Prefazione

Questo libro ha come obiettivo quello di insegnare la programmazione in Java in modo semplice e immediato, andando direttamene al sodo e cercando di concentrarsi sull'essenziale.
Dire le cose essenziali, tuttavia, non comporta tralasciare informazioni importanti e, infatti, per ogni argomento si è cercato di mostrare i punti salienti, corredandoli di esempi e listati.
Il lettore tenga presente, tuttavia, che questo non è un libro *for dummies* poiché qui sono presenti tutti gli argomenti che una guida completa si prefigge di trattare a cui sono associati svariati listati da studiare, compilare e provare.

Organizzazione del libro

Il libro è organizzato nei capitoli elencati di seguito.

- *Capitolo 1 Introduzione al linguaggio*: introduciamo il lettore a una panoramica del linguaggio e spieghiamo i concetti propedeutici per compilare ed eseguire un programma Java.
- *Capitolo 2 Variabili: costanti, letterali e tipi*: mostriamo come dichiarare e definire i dati di un programma e come attribuire a essi un tipo di appartenenza.
- *Capitolo 3 Array*: spieghiamo la struttura di dati array e come manipolarla. Mostriamo, altresì, la gestione degli array multi-dimensionali.
- *Capitolo 4 Operatori*: mostriamo tutti gli operatori che il linguaggio mette a disposizione per manipolare i dati. Si passa dall'illustrazione dei semplici operatori aritmetici a quella dei complessi operatori bitwise;
- *Capitolo 5 Strutture di controllo*: vediamo come possiamo gestire l'esecuzione del flusso di un programma attraverso le strutture di iterazione e di selezione;
- *Capitolo 6 Metodi*: mostriamo come progettare i metodi ovvero le funzioni basilari di un programma che eseguono determinate operazioni.

NOTA

Nell'organizzazione dei capitoli si è preferito introdurre i metodi prima del costrutto di classe, poiché essi sono "strutture" sintattiche più semplici, in modo da affrontare i concetti essenziali del linguaggio in ordine crescente di complessità. Filosoficamente, questa scelta si sposa bene con l'inquadramento del paradigma a oggetti (basato sulle classi) inteso come estensione del paradigma procedurale (basato sulle procedure). Non a caso, le strutture di controllo della programmazione a oggetti, che vengono poi utilizzate all'interno dei metodi, sono le stesse del paradigma procedurale.

- *Capitolo 7 Programmazione basata sugli oggetti*: vediamo come si crea un nuovo tipo di dato attraverso il costrutto di classe. Illustriamo che cosa sono i membri di una classe e la loro visibilità (*information hiding*) e come si possono definire al suo interno. Analizziamo i metodi costruttori, la keyword `this` e la progettazione di classi annidate. Infine, trattiamo i tipi enumerati.
- *Capitolo 8 Programmazione orientata agli oggetti*: mostriamo come si creano gerarchie di classi (ereditarietà) e che cos'è il polimorfismo. Parliamo, inoltre, di classi astratte e di interfacce. Chiudiamo il capitolo con l'illustrazione delle classi anonime.
- *Capitolo 9 Programmazione generica*: spieghiamo come si creano metodi e classi generiche, ovvero come si effettua una programmazione che manipola tipi parametrici.
- *Capitolo 10 Errori software*: illustriamo come si gestiscono gli errori software la programmazione delle eccezioni.
- *Capitolo 11 Package*: vediamo come si creano librerie di codice e come si rendono disponibili (*deployment*) in un ambiente di sviluppo o di produzione.
- *Capitolo 12 Annotazioni*: analizziamo come si creano e si utilizzano dei metadati che annotano gli elementi (variabili, metodi e così via) del linguaggio Java.
- *Capitolo 13 Documentazione del codice sorgente*: mostriamo come utilizzare dei tag "speciali" che consentono di formattare il codice sorgente in modo che sia possibile generare per esso un'adeguata documentazione.
- *Capitolo 14 Caratteri e stringhe*: impariamo a utilizzare le classi principali che consentono di manipolare i caratteri e le stringhe (`Character`, `String` e così via).
- *Capitolo 15 Espressioni regolari*, spieghiamo che cosa sono e come impiegare le *regex* per costruire sofisticati pattern per la ricerca di corrispondenze di caratteri.
- *Capitolo 16 Collezioni*: studiamo come utilizzare le collezioni o contenitori (`Collection`, `List`, `Map` e così via) che sono costituite da gruppi di elementi tra di loro naturalmente collegati che possono subire operazioni di manipolazione.
- *Capitolo 17 Programmazione concorrente*: trattiamo i concetti di processo e thread e mostriamo come scrivere in Java programmi che sono in grado di eseguire più operazioni in parallelo.
- *Capitolo 18 Input/Output: stream e file*: illustriamo come effettuare le comuni operazioni di input e output dei dati (stream, file e così via).
- *Capitolo 19 Progettazione di interfacce utente*: impariamo come scrivere programmi Java dotati di una GUI (*Graphical User Interface*) mediante l'utilizzo del potente framework Swing.

- *Capitolo 20 Programmazione di rete*: studiamo che cos'è una rete (suite TCP/IP) e le API che Java mette a disposizione per la programmazione dei socket e dei datagram.
- *Capitolo 21 Programmazione dei database*: vediamo come scrivere applicazioni che si interfacciano a una base dati con l'ausilio dell'architettura JDBC.
- *Capitolo 22 Sviluppo di applicazioni web*: impariamo come utilizzare la piattaforma JEE 6 (Java Enterprise Edition) per sviluppare programmi per il web (servlet, JavaServer Pages, JavaServer Faces e così via).
- *Appendice A Installazione e configurazione della piattaforma JSE*: vediamo come installare e configurare la piattaforma standard di Java.
- *Appendice B Installazione e configurazione della piattaforma JEE*: vediamo come installare e configurare la piattaforma di Java che è destinata alla progettazione di applicazioni web.
- *Appendice C Installazione e configurazione di MySQL*: illustriamo come installare e configurare il database server MySQL.
- *Appendice D Installazione e utilizzo di NetBeans*: mostriamo come installare e utilizzare l'IDE (*Integrated Development Environment*) NetBeans.
- *Appendice E Applet*: vediamo come scrivere dei programmi Java, denominati *applet*, che girano all'interno di un browser web.

Convenzioni utilizzate

Ogni argomento del libro è, ovviamente, diviso in capitoli. Ogni capitolo è numerato in ordine progressivo e denominato significativamente nel suo obiettivo didattico (per esempio **Capitolo 2 Variabili, costanti, letterali e tipi**). I capitoli sono, poi, suddivisi in paragrafi di pertinenza.
All'interno dei paragrafi possiamo avere dei blocchi di testo o di grafica, ausiliari alla teoria, denominati come segue:

- **Listato NrCapitolo.NrProgressivo Descrizione…** per i listati del codice sorgente;
- **Decompilato NrCapitolo.NrProgressivo Descrizione…** per i listati dei file `.class` decompilati;
- **Sintassi NrCapitolo.NrProgressivo Descrizione…** per la sintassi di un costrutto del linguaggio;
- **Snippet NrCapitolo.NrProgressivo Descrizione…** per un frammento di codice sorgente;
- **Shell NrCapitolo.NrProgressivo Descrizione…** per un comando di shell;
- **Warning NrCapitolo.NrProgressivo Descrizione…** per un avviso (*warning*) del compilatore;
- **Errore NrCapitolo.NrProgressivo Descrizione…** per un errore di compilazione o di esecuzione;
- **Output NrCapitolo.NrProgressivo Descrizione…** per l'output di un programma;

- **Figura NrCapitolo.NrProgressivo Descrizione…** per una figura;
- **Tabella NrCapitolo.NrProgressivo Descrizione…** per una tabella

Per esempio, il blocco denominato **Listato 6.2 Classe Overload** indica il listato di codice numero `2` del capitolo `6` avente come descrizione la classe `Overload`.
Per quanto attiene ai listati, abbiamo adottato le seguenti convenzioni:

- i caratteri punti di sospensione **...** eventualmente presenti indicano che alcune parti del listato sono state, in quel punto, omesse. Ovviamente le medesime parti sono presenti nei relativi file `.java` allegati al libro. Gli stessi caratteri possono talvolta trovarsi anche negli output di un programma eccessivamente lungo;
- i caratteri formattati in **grassetto** indicano parti di codice particolarmente importanti.

Codice sorgente e classi

Presso l'indirizzo `http://www.apogeonline.com/libri/9788850329885/scheda` è possibile scaricare un archivio ZIP contenente i listati di ogni capitolo.
Per rendere agevole la compilazione e l'esecuzione dei programmi riteniamo utili i seguenti consigli:

- creare, se si utilizza Windows, le seguenti strutture di directory:
  - per i sorgenti `C:\MY_JAVA_SOURCES`;
  - per le classi `C:\MY_JAVA_CLASSES`;
  - per i package `C:\MY_JAVA_PACKAGES`;
  - per gli archivi `JAR C:\MY_JAVA_JARS`;
  - per la documentazione `C:\MY_JAVA_DOCUMENTATION`.
- creare, se si utilizza GNU/Linux, le seguenti strutture di directory:
  - per i sorgenti `/opt/MY_JAVA_SOURCES`;
  - per le classi `/opt/MY_JAVA_CLASSES`;
  - per i package `/opt/MY_JAVA_PACKAGES`;
  - per gli archivi JAR `/opt/MY_JAVA_JARS`;
  - per la documentazione `/opt/MY_JAVA_DOCUMENTATION`.
- copiare il listato da compilare nella cartella `MY_JAVA_SOURCES`;
- compilare il listato utilizzando il comando di compilazione `javac` con il flag `-d` che indica il percorso dove generare i file `.class`. Per esempio, per compilare il listato `UsoDiChar.java` invocare, dalla directory `MY_JAVA_SOURCES` il compilatore come:
  - `javac -d C:\MY_JAVA_CLASSES UsoDiChar.java`, per il sistema operativo Windows;
  - `javac -d /opt/MY_JAVA_CLASSES UsoDiChar.java`, per il sistema operativo Gnu/Linux.

- eseguire il programma dalla cartella `MY_JAVA_CLASSES`, ricordandosi di anteporre al nome della classe che lo rappresenta, il nome del package `com.pellegrinoprincipe` (tranne se, in alcuni capitoli del libro, è diversamente indicato).
- creare i package nella cartella `MY_JAVA_PACKAGES`;
- creare gli archivi nella cartella `MY_JAVA_JARS;`
- non utilizzare, per la compilazione e l'esecuzione dei programmi, nessun ambiente di sviluppo avanzato tipo Eclipse o NetBeans, poiché utilizzeremo, per i suddetti compiti e a linea di comando, direttamente i comandi `javac` e `java`.

In conclusione, desideriamo sottolineare che:

- quando negli esempi del libro si invocheranno i comandi di compilazione ed esecuzione del codice (o altri comandi, come per esempio `jar`) daremo per assodato il rispetto dei percorsi di directory precedentemente indicati. Ciò significa che, se per esempio invocheremo il comando di compilazione, presupporremo che abbiate già cambiato la corrente directory in `MY_JAVA_SOURCES` e che, in tale percorso, si troverà il file sorgente indicato;
- gli eventuali comandi di shell presentati sono indicati secondo le convenzioni del sistema Windows e, pertanto, per i sistemi GNU/Linux occorre attenersi alle specifiche regole.