizia a introdurre le caratteristiche multimediali di Java. La tradizionale programmazione in C e C++ si è sempre limitata a gestire gli input/output come normali caratteri e, anche se alcune versioni del linguaggio C++ sono supportate da librerie di classi in grado di produrre delle immagini, l'uso di queste librerie rende le applicazioni non portabili. Il bello delle funzionalità grafiche di Java è che sono indipendenti dalla piattaforma e, quindi, estremamente portabili.

Il capitolo 2 introduce la creazione di applet e applicazioni con l'utilizzo di interfacce utente grafiche (GUI). Il capitolo 3 prosegue la discussione del capitolo 2. Ancora una volta, il segreto di Java sta nell'indipendenza dalla piattaforma; un'applet o un'applicazione sviluppate in Java possono, infatti, essere eseguite su tutte le piattaforme Java. Questo libro prende in esame in particolar modo i componenti GUI Swing, che sono dei componenti GUI indipendenti dalla piattaforma scritti completamente in Java; ciò li rende estremamente flessibili e personalizzabili. Questi componenti possono assumere l'aspetto dell'interfaccia utente della piattaforma utilizzata dal computer sul quale sono in esecuzione, oppure possono utilizzare il *look and feel* standard di Java, che garantisce un'interfaccia utente grafica uniforme su tutte le piattaforme.

Il capitolo 4 è forse uno dei capitoli più importanti del libro. È un dato di fatto che non tutto funziona sempre alla perfezione, soprattutto alla luce delle velocità con cui operano, attualmente, i computer (normalmente centinaia di milioni di operazioni al secondo). In Java, quando un componente (ovvero una classe) incontra delle difficoltà, può "lanciare" un'eccezione; l'ambiente di quel componente, a sua volta, è programmato in modo da "gestire" quell'eccezione. Le capacità di gestione delle eccezioni di Java sono particolarmente orientate al mondo degli oggetti, dove i programmatori realizzano sistemi software utilizzando in gran parte componenti riutilizzabili creati da altri programmatori. Per usare un componente Java, è necessario conoscere quindi non solo come questo componente si comporta quando "tutto va bene", ma anche le eccezioni che lancia quando "qualcosa va male".

Il capitolo 5 prende in esame la programmazione di applet e applicazioni in grado di eseguire più attività in parallelo. Attualmente, i processori stanno diventando così poco costosi che è possibile costruire computer con più processori che lavorano in parallelo (i

cosiddetti *multiprocessori*). La tendenza, quindi, è quella di avere computer in grado di eseguire più attività contemporaneamente. La maggior parte dei linguaggi di programmazione attualmente disponibili, tra cui C e C++, non offrono funzionalità per gestire più operazioni parallele; questi linguaggi, infatti, sono definiti *linguaggi di programmazione sequenziale* o *single-thread*. Java permette invece di realizzare delle applicazioni multithreaded, ovvero applicazioni in grado di specificare che più attività possono essere eseguite contemporaneamente. Java è quindi in grado di gestire al meglio le applicazioni multimediali più sofisticate, basate sui multiprocessori e sulle reti, ovvero le applicazioni del futuro. Come vedrete in questo capitolo, poi, il multithreading è efficace anche sui sistemi con processore singolo.

Il capitolo 6 prende in esame le capacità multimediali di Java nell'ambito delle immagini, delle animazioni, degli audio e dei video, introducendo anche il *Java Media Player*. Sembra incredibile che gli studenti di un primo corso di programmazione possano già scrivere applicazioni con tutte queste capacità, ma con Java è possibile. Ciò che è possibile fare con le capacità multimediali di Java non ha limiti; immaginate di avere accesso (attraverso Internet e la tecnologia CD-ROM) a immense librerie di immagini, audio e video, da potere utilizzare per creare delle applicazioni. Ormai più della metà dei computer venduti in tutto il mondo contengono dispositivi multimediali, come lettori CD e schede audio.

Il capitolo 7 prende in esame l'input/output di dati che avviene attraverso i flussi di dati diretti da e per i file. Questo capitolo è uno dei più importanti per i programmatori che svilupperanno applicazioni di tipo commerciale. Come fa un programma a passare i dati a un dispositivo di memorizzazione secondaria, come per esempio un disco? Come fa un programma a recuperare i dati già memorizzati su disco? Cosa sono i file sequenziali? Cosa sono i file ad accesso casuale? Cos'è il buffering e come aiuta i programmi con un'elevata quantità di input/output?

Il capitolo 8 mostra invece come Java possa essere utilizzato per accedere ai database relazionali. Qualsiasi azienda, al giorno d'oggi, vive sui propri dati; Java offre un'ampia varietà di classi che permettono di utilizzare e analizzare i dati dei database. In questo capitolo ci concentreremo soprattutto sulla capacità di Java di utilizzare JDBC (Java Database Connectivity) per collegarsi a un'origine dati Microsoft ODBC (Open Database Connectivity) attraverso il "ponte" JDBC-ODBC. Gli esempi di questo capitolo usano un database Microsoft Access (books.mdb) al quale è possibile accedere attraverso SQL (Structured Query Language).

I capitoli 9, 10 e 11 parlano di come sia possibile scrivere programmi in grado di comunicare nell'ambito delle reti. Cos'è un client? Cos'è un server? Come fanno i client a chiedere ai server di eseguire i propri servizi? Come fanno i server a inviare i risultati ai clienti? Cos'è un URL (uniform resource locator)? Come fa un programma a caricare pagine Web? Come è possibile utilizzare Java per sviluppare applicazioni di tipo collaborativo? Questi capitoli prendono in esame entrambi i lati di una relazione di tipo *client-server*, dove il client richiede l'esecuzione di un'azione e il server esegue questa azione rispondendo poi al client. Questo modello di comunicazione basato sulla domanda-risposta costituisce la base dei *servlet* Java, ovvero il più alto livello di networking di Java.

Il capitolo 9 analizza il networking all'interno di un modello in cui i browser Web comunicano con dei server Web quando gli utenti navigano in Internet. Quando un utente seleziona un sito Web in cui navigare con il proprio browser (l'applicazione client), al

corrispondente server Web (l'applicazione server) viene inviata una richiesta, alla quale il server normalmente risponde inviando la pagina HTML che il browser deve poi visualizzare. In questo modello, le comunicazioni tra client e server vengono gestite automaticamente. Il capitolo inizia a presentare le applicazioni distribuite di tipo multi-tier (a più livelli), dove parte dell'applicazione opera su computer separati distribuiti lungo una rete. Questo capitolo sfrutta la tecnologia dei flussi vista nel capitolo 7 e la tecnologia di database vista nel capitolo 8 per iniziare a realizzare applicazioni Java destinate a essere impiegate nel mondo reale.

Il capitolo 10 continua la presentazione dei programmi in grado di comunicare sulle reti e delle applicazioni multi-tier. Remote Method Invocation (RMI) permette ai programmi Java di comunicare tra di loro per mezzo di chiamate di metodo che vengono automaticamente inviate lungo la rete. RMI è più complesso dei servlet, in quanto richiede più lavoro da parte del programmatore per impostare le interazioni iniziali tra le varie applicazioni. Una volta che questo meccanismo è stato impostato, però, le comunicazioni sulla rete sono trasparenti per l'applicazione. Anche in questo caso, il capitolo fa uso della tecnologia dei flussi vista nel capitolo 7.

Il capitolo 11 introduce il networking Java di livello più basso, ovvero i socket basati su flusso e i datagrammi. Viene illustrato come sia possibile creare dei client e dei server semplici in grado di gestire tutti gli aspetti di networking di livello più basso. Anche in questo caso, è Java a occuparsi direttamente di tutte le attività più complesse, sfruttando ancora la tecnologia dei flussi del capitolo 7. Gli esempi presentati in questo capitolo mostrano un'applet che interagisce con il browser in cui viene eseguito, la creazione di un mini browser Web, la comunicazione tra due programmi Java che usano socket basati su flusso, la comunicazione tra due programmi Java che usano pacchetti di dati, e come creare un server multithreaded in grado di interagire con più di un client per volta.

Il capitolo 12 prende in esame l'organizzazione dei dati in gruppi, come per esempio gli elenchi collegati, gli stack (o pile), le code e le strutture ad albero. Ogni struttura di dati possiede delle proprietà importanti, utili in un'ampia varietà di applicazioni. Vengono presentati tutti gli aspetti della realizzazione di queste strutture di dati, utilizzando varie classi estremamente utili. Il capitolo presenta dei principi generali che possono essere utilizzati durante l'implementazione di moltissime classi. Anche se è utile conoscere il funzionamento di queste classi, i programmatori Java scoprono presto che la maggior parte delle strutture di dati necessarie sono già disponibili nelle librerie di classi, come per esempio `java.util`, della quale si parla nei capitoli 13 e 14. Il capitolo 12 approfondisce ulteriormente i temi discussi nei capitoli 8 e 9 del volume Fondamenti riguardo alla programmazione basata su e orientata agli oggetti.

Il capitolo 13 prende in esame varie classi del package `java.util`, approfondendo ulteriormente il concetto di riutilizzo del software. Le classi si trovano all'interno delle librerie di classi perché sono generalmente utili, corrette, portabili, e così via. Qualcuno ha lavorato molto per preparare queste classi, quindi perché non usarle? La nostra convinzione è che le librerie di classi continueranno a crescere in modo esponenziale nei prossimi anni. Se così sarà, la vostra bravura di programmatori dipenderà da quanto conoscerete le classi già esistenti, e da come saprete riutilizzarle in modo intelligente per sviluppare rapidamente programmi software di ottima qualità. Questo capitolo prende in esame molte classi. Due delle più utili sono `Vector` (un array dinamico in grado di crescere o ridursi se necessario)

e Stack (una struttura di dati dinamica che permette inserimenti e cancellazioni dalla propria estremità superiore). Il bello di queste due classi è che sono collegate tra di loro attraverso l'ereditarietà, di cui si parla nel capitolo 9 del volume Fondamenti, così che il package `java.util` stesso implementi alcune classi al posto di altre, sfruttando al massimo il concetto di riutilizzo.

Il capitolo 14 presenta alcune delle nuove classi del package `java.util`, che forniscono implementazioni predefinite di molte delle strutture di dati viste nel capitolo 12. Anche questo capitolo riprende il concetto di riutilizzo. Queste classi prendono come modello una libreria di classi simile del linguaggio C++ (la Standard Template Library). Le *collezioni* forniscono ai programmatori Java un insieme di strutture di dati per la memorizzazione e il recupero dei dati, oltre a un insieme di algoritmi (procedure) che permettono ai programmatori di manipolare questi dati (per esempio, cercando un particolare dato, oppure organizzando i dati in un determinato ordine).

Il capitolo 15 spiega come sia possibile organizzare le classi Java all'interno di componenti software riutilizzabili. I JavaBean possono essere personalizzati graficamente all'interno degli ambienti di sviluppo Java. Il capitolo introduce il JavaBeans Development Kit (BDK) e il BeanBox, che è possibile usare per testare i propri *bean*. Il BeanBox illustra i concetti chiave per la manipolazione di un bean all'interno di un tipico ambiente di sviluppo grafico. Il capitolo prende in esame gli aspetti di progettazione che stanno alla base dello sviluppo dei bean. Oltre a ciò, spiegheremo come sia possibile creare un file Java Archive (JAR) per eseguire un'applicazione semplicemente facendo clic sul nome del suo file all'interno del file manager del proprio sistema (un tipico modo di eseguire le applicazioni su molte piattaforme).

## La metodologia di insegnamento

Questo libro contiene un'ampia varietà di esempi, esercizi e progetti che prendono spunto da molte situazioni, e che offrono agli studenti la possibilità di risolvere problemi reali. Il libro prende in esame i principi della progettazione del software, insistendo sull'importanza della chiarezza dei programmi ed evitando l'uso di terminologia complessa a favore di esempi chiari e diretti, il cui codice è stato testato sulle piattaforme Java più diffuse.

## L'apprendimento attraverso il codice

Il libro presenta una grande quantità di esempi basati sul codice. Ogni argomento viene presentato nell'ambito di un programma Java completo e funzionante (applet o applicazione), seguito sempre da una o più finestre che mostrano l'output del programma in questione. Viene quindi usato il linguaggio per insegnare il linguaggio, e la lettura di questi programmi offre un'esperienza molto simile alla loro esecuzione reale su di un computer.

## L'accesso al World Wide Web

Tutto il codice presente nel libro si trova anche in Internet, all'indirizzo <http://www.apogeeonline.com/education/booksite>. Il nostro consiglio è quello di scaricare tutto il codice, eseguendo poi ogni singolo programma via via che appare nel testo. Potete anche modificare il codice degli esempi e vedere cosa succede, imparando così a programmare programmando. (Tutto questo materiale è protetto da diritti d'autore,

quindi utilizzatelo liberamente per studiare Java, ma non pubblicatene alcuna parte senza l'esplicito permesso da parte degli autori e della casa editrice.)

## Obiettivi

Ogni capitolo inizia con la presentazione degli *Obiettivi*. Gli studenti possono così sapere in anticipo ciò che andranno ad apprendere e, alla fine della lettura del capitolo, potranno verificare se hanno raggiunto o meno questi obiettivi.

## Il codice e gli esempi

Le funzionalità di Java vengono presentate nell'ambito di programmi Java completi e funzionanti. Ogni programma è seguito dalle immagini degli output che vengono prodotti, così che gli studenti possano assicurarsi della correttezza dei risultati. I programmi presentati vanno da poche linee di codice a esempi composti da varie centinaia di righe di codice. Gli studenti dovrebbero scaricare tutto il codice dal sito Web <http://www.apogeeonline.com/education/booksite>, eseguendo poi ogni programma via via che questo viene presentato all'interno del testo.

## Consigli e suggerimenti

Il libro offre molti suggerimenti riguardanti la programmazione, per aiutare gli studenti a concentrarsi sugli aspetti più importanti dello sviluppo di programmi. Questi consigli vengono forniti attraverso le sezioni chiamate *Buona abitudine*, *Errore tipico*, *Collaudo e messa a punto*, *Obiettivo efficienza*, *Obiettivo portabilità*, *Ingegneria del software*, *L'esperienza dell'utente*.



### Buona abitudine

Il concetto più importante per chi inizia a programmare è la chiarezza, e all'interno delle sezioni *Buona abitudine* vengono presentate delle tecniche per la scrittura di programmi chiari, comprensibili e più facilmente gestibili.



### Errore tipico

Tutti gli studenti che affrontano per la prima volta un nuovo linguaggio tendono a commettere frequentemente gli stessi errori. Le sezioni *Errore tipico* aiutano gli studenti a evitare di commettere questi comuni errori.



### Collaudo e messa a punto

Queste sezioni forniscono consigli circa le attività di test e debugging dei programmi Java, anche a livello preventivo.



### Obiettivo efficienza

In base alla nostra esperienza, insegnare agli studenti come scrivere programmi chiari e comprensibili deve costituire l'obiettivo più importante di qualsiasi corso di programmazione. Gli studenti, però, vogliono normalmente imparare a scrivere programmi che vengano eseguiti in modo veloce, che utilizzino poca memoria,

che richiedano una minima quantità di comandi e che offrano prestazioni eccellenti. Le sezioni *Obiettivo efficienza* offrono suggerimenti su come migliorare le prestazioni dei propri programmi.



### Obiettivo portabilità

Alcuni programmatori pensano che, implementando un'applicazione Java, questa sia immediatamente portabile su tutte le piattaforme Java; sfortunatamente, non è sempre così. Le sezioni *Obiettivo portabilità* aiutano gli studenti a scrivere codice realmente portabile, fornendo inoltre informazioni su come Java sia in grado di raggiungere questo elevato livello di portabilità.



### Ingegneria del software

La programmazione orientata agli oggetti implica una totale rivisitazione del modo in cui vengono costruiti i sistemi software. Java è un linguaggio molto efficace nell'ambito della progettazione del software, e le sezioni *Ingegneria del software* prendono in esame gli aspetti architetturali e progettuali che influiscono sulla realizzazione di sistemi software, specialmente nel caso di sistemi su vasta scala. Molte di queste informazioni saranno utili agli studenti nei corsi più avanzati, oltre che nel mondo del lavoro.



### L'esperienza dell'utente

Le sezioni *L'esperienza dell'utente* prendono in esame le interfacce grafiche utente, aiutando gli studenti a progettare le proprie interfacce grafiche in conformità con le convenzioni previste dal mondo reale.

## Esercizi di autovalutazione

Il libro propone molti esercizi corredati di risposte, così che gli studenti possano prepararsi alle esercitazioni vere e proprie. Gli studenti dovrebbero essere incoraggiati a svolgere tutti questi esercizi di autovalutazione.

## Esercizi

Ogni capitolo si conclude con un insieme di esercizi di vario tipo, che permettono agli insegnanti di adattare le proprie lezioni alle esigenze particolari di ogni classe. Gli esercizi verificano l'apprendimento dei concetti e dei termini più importanti, chiedono agli studenti di scrivere singole istruzioni Java, porzioni di classi e metodi Java, oppure metodi, classi, applet e applicazioni Java complete.

## Indice analitico

Alla fine del libro è possibile trovare un indice analitico completo, molto utile sia a chi legge questo libro per la prima volta che ai programmatori che lo usano come riferimento.