

Appunti di Linguaggi di Programmazione

Riccardo Lo Iacono & Stefano Graffeo

Dipartimento di Matematica & Informatica
Università degli studi di Palermo
Sicilia
a.a. 2022-2023

Indice.

1	Introduzione e sintassi base	2
1.1	Commenti	2
1.2	Tipi primitivi	2
1.3	Operatori	4
2	Le classi	5
2.1	Variabili e valori di default	6
2.2	Metodo Main	6
2.3	Compilazione	7
2.4	Garbage Collector	7

– 1 – Introduzione e sintassi base.

Java è un linguaggio semplice, orientato ad oggetti, distribuito, interpretato, robusto, sicuro, indipendente dall'architettura, portabile, performante, dinamico e multi-threaded. Concentrandoci sull'orientamento ad oggetti, esso è un approccio basato su classi contenenti le informazioni nei dati utilizzabili tramite i metodi e servono a creare oggetti. Inoltre ciò permette il riutilizzo nel codice.

I programmi in Java sono divisi in classi descritte in uno o più file di testo con estensione “.java”.

Nota: Un singolo file può avere più classi ma al più una classe può essere marcata come **public**.

Nota: Il punto di accesso di un programma Java è il metodo **main** contenuto nella classe principale.

– 1.1 – Commenti.

Analogamente a quanto vale per il linguaggio C, si hanno

- `//commento`: definisce commenti in riga;
- `/*commento*/`: definisce commenti su più righe;
- `/**commento*/`: definisce la documentazione JavaDoc¹.

– 1.2 – Tipi primitivi.

Come per gli altri linguaggi di programmazione, Java definisce dei tipi di dati:

- **byte**: intero 8 bit con segno;
- **short**: intero 16 bit con segno;
- **int**: intero 32 bit con segno;
- **long**: intero 64 bit con segno;
- **float**: floating-point 32 bit con segno;
- **double**: floating-point 64 bit con segno;
- **boolean**: due elementi (true, false) 8 bit;
- **char**: carattere 16 it (Unicode).

¹Della javaDoc si parlerà a seguito.

– 1.2.1 – Dichiarazione.

Le dichiarazioni avvengono esattamente come nel linguaggio C, in esse quindi si può fare anche un'assegnazione diretta e/o anche dichiarazioni multiple.

Nota: Una dichiarazione non può iniziare con una cifra né si può utilizzare la parola riservata **class**.

– 1.2.2 – Letterali.

Le dichiarazioni dirette risultano simili al linguaggio C o in alcuni casi più specifiche. Vediamo i casi:

- **caratteri:** in Unicode o dichiarazione tra apici;
 - `\n`: avanzamento di riga;
 - `\t`: tabulazione caratteri.
- **stringhe:** racchiuse tra virgolette;
- **interi:**
 - `int`: suffisso assente;
 - `long`: suffisso **l** o **L**.

Inoltre si può definire la base di un intero:

- base decimale: inizia per 0 o un numero che inizia per 1...9 \Rightarrow 0129L;
- base ottale: inizia con 0 \Rightarrow 0777;
- base esadecimale: inizia con 0x \Rightarrow 0x127;
- **reali:** devono necessariamente avere una cifra nella parte intera o frazionaria e almeno un elemento tra punto decimale, esponente o suffisso. Si distinguono in:
 - **float**: suffisso **f** o **F**;
 - **double**: suffisso assente o **d** o **D**;

– 1.3 – Operatori.

Come nel linguaggio C esistono operatori utilizzabili per effettuare confronti o operazioni tra dati. Si distinguono in: 1

- **operatori di relazione:** utilizzabili per tipi numerici e booleani sono:
 - `==`;
 - `!=`.
- **operatore di complemento logico:** utilizzabile per complementare il risultato di un'espressione booleana: `!`;
- **operatori logici:** vengono utilizzati per le espressioni booleane, si dividono in:
 - **AND:** indicato con `&` o se in una condizione con `&&`;
 - **OR:** indicato con `|` o se in una condizione con `||`;
 - **OR esclusivo:** indicato con `^`.
- **confronto numerico:** per confrontare due tipi numerici si usano:
 - `>`;
 - `>=`;
 - `<`;
 - `<=`.

Si può anche usare `==`;

- **incremento e decremento postfisso e prefisso:** come nel linguaggio C si può incrementare o decrementare un valore di 1 senza dovere esprimere l'istruzione completa. Questa operazione può essere espressa prima(prefissa) o dopo(postfissa) la variabile in base alle necessità. Queste operazioni vengono effettuate tramite `++` e/o `--`;
- **complemento bit a bit:** utilizzabile per ottenere il complemento bit a bit di un tipo numerico: `~`;
- **operatore condizionale:** anche detto operatore ternario, funziona come nel linguaggio C, se la condizione è soddisfatta viene effettuata la prima istruzione altrimenti la seconda: `condizione ? istruzione1 : istruzione2`;
- **concatenazione:** concatena una stringa con un'altra stringa o tipo numerico(in questo caso esso viene trasformato in stringa). La concatenazione viene effettuata tramite l'operatore `+`.

– 2 – Le classi.

I file .java devono essere denominati con la classe principale definita al loro interno. Le classi sono modelli per creare e manipolare oggetti. Ogni oggetto ha:

- un'identità (quindi codice identificativo e memoria);
- uno stato (definito da **attributi**);
- un comportamento (definito da operazioni dette **metodi** che ne cambiano lo stato).

Ogni oggetto è istanza di una classe. Per descrivere gli stati possibili di un oggetto, una classe utilizza variabili dette attributi mentre per i comportamenti si definiscono i metodi.

La sintassi di una classe è:

```
<modificatore-visibilita> <NomeClasse>{  
    <dichiarazione_attributi>  
    <dichiarazione_metodi>  
}
```

La sintassi di un metodo, invece, è molto simile a quella di una funzione nel linguaggio C:

```
<modVisibilita> <tipoRitorno> <nomeMetodo>(<parametri>){  
    <corpo>  
}
```

I principali modificatori di visibilità sono **public** e **private** ed è assegnabile sia ai metodi che agli attributi. Inoltre si possono definire degli attributi costanti mediante la parola chiave **final**. Le classi definiscono al loro interno un **costruttore** ovvero un metodo che definisce lo stato iniziale degli oggetti.

Il costruttore di default, ovvero senza parametri, è definito dal compilatore se e solo se non ci sono altri costruttori.

Nota: Utilizzare il modificatore di visibilità **private** per metodi, attributi o classi non permette di accedere ad essi dall'esterno della classe.

– 2.1 – Variabili e valori di default.

In Java le variabili si distinguono in due tipi: di

- **di istanza**: se sono definite come attributi di una classe;
- **locali**: se compaiono solo nei metodi.

Inoltre variabili (locali e di istanza) senza inizializzazione esplicita hanno valore di default:

- `null`: riferimento ad oggetti;
- `0`: tipi numerici;
- `'\u0000'`: per i `char`;
- `false`: tipi booleani.

– 2.2 – Metodo Main.

Per costruire gli oggetti e cambiarne lo stato si usa il metodo `main` dichiarato con:

```
public static void main(String[] args).
```

Nel metodo `main` funge da client per tutte le classi quindi in esso si scrive il programma eseguibile che genererà gli eventuali oggetti. Solitamente, il metodo `main` si mette in un file a parte denominato **Main.java**.

Nel metodo `main` per richiamare metodi o attributi di una classe si utilizza la notazione punto: `<nomeOggetto>.<nomeMetodo>(<parametri>)`.

Esempio:

```
public static void main (String [] args){  
    Serbatoio = new Serbatoio ();  
    s.rifornisci (12);  
    System.out.println ("Il livello è:" + s.getLivello ());  
}
```

– 2.3 – Compilazione.

Per la compilazione e l'esecuzione dei file java si utilizzano rispettivamente:

- `javac <nomeFile>.java;`
- `java <nomeFile>.`

Il primo compila il file `.java` ed in caso di successo genera l'eseguibile `<nomeFile>.class` mentre il secondo esegue appunto il file `.class`.

Nota: Se più classi in un file `.java` contengono un metodo `main` verrà eseguito quello con lo stesso nome del file.

Diversi file java nella stessa cartella possono accedere a tutti i loro attributi e metodi a meno che non siano marcati `private`.

– 2.4 – Garbage Collector.

In Java non è necessario deallocare manualmente la memoria occupata da un oggetto in quanto è dotato di un sistema chiamato **Garbage Collector** che si occupa di liberare la memoria di oggetti privi di riferimento. A proposito di ciò la Java Virtual Machine gestisce un contatore dei riferimenti per ogni oggetto in modo da deallocare memoria qualora tale contatore si azzerasse.