Introduzione

Questo libro ha come obiettivo quello di insegnare la programmazione in Java in modo semplice e immediato, andando direttamente 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, quindi, che questo non è un libro leggero o superficiale, poiché affronta tutti gli argomenti che una guida completa si prefigge di trattare a cui sono associati innumerevoli listati e snippet di codice 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 multidimensionali.
- 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 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 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 ad oggetti (basato sulle classi) inteso come estensione del paradigma procedurale (basato sulle procedure). Non a caso, le strutture di controllo della programmazione ad 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": descriviamo come si creano gerarchie di classi (ereditarietà) e che cos'è il polimorfismo. Parliamo, inoltre, di classi astratte e 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 parametrizzati.
- Capitolo 10, "Programmazione funzionale": introduciamo il lettore allo studio del paradigma della programmazione funzionale attraverso un percorso di apprendimento graduale e completo, che parte dall'analisi dei concetti propedeutici quali il lambda calcolo, l'immutabilità dello stato, le closure e così via, e termina con un dettaglio di come i progettisti di Java hanno introdotto i pattern propri di tale paradigma nel linguaggio. Tra gli argomenti: le interfacce funzionali, i metodi di default, le lambda expression, il target typing.

NOTA

In alcuni listati, laddove necessario, si è preferito lasciare l'implementazione delle interfacce funzionali piuttosto che scrivere l'equivalente lambda expression. Ciò per due ragioni: la prima di carattere teorico-didattica, legata a garantire al lettore, in questa fase di studio preliminare, una maggiore comprensione di quello che effettivamente avviene con l'utilizzo delle interfacce funzionali; la seconda di carattere pratico-didattica, connessa a permettere al lettore di esercitarsi a cambiare, in quelle parti di codice dove è presente un'interfaccia funzionale, l'equivalente lambda expression. Quanto detto ha una sola eccezione nel Capitolo 17, dove viene trattata l'astrazione *stream*, che è stata pensata proprio per supportare nativamente le lambda expression; pertanto in quel contesto, nei listati relativi non sono state usate le interfacce funzionali ma direttamente tali lambda expression.

- Capitolo 11, "Errori software": illustriamo come si intercettano e gestiscono gli errori software grazie all'utilizzo di un meccanismo definito *gestione delle eccezioni*.
- Capitolo 12, "Package": vediamo come si creano librerie di tipi correlati e come si rendono disponibili (*deployment*) in un ambiente di sviluppo o di produzione.

- Capitolo 13, "Annotazioni": analizziamo come si creano e utilizzano dei metadati che annotano gli elementi (variabili, metodi e così via) del linguaggio Java.
- Capitolo 14, "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 15, "Caratteri e stringhe": impariamo a utilizzare le classi principali che consentono di manipolare i caratteri e le stringhe (`Character`, `String`, `StringBuilder` e così via).
- Capitolo 16, "Espressioni regolari": spieghiamo cosa sono e come impiegare le *regex* per costruire dei sofisticati pattern per la ricerca di corrispondenze di caratteri.
- Capitolo 17, "Collezioni": studiamo come utilizzare le collezioni, o contenitori (`Collection`, `List`, `Map` e così via), costituite da gruppi di elementi tra loro naturalmente collegati che possono subire operazioni di manipolazione. Introduciamo, infine, allo studio dell'astrazione *stream*, che rappresenta una sequenza di elementi su cui è possibile compiere operazioni aggregate, di tipo *filter-map-reduce*, sia sequenziali sia parallele.
- Capitolo 18, "Programmazione concorrente": trattiamo i concetti di processo e thread e di come scrivere in Java programmi che sono in grado di eseguire più operazioni in parallelo.
- Capitolo 19, "Input/Output: stream e file": illustriamo come effettuare le comuni operazioni di input e output dei dati (stream, file e così via).
- Capitolo 20, "Progettazione di interfacce utente": impariamo a scrivere programmi Java dotati di una GUI (*Graphical User Interface*) mediante l'utilizzo del potente framework Swing.
- Capitolo 21, "Programmazione di rete": studiamo cos'è una rete (suite TCP/IP) e le API che Java mette a disposizione per la programmazione dei socket e dei datagram.
- Capitolo 22, "Programmazione dei database": analizziamo i concetti propedeutici dei database relazionali come le tabelle e le relazioni tra di esse, diamo uno sguardo al linguaggio SQL e studiamo l'architettura software JDBC, che consente di scrivere applicazioni che si interfacciano con una base di dati in modo semplice, consistente e indipendente dal database usato.
- Capitolo 23, "Sviluppo di applicazioni web": impariamo a utilizzare la piattaforma JEE 7 (*Java Enterprise Edition*) per progettare e sviluppare applicazioni 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 in ambiente Windows e in ambiente GNU/Linux.
- Appendice B, "Installazione e configurazione della piattaforma JEE": vediamo come installare e configurare la piattaforma di Java destinata alla progettazione e allo sviluppo di applicazioni web.
- Appendice C, "Installazione e configurazione di MySQL": illustriamo come installare e configurare il database server MySQL sia per sistemi Windows sia per sistemi

GNU/Linux. Analizziamo, infine, come installare e configurare Connector/J, il driver JDBC ufficiale per il server MySQL.

- Appendice D, "Installazione e utilizzo di NetBeans": mostriamo come installare e utilizzare l'IDE (*Integrated Development Environment*) NetBeans per la creazione di applicazioni Java per le piattaforme JSE e JEE.
- Appendice E, "Applet": vediamo come scrivere dei programmi Java denominati *applet*, che girano all'interno di un browser web.

Struttura del libro e convenzioni

Gli argomenti del libro sono, ovviamente, organizzati 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, a supporto 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 ArgomentoImmodificabile" indica il listato di codice numero 2 del Capitolo 6 avente come descrizione la classe `ArgomentoImmodificabile`.
Per quanto attiene ai listati, abbiamo adottato la seguente convenzione: i puntini di sospensione (…) eventualmente presenti indicano che in quel punto sono state omesse alcune parti del listato. 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.

Codice sorgente e classi

All'indirizzo http://www.apogeonline.com/libri/9788850333080/scheda è possibile scaricare un archivio ZIP che contiene tante cartelle quanti sono i capitoli del libro. Ciascuna cartella, denominata Cap01, Cap02 e così via, contiene a sua volta due sottocartelle, denominate Listati e Snippets.
All'interno della cartella Listati sono presenti, in modo indipendente, sia tutti i file sorgente .java del capitolo di pertinenza con le eventuali risorse complementari, sia altre sottocartelle direttamente correlate a specifici progetti caricabili con l'IDE NetBeans, che ricalcano come denominazione, laddove opportuno, i rispettivi file sorgenti.
All'interno della cartella Snippets si trovano invece dei file .txt che contengono gli snippet di codice eventualmente presenti nel capitolo di pertinenza.

Utilizzo dei comandi javac e java per compilare ed eseguire i programmi Java

Per rendere agevole la compilazione e l'esecuzione dei programmi Java, indipendentemente dall'IDE, riteniamo utili i consigli riportati di seguito.
Se si utilizza Windows, creare 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.

Se si utilizza GNU/Linux, creare 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 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, per esempio javac -d C:\MY_JAVA_CLASSES UsoDiChar.java per il sistema operativo Window e javac -d /opt/MY_JAVA_CLASSES UsoDiChar.java per il sistema operativo Gnu/Linux.
Inoltre, se vi sono più classi da compilare, magari perché una classe utilizza i servizi di un'altra classe, si può lanciare il comando di compilazione, come javac -d c:\MY_JAVA_CLASSES Time.java Time_Client.java oppure javac -d /opt/MY_JAVA_CLASSES Time.java Time_Client.java.

Procedere poi come segue.

1. Eseguire il programma dalla cartella `MY_JAVA_CLASSES`, ricordandosi di anteporre il nome del package `com.pellegrinoprincipe` al nome della classe che lo rappresenta (tranne dove diversamente indicato).
2. Creare i package nella cartella `MY_JAVA_PACKAGES`.
3. Creare gli archivi nella cartella `MY_JAVA_JARS`;

In conclusione, sottolineiamo quanto segue.

- 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; per i sistemi GNU/Linux occorre attenersi alle regole specifiche di questo sistema operativo.

Utilizzo dell'IDE NetBeans per compilare ed eseguire i programmi Java

Se lo si desidera, è possibile utilizzare l'IDE NetBeans per editare, compilare, eseguire e "debuggare" il codice sorgente presente nel libro. A tal fine consultare l'Appendice D per una spiegazione in merito all'utilizzo introduttivo di tale IDE e alla creazione e all'impiego dei progetti.
Infine, è importante precisare che tutti i progetti qui presentati presuppongono un'installazione di default del JDK nel percorso `C:\Program Files (x86)\Java\jdk1.8.0` per Windows e `/opt/jdk1.8.0` per GNU/Linux.