Metodologie di Programmazione

Alessia Cassetta

March 2024

Indice

1 Introduzione

Tipo	Intervallo	Dimensione
double	parte intera: ±10308, parte frazionaria: circa 15 cifre decimali significative	8 byte
long	-9223372036854775808…9223372036854775807	8 byte
int	-2147483648…2147483647	4 byte
float	parte intera: ±1038, parte frazionaria: circa 7 cifre decimali significative	4 byte
boolean	true, false	1 byte
char	Rappresenta tutti i caratteri codificati con Unicode	2 byte
short	-32768…32767	2 byte
byte	-128…127	1 byte

Figura 1: **Python VS Java**

1.1 La programmazione e Generalità

Molto più semplice che apprendere una lingua straniera. Poche parole chiave da imparare. Portabilità del bagaglio: la maggior parte di ciò che imparate in Java, potete applicarlo in Python, C, in C++ o nella maggior parte degli altri linguaggi di programmazione.

E' fondamentale apprendere a programmare (a oggetti), non (solo) apprendere Java:

- sapere quali dettagli sono rilevanti in quale situazione
- modellare la realtà mediante l'utilizzo di oggetti
- astrarre e generalizzare per poter riutilizzare

Polimorfismo: E' possibile utilizzare la classe base, senza dover conoscere necessariamente la classe specifica di un oggetto. Permette di scrivere codice che non dipende dalla classe specifica. Posso aggiungere nuove sottoclassi anche in seguito!

1.2 Perché Java come linguaggio di riferimento?

Un linguaggio di programmazione potente, orientato agli oggetti. Creato da Sun Microsystems (ora Oracle).

Precursori: Smalltalk (fine ‘70), C++ (inizio ‘80). Costruito per essere “sicuro”, cross-platform e internazionale.

Continuamente aggiornato con nuove feature e librerie.

Multithreaded: Supporta nativamente programmi che gestiscono attività eseguite in contemporanea (thread). Facilita la costruzione di applicazioni interattive.

Interpretato e compilato: il bytecode è tradotto “al volo” in istruzioni macchina native. Rende più veloce e snello il processo di sviluppo.

2 Tipi di dato di base (o primitivi)

I tipi di dati di base sono built-in, ovvero sono predefiniti nel linguaggio. È fondamentale essere a conoscenza di quali siano i tipi di base e quali non lo siano. Per ragioni di efficienza e di allocazione della memoria.

Un tipo di dati è un insieme di valori e di operazioni definite su tali valori: Interi (es. 27), Reali (es. 27.5) Booleani (true o false), Caratteri (es. ‘a’), Stringhe (es. “questa non è una stringa”).Una variabile è un nome usato per riferirsi a un valore di un tipo di dati (es. contatore).

Tipo	Intervallo	Dimensione
double	parte intera: ±10308, parte frazionaria: circa 15 cifre decimali significative	8 byte
long	-9223372036854775808…9223372036854775807	8 byte
int	-2147483648…2147483647	4 byte
float	parte intera: ±1038, parte frazionaria: circa 7 cifre decimali significative	4 byte
boolean	true, false	1 byte
char	Rappresenta tutti i caratteri codificati con Unicode	2 byte
short	-32768…32767	2 byte
byte	-128…127	1 byte

Figura 2: Enter Caption

Una variabile è creata mediante una dichiarazione:

```
int contatore;
```

Il valore viene assegnato a una variabile mediante una assegnazione:

```
contatore = 0;
```

Un'istruzione può includere una dichiarazione e una assegnazione allo stesso tempo:

```
int contatore = 0;
```

In Java siamo costretti a specificare il tipo della variabile.
Questo tipo non può più cambiare (ovvero è statico). Tuttavia, esistono alcuni elementi di dinamicità (che vedremo più avanti). Non posso utilizzare una variabile senza prima dichiararla e non posso assegnare a una variabile tipi incompatibili tra loro.

Il nome assegnato a una variabile è detto identificatore, ovvero una sequenza di lettere, cifre, _ e $ la prima delle quali non è una cifra. Gli identificatori sono case-sensitive. Non possono essere utilizzate alcune parole riservate (es. public, static, int, double, ecc.). Si utilizza la notazione a cammello (Camel case). Le variabili devono avere un **nome sensato**.

Esempio di Codice

```
int a, b;
a = 5;
b = a + 10;
int c = a+b;
a = c - 3;
```

2.1 I letterali

Un **letterale** è una rappresentazione a livello di codice sorgente del valore di un tipo di dati. Ad esempio: 27 o -32 sono letterali per gli interi; 3.14 è un letterale per i double; true o false sono gli unici due letterali per il tipo booleano; "Ciao, mondo" è un letterale per il tipo String.

2.2 Distinguere tra costanti intere e in virgola mobile

Interi:

- Le costanti di tipo int sono semplicemente numeri nell'intervallo [-2, +2] miliardi circa.
- Le costanti di tipo long vengono specificate con il suffisso l o L (ad esempio, 1000000000000L).

Numeri in virgola mobile:

- Le costanti di tipo double sono semplicemente numeri con la virgola (punto).
- Le costanti di tipo float hanno il suffisso f o F (ad esempio, 10.5f).

In tutti i casi si può utilizzare il trattino basso (_) per separare le cifre (es. 100_000 per indicare 100000, 1_234 per indicare 1234 ecc.). Si può ottenere un intero da una rappresentazione binaria anteponendo 0b alla stringa binaria (ad esempio, 0b101 vale 5).

2.3 Altre generalità

Espressioni: è un letterale, una variabile o una sequenza di operazioni su letterali e/o variabili che producono un valore.

Assegnazione di un'espressione a una variabile: supponiamo di voler effettuare l'assegnazione:

```
1 c = a*2+b
```

1) Calcola il valore dell'espressione destra (5*2+15).
2) Assegna il valore (25) alla variabile di destinazione (c).

2.4 Conversioni di tipo

Conversione Esplicita: utilizzando un metodo che prende in ingresso un argomento di un tipo e restituisce un valore di un altro tipo. Integer.parseInt(), Double.parseDouble(), Math.round(), Math.floor(), Math.ceil() ecc...

Cast esplicito: anteponendo il tipo desiderato tra parentesi **(int)2.71828** produce un intero di valore 2. Se il tipo di partenza è più preciso (es. double), le informazioni aggiuntive vengono eliminate nel modo più ragionevole (es. da double a int viene eliminata la parte frazionaria).

Cast implicito: se il tipo di partenza è meno preciso, Java converte automaticamente il valore al tipo più preciso: double d = 2; **Attenzione: la somma di due caratteri dà un intero!**

2.4.1 Regole per il cast implicito

Il cast implicito avviene in fase di assegnazione:

- byte, short e char possono essere promossi a int
- int può essere promosso a long
- float può essere promosso a double

O in fase di calcolo di un'espressione: se uno dei due operandi è double, l'intera espressione è promossa a double. Altrimenti, se uno dei due operandi è float, l'intera espressione è promossa a float.

3 Classi e Oggetti

Possono:

- Modellare gli oggetti del mondo reale
- Rappresentare oggetti grafici
- Rappresentare entità software (file, immagini, eventi, ecc.).
- Rappresentare concetti astratti (regole di un gioco, posizione di un giocatore).
- Rappresentare stati di un processo, di esecuzione, ecc...

Una classe è un pezzo del codice sorgente di un programma che describe un particolare tipo di oggetti.
Le classi vengono definite dal programmatore e forniscono un prototipo astratto per gli oggetti di un particolare tipo. Ne definisce la struttura in termini di **attributi** (stato degli oggetti) e di **metodi** (comportamenti) degli oggetti.

La **classe** è definita mediante parte del codice sorgente del programma ed è implementata dal programmatore.
Specifica la struttura degli attributi dei suoi oggetti e il comportamento dei suoi oggetti tramite metodi.

Un **oggetto** è un entità della classe programmata in esecuzione e viene creata quando un programma ”gira” nel main o in un altro metodo.
Contiene specifici valori dei campi che possono cambiare durante l'esecuzione. Inoltre, si comporta nel modo prescritto dalla classe quando il metodo corrispondente viene chiamato a tempo di esecuzione.

3.1 Classi e file sorgenti

Ogni classe è memorizzata in un file separato. Il nome del file deve essere lo stesso della classe con estensione .java
I nomi di classe iniziano sempre con una maiuscola e sono case-sensitive.

3.1.1 Librerie

I programmi Java normalmente non sono scritti da zero ma esistono migliaia di classi di librerie per ogni esigenza. Le classi sono organizzate in **package**.

3.2 Attributi e Metodi

Un **attributo** costituisce la memoria privata di un oggetto e ha un tipo di dato. Ogni campo ha un nome fornito dal programmatore.

Ad esempio:

```
public class Hotel{
    private int k;
    private String nome;
    private int numeroStanze;
}
```

I **metodi** sono tipicamente pubblici, ovvero visibili a tutti. Il nome del metodo per convenzione inizia con una lettera minuscola, mmentre le parole seguenti iniziano con una lettera maiuscola (camelCase).

Ad esempio:

```
public String testMethod(paramtee){
    //instructions
}
```

In questo modo ho definito una classe che restituisce una stringa ed ottiene in input dei parametri.
Come vengono chiamati i metodi:

```
public void reset(){value = 0;}
public void reset(int valueIn){value = valueIn;}
static public void main(String[], args){
    Counter contatore1 = new Counter(42);
    contatore1.count();
    contatore1.reset();   // chiama il primo metodo reset
    contatore1.reset(10); //chiama il secondo
}
```

Quando troviamo la parola **void** nell'intestazione del metodo, indica che il metodo non restituisce alcun valore.

3.3 Costruttori

Servono ad inizializzare gli attributi dell'oggetto e posseggono lo stesso nome della classe. Non hanno valori di uscita. Non è obbligatorio specificare un costruttore dato che Java crea per ogni classe un costruttore di default "vuoto".

```
public Counter(int valueIn){
    value = valueIn;
}
```

```
4 static public void main(String[], args){
5     Counter contatore1 = new Counter(42);
6 }
```