DMIF, Università degli Studi di Udine

I costrutti imperativi e array in Java

Emanuele Scapin, Ph.D. Student
scapin.emanuele@spes.uniud.it

14 Giugno 2021

Indice

Una definizione

La **programmazione imperativa** è un paradigma di programmazione secondo cui un programma viene inteso come un insieme di istruzioni, ciascuna delle quali può essere pensata come un "ordine" che viene impartito alla macchina virtuale del linguaggio di programmazione utilizzato.

Caratteristiche principali di Java

- Java è un linguaggio object oriented basato su: classi, oggetti, metodi; ingloba comunque costrutti imperativi.
- Un linguaggio orientato agli oggetti: i dati sono rappresentati come oggetti e le operazioni come metodiche operano su essi.
- Pensato per lo sviluppo di applicazioni in rete.
  - Semplice
  - Robusto
  - Indipendente dalla piattaforma
  - Sicuro (?)
- Sintassi simile a **C** e **C++**.

Assegnazione

Esempi

```
int x = 5;

float f = 0.75;

double d = 3.2;

boolean t = true; // oppure false

char c = 'z';
```

Selezione - IF

If

```
if ( espressione booleana )
   istruzione1
else
   istruzione2
```

Selezione - IF

Esempio

```
if ( x < y ) {
   y = y - x;
} else {
   x = x - y;
}
```

Oppure

Esempio

```
if ( x < y ) {
    y = y - x;
}
```

Iterazioni - While e Do-While

While

```
while (espressione booleana)
    istruzione
```

Do-While

```
do
    istruzione
while (espressione booleana)
```

Le condizione complesse si definiscono utilizzando le operazioni booleane: && AND, || OR, ! NOT,

Iterazioni - While e Do-While

Esempio

```
while ( x < y ) {

    x = x + m;

}
```

Iterazioni - For

For

```
for(espressione di inizializzazione; espressione booleana;
espressione di incremento) {
    istruzione
}
```

Iterazioni - For

Esempio

```
for ( int i=1; i<n; i=i+1 ) {

    v[i-1] = v[i];

}
```

```
int a[] = new int[10];
```

La creazione fa una inizializzazione implicita, con valore 0 per int e double, false per i boolean.

Assegnazione

```
int x[] = new int[10];
int y[] = new int[50];

x[3] = y[5];
```

Copia di un array

Se si vuole copiare un array in un altro:

```
int vet1[] = new int[20];
int vet2[] = new int[20];

for(int i=0; i< 20; i++)
    vet1[i] = vet2[i];
```

Copia di un parte di array

Se si vuole copiare una parte di array in un altro:

```
int vet1[] = new int[20];
int vet3[] = new int[10];

for(int i=0; i< 10; i++)
    vet3[i] = vet1[i];
```

Copia la prima metà di vet1 in vet3.

Copia in ordine inverso

Se si vuole copiare gli elementi di vet1 in vet2 in ordine inverso:

```
int vet1[] = new int[20];
int vet2[] = new int[20];

for(int i=0; i< 20; i++)
    vet2[i] = vet1[19-i];
```

Lunghezza di un array

Ad ogni variabile array è associata implicitamente una variabile di istanza **length**:

```
int [] seq = new int[5];
int len = seq.length;
```

la variabile len avrà valore **5**.

```
for(int i=0; i<vet1.length; i++)
    vet1[i] = vet2[i];
```

Array di caratteri (char)

Si possono dichiarare array di caratteri (char).

Dichiarazione e assegnazione

```
char[] v = { 's', 't', 'o', 'p' };
```

Il carattere va indicato tra apici singoli (').

```
int c = 0;
for(int i=0; i<v.length; i++) {
    if (v[i] == 'a')
        c++; // equivalente a c = c+1;
}
```

La variabile c conta il numero di caratteri uguali ad 'a'.

La classe String

Si possono dichiarare oggetti String in vario modo.

Dichiarazione e assegnazione

```
String str = new String();
String str = "abc"; // equivale a String str = new String("abc");
String str = ""; // stringa vuota
```

Le stringhe vanno indicate tra doppio apice (").

Il metodo **int length()** della classe String calcola la lunghezza della stringa, ovvero il numero di caratteri che compongono la stringa.

```
"buongiorno".length(); // vale 10
"".length();   // vale 0

String str = "abc";
System.out.println(str.length()); // stampa 3
```

l metodo **char charAt(int pos)** della classe String permette di estrarre da una stringa il carattere che occupa una certa posizione.
La posizione fornita deve essere compresa tra 0 e length()-1.

```
"buongiorno".charAt(3); // restituisce 'n'

String str = "abc";
System.out.println(str.charAt(0)); // stampa 'a'
System.out.println(str.charAt(2)); // stampa 'c'
```

Import di librerie

La libreria contiene componenti per la gestione dell'input, la programmazione del database e molto altro ancora.
La libreria è divisa in pacchetti e classi. Ciò significa che puoi importare una singola classe (insieme ai suoi metodi e attributi) o un intero pacchetto che contiene tutte le classi che appartengono al pacchetto specificato.
Per utilizzare una classe o un package dalla libreria, è necessario utilizzare la parola chiave **import**:

```
import java.util.*;    // importa tutto il package
import java.util.Date; // importa la classe Date

import campus_toolkit.*;
```

La procedura

Un metodo in Java si dice procedura quando non restituisce alcun risultato, ma esegue delle azioni.

```
public static void mioMetodo() {
   // codice da eseguire
}
```

- **mioMetodo** è il nome del metodo
- **static** significa che il metodo appartiene alla classe e non a un oggetto della classe
- **void** significa che questo metodo non restituisce un valore

Per chiamare il metodo: *mioMetodo();*

La funzione

Un metodo in Java si dice funzione quando restituisce un valore.

```
public int add(int a, int b) {
    return a+b;
}
```

- **add** è il nome del metodo
- **int** significa che restituirà come risultato un valore intero

```
int c = add(a,b);
System.out.println(add(a,b)); // stampa
```