CORSO JAVA AESYS
Parte Prima

2022

www.aesystech.it

Stile di codifica, tipi di dati ed array

- Stile di codifica
- Tipi di dati primitivi
- Tipi di dati reference
- Array
- Array multidimensionali

Stile di codifica

Con il termine **stile di codifica** intendiamo una serie di regole obbligatorie e non, che caratterizzano solitamente il codice Java. Delle regole obbligatorie non si può fare ovviamente a meno, ma anche le facoltative sono molto importanti, perché utilizzate dai programmatori di tutto il mondo, e quindi il loro uso semplifica enormemente la leggibilità del codice. Il linguaggio Java:

- ❑ Schema Libero
- ❑ Commenti (a singola **linea, multilinea** e **Javadoc)**
- ❑ Regole per gli identificatori
- ❑ Standard e convenzioni per i nomi
- ❑ **Convenzione per le classi**
- ❑ **Convenzione per le variabili**
- ❑ **Convenzione per i metodi**

Tipi di dati primitivi 1/2

- Tipi di dati interi
  I **tipi di dati interi** sono quattro: byte, short, int e long. Essi condividono la stessa funzionalità (tutti possono immagazzinare numeri interi positivi o negativi) ma differiscono per quanto riguarda il proprio intervallo di rappresentazione. Infatti un byte può immagazzinare un intero utilizzando un byte (otto bit). Il più utilizzato è indubbiamente l'int che è a 32 bit. Meno utilizzati ci sono il byte (8 bit), lo short (16 bit) e il long (a 64 bit). I **tipi a virgola mobile** sono due: float (a 32 bit) e il più utilizzato double (64 bit). Per tutti questi dati esistono varie problematiche (come la **promotion)** per quanto riguardano le operazioni aritmetiche. Il **cast** permette di adattare un certo tipo di dato ad essere trattato come un altro tipo di dato.
- Tipi di dati a virgola mobile
  Per memorizzare i numeri decimali Java utilizza i tipi a **virgola mobile** (in inglese **floating point).** Si parla di virgola mobile perché un numero decimale è possibile rappresentarlo spostando la virgola utilizzando la notazione scientifica. Per esempio, 0,0001 è equivalente a $1 * 10^{-4}$, che con la notazione scientifica si scrive 1E4. I due tipi a virgola mobile che Java supporta con i loro intervalli di rappresentazione (basati sullo standard di decodifica IEEE-754)

Tipi di dati primitivi 2/2

- Il tipo di **carattere logico-booleano**
  boolean, che può assumere come valore solo due literals: true o false. Un **literal** viene definito come un valore contenente lettere o simboli specificati nel codice sorgente invece che al runtime. L'unico tipo di dato primitivo letterale è il **carattere** char, che può essere utilizzato per immagazzinare un singolo carattere incluso tra una coppia di apici singoli. Un carattere è comunque rappresentato da un numero intero ed utilizza lo standard di decodifica Unicode.
- Tipo di dato primitivo letterale
  Il tipo char permette di immagazzinare caratteri (uno per volta).
  Questo tipo di dato viene memorizzato in 2 byte (16 bit) e utilizza lo standard di decodifica dei caratteri noto come **Unicode.** Java 9 supporta la versione 8.0 di Unicode.

Tipi di dati primitivi 2/2

- Il tipo di **carattere logico-booleano**
  boolean, che può assumere come valore solo due literals: true o false. Un **literal** viene definito come un valore contenente lettere o simboli specificati nel codice sorgente invece che al runtime. L'unico tipo di dato primitivo letterale è il **carattere** char, che può essere utilizzato per immagazzinare un singolo carattere incluso tra una coppia di apici singoli. Un carattere è comunque rappresentato da un numero intero ed utilizza lo standard di decodifica Unicode.
- Tipo di dato primitivo letterale
  Il tipo char permette di immagazzinare caratteri (uno per volta).
  Questo tipo di dato viene memorizzato in 2 byte (16 bit) e utilizza lo standard di decodifica dei caratteri noto come **Unicode.** Java 9 supporta la versione 8.0 di Unicode.

Tipi di dati non primitivi: reference

- Il tipo di **dato Reference**
  Un **reference** è il “nome che diamo ad un oggetto”, una particolare variabile che contiene come valore l’indirizzo a cui deve puntare, e che deve conoscere il tipo che andrà a puntare. Quando un oggetto viene istanziato tutte le sue variabili d’istanza vengono automaticamente inizializzate ai propri **valori nulli.**
- Passaggio di parametri
  Il **passaggio di parametri** ad un metodo avviene sempre **per valore**, nel senso che nel momento in cui si passa un valore primitivo ad un parametro di un metodo, verrà copiato nel parametro questo valore. Anche quando si passa un reference verrà passato il suo valore (l’indirizzo del reference) e copiato nel parametro del metodo.
  Quando si istanzia un oggetto tutte le variabili d’istanza vengono inizializzate ai propri valori nulli

Gli array in Java

Un **array** è una collezione indicizzata di dati primitivi, o di reference o di altri array. Gli array permettono di utilizzare un solo nome per individuare una collezione costituita da vari elementi, che saranno accessibili tramite indici interi. In Java gli array sono, in quanto collezioni, oggetti. La sintassi degli array però, differisce da quella degli altri oggetti.

Per utilizzare un array bisogna passare attraverso tre fasi: dichiarazione, creazione ed inizializzazione.

- Dichiarazione

  Di seguito presentiamo due dichiarazioni di array. Nella prima dichiariamo un array di char (tipo primitivo), nella seconda dichiariamo un array di istanze di File (classe appartenente al package java.io):

  char alfabeto []; oppure char [] alfabeto;

  File files []; oppure File [] files;

- Creazione

  Un array è un oggetto speciale in Java e, in quanto tale, va istanziato in modo speciale. La sintassi è la seguente:
  alfabeto = new char[21]; files = new File[3];

- Inizializzazione

  Per inizializzare un array, bisogna inizializzarne ogni elemento singolarmente:

  alfabeto [0] = 'a'; alfabeto [1] = 'b'

L'indice di un array inizia sempre da zero. Quindi un array dichiarato di 21 posti, avrà come indice minimo 0 e massimo 20 (un array di dimensione n implica il massimo indice a n-1)

Array Multidimensionali

Esistono anche array multidimensionali che sono array di array. A differenza della maggior parte degli altri linguaggi di programmazione, in Java un array bidimensionale non deve per forza essere rettangolare. Un esempio:

```
int arrayNonRettangolare [][] = new int[4][];
arrayNonRettangolare [0] = new int[2];
arrayNonRettangolare [1] = new int[4];
arrayNonRettangolare [2] = new int[6];
arrayNonRettangolare [3] = new int[8];
arrayNonRettangolare [0][0] = 1;
arrayNonRettangolare [0][1] = 2;
arrayNonRettangolare [1][0] = 1;
```

oppure, equivalentemente:

```
int arrayNonRettangolare [][] = {
    {1,2},
    {1,0,0,0},
    {0,0,0,0,0,0},
    {0,0,0,0,0,0,0,10}
};
```