# Ripetizioni Filippo

Marini Mattia

2025



## Indice



## Esercizi

# 1 Java

Java è molto simile dal punto di vista della sintassi al c++. Non sarà molto complicato il passaggio

## 1.1 Aspetti simili a c++

La sinttassi di Java è molto simile a quella di c++, ecco gli aspetti che rimangono invariati o quasi:

### 1.1.1 Dichiarazione delle variabili:

| Sinstassi | Differenze rispetto al `c++` |
|---|---|
| `int x = 15` | invariato |
| `long x = 15` | invariato |
| `float x = 15.0f` | nota il `15.0f`, dove `f` sta per float |
| `double x = 15.0` | float ma con precision maggiore, 64 bit |
| `boolean x = true` | `boolean` anzichè `bool` |
| `String s = "stringa"` | dato `String` sarebbe una classe ma è trattato come tipo primitivo, dato che è usato molto frequentemente |
| `String v[] = new String[15]` | vettore di stringhe di dimensione 15 |
| `int v[] = new int[15]` | vettore di interi di dimensione 15 |

### 1.1.2 Commenti:

- Commento riga singola: `// commento`
- Commento righe multiple: `/* commento */`

### 1.1.3 Cicli:

- Ciclo for:

```
for(int i = 0; i<15; i++){
  // qualcosa
}
```

- Ciclo while:

```
while(i < 15){
  // qualcosa
}
```

### 1.1.4 Operatori

| Operatore | Descrizione |
|---|---|
| `+` `-` `*` `/` | operatori matematici |
| `&&` | and logico |
| `\|\|` | or logico |
| `!` | not logico |

### 1.1.5 If e switch

```
if(a<b) {
  //codice
}
else(if a>b) {
  //codice
}
else {
  //codice
}
```

```
switch(espressione) {
  case x:
    // codice
    break;
  case y:
    // codice
    break;
  default:
    // codice
}
```

### 1.1.6 Funzioni

```
void nome_funzione1 (int arg_1, String arg_2){
  //corpo funzione
}

int nome_funzione2 (int arg_1, String arg_2){
  //corpo funzione
  return 5;
}
```

Una funzione è quindi definita indicando nel seguente ordine, esattamente come in c++:

1. Tipo di ritorno (o `void` se non ritorna nulla)
2. Nome della funzione
3. Parametri, racchiusi fra parentesi tonde e separati da virgole
4. Corpo della funzione fra graffe

## 1.2 Differenze

### 1.2.1 Stampa su terminale

Una delle feature usata moltissimo, ma completamente diversa dal c++ è la stampa su terminale:

```
System.out.println("Stampa questa cosa");
//stampa andando a capo prima di stampare

System.out.print("Stampa questa cosa");
//stampa SENZA andando a capo
```

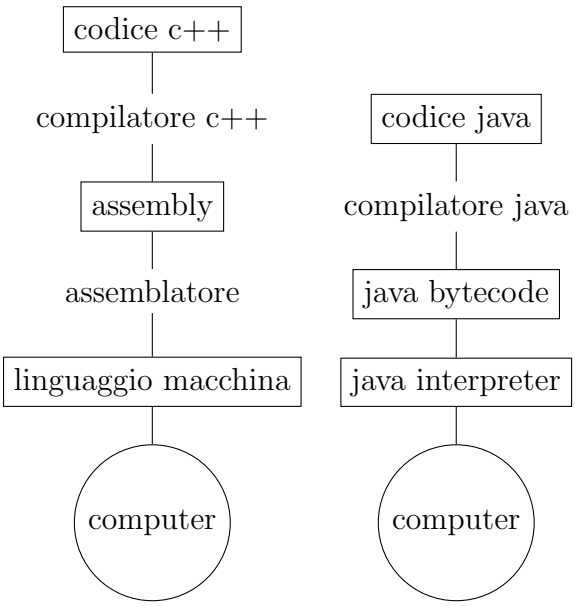
nota che le stringhe si possono concatenare con l'operatore +:

```
String a = "Hello";
String b = "world";

System.out.println(a + b);
//stampa "Hello world"

String c = a + b;
//Inizializza c a "Hello world"
```

### 1.2.2 Linguaggio interpretato vs compilato

A differenza di c++, java è un linguaggia interpretato

- *Linguaggio compilato*: il codice è "dato in pasto" a un compilatore, il quale lo converte in linguaggio macchina (di fatto in una sequenza di, 0 ed 1)
- *Linguaggio interpretato*: il codice è "dato in pasto" ad un compilatore, il quale lo converte però in bytecode, ossia un linguaggio di basso livello (molto difficile da leggere e scrivere), il quale è in grado di essere letto da un *interprete*




### 1.2.3 Memoria e puntatori

Java è un linguaggio ad alto livello che gestisce la memoria in maniera diversa rispetto al c++:

- c++: il compito di allocare e deallocare la memoria non più utilizzata è del programmatore
- java: la memoria viene deallocata in maniera automatica tramite un meccanismo chiamato garbage collection

Visto che in java la memoria è gestita in in maniera automatica, il programmatore non ne ha accesso diretto tramite puntatori: al contratio, i puntatori non esistono

### 1.2.4 Object orientation

Sebbene c++ sia un linguagio che permette di utilizzare classi ed oggetti, in java l'object orientation è forzata: ogni parte del programma deve essere contenuta all'interno di una classe

## 1.3 Esempio programma

```
class HelloWorld {
    public static void main(String[] args) {
        System.out.println("Hello, World!");
    }
}
```

Cose da notare:

- La funzione main è contenuta all'interno di una classe "HelloWorld", il cui nome è arbitrario
- La funzione main è marcata come `static`, ciò vuol dire che la funzione esiste anche se non esiste un oggetto di tipo "HelloWorld", affroteremo meglio il modificatore `static` più avanti, per ora possiamo ignorarlo
- La funzione main è marcata come `public`, ciò vuol dire che la funzione è accessibile ovunque. Afffronteremo meglio questo modificatore più avanti, per ora possiamo ignorarlo
- La funzione main prende in ingresso un vettore di stringhe. Nel caso si avviasse l'applicazione da terminale e possibile passare al main dei parametri nel seguente modo:

  ```
  cd cartella_applicazione
  ./nome_applicazione parametro_1 parametro_2 ...
  ```

  in questo caso il vettore di stringhe `args` conterrà `parametro_1` e `parametro_2`. Penso non lo userete mai ma è buono saperlo

## 1.4 Funzioni utili

In java sono definite alcune funzioni utilissime. Qui una lista (non esaustiva) delle più comuni:

### 1.4.1 Stringhe

Supponiamo di avere `String s = "stringa stringa";`

| Funzione | Descrizione |
|---|---|
| `s.lenght()` | Ritorna il numero di caratteri conenuti nella stringa (7 nel caso d'esempio) |
| `s.charAt(int index)` | Ritorna il carattere in posizione `index` |
| `s.indexOf(char carattere)` | Ritorna l'indice della prima occorrenza di `carattere` in `s` |
| `s.indexOf(String stringa)` | Ritorna l'indice della prima occorrenza della sotto-stringa `stringa` in `s` |

```
String s = "stringa stringa";
s.lenght(); // 15
s.charAt(2); // 'r'
s.indexOf('r'); // 2
s.indexOf("ga"); // 5
```

### 1.4.2 Vettori

Supponiamo di avere `int v[] = new int[15];`

| Funzione | Descrizione |
|---|---|
| `v.lenght` | ritorna il numero di elemnti contenuti nel vettore |

## 1.5 Parti utili della libreria standard

La libreria standard di java offre moltissime classi utili. Vediamo qui le più comuni

### 1.5.1 Math

| Funzione | Descrizione |
|---|---|
| `Math.exp(float n)` | Ritorna $e^n$ |
| `Math.log(float n)` | Ritorna $ln\,(n)$ |
| `Math.abs(float x)` | Ritorna $\|x\|$ (valore assoluto di $x$) |
| `Math.sin(float x)` | Ritorna $\sin(x)$ |
| `Math.cos(float x)` | Ritorna $\cos(x)$ |
| `Math.tan(float x)` | Ritorna $\tan(x)$ |
| `Math.asin(float x)` | Ritorna $\arcsin(x)$ |
| `Math.acos(float x)` | Ritorna $\arccos(x)$ |
| `Math.atan(float x)` | Ritorna $\arctan(x)$ |
| `Math.max(float a, float b)` | Ritorna l'elemento maggiore fra $a$ e $b$ |
| `Math.min(float a, float b)` | Ritorna l'elemento maggiore fra $a$ e $b$ |
| `Math.floor(float x)` | Arrotonda per difetto $x$ |
| `Math.ceil(float x)` | Arrotonda per eccesso $x$ |
| `Math.round(float x)` | Arrotonda $x$ |

### 1.5.2 ArrayList

Utile per avere un vettore di lunghezza variabile. Si inizializza nel seguente modo:

```
ArrayList< tipo > nomeArray = new ArrayList< tipo >(dimensione)
```

Nota che:

- `dimensione` si può omettere, ottenendo un vettore con dimensiane nulla
- `tipo` deve essere una classe. Se voglio un arrayList di tipi primitivi devo utilizzare le classi wrapper (`Integer, Boolean, Double` ...)

| Metodo | Descrizione |
|---|---|
| `v.get( index )` | ritnorna l'elemento all'indice `index` |
| `v.set( index , element)` | setta l'elemento a indice `index` |
| `v.add( element )` | inserisce `element` in fondo |
| `v.remove ( index )` | rimuove l'elemento a indice `index` |
| `v.size()` | ritorna il numero di elementi conetnuti |
| `v.clear()` | rimuove tutti gli elementi |

### 1.5.3 Scanner

La classe scanner ci permette di leggere input utente da terminale (e anche da file, ma non ci servirà). Uno scanner si inizializza così:

```
Scanner nomeScanner = new Scanner(System.in);
```

Per leggere l'input da terminale ci sono i seguenti comandi:

| Metodo | Descrizione |
|---|---|
| nextBoolean( ) | Legge un boolean da terminale |
| nextByte() | Legge un byte da terminale |
| nextDouble() | Legge un double da terminale |
| nextFloat () | Legge un float da terminale |
| nextInt() | Legge un int da terminale |
| nextLine() | Legge una String da terminale |
| nextLong( ) | Legge un long da terminale |
| nextShort() | Legge uno short da terminale |

Il metodo più comune è `nextLine()`, dato che ci restituisce l'intera riga come stringa, anche se contiene numeri

### 1.5.4 Random

Random fornisce un modo comodo per generare numeri casuali. Inizializza con

```
Random nome = new Random()
```

| Metodo | Descrizione |
|---|---|
| nextInt( range ) | genera un numero casuale nel range $[0, \text{range})$ |
| nextFloat() | genera un float in range $[0.0, 1.0]$ |
| nextDouble() | genera un double in range $[0.0, 1.0]$ |

### 1.5.5 LinkedList

Lista linkata. Utilizzabile in modo molto simile al `ArrayList`. L'accesso agli elementi è molto più lento, rimozioni inserimenti sono molto più veloci

### 1.5.6 Altre strutture dati utili

- `HashSet`: insieme matematico, possibile vedere se un elemento è contenuto in esso in maniera efficiente
- `Map`: utile poter collegare ad ogni valore un altro valore detto chaive. Si può risalire al valore tramite la chiave in maniera efficiente
- `Stack`
- `Queue`
- `PriorityQueue`: struttura nella quale è possibile accedere all'elemento maggiore in maniera efficiente
- `SortedSet`: struttra nella quale i dati mantengono sempre un ordinameto crescente. Non ammette duplicati

## 1.6 Esercizi

Collezione di esercizi, divisi per categoria

**Esercizio 1:** *Hello World (hello_world)*

Scrivere un programma in Java che prenda in input una stringa `s` e un numero `n` e stampi `s` `n` volte, accompagnata dal numero di riga. Ad esempio, dati in input `userin` e `4`, l'otput sarà:

```
1. userin
2. userin
3. userin
4. userin
```

**Esercizio 2:** *Hello classe (estrazione)*

Crea un programma per giocare a una sorta di roulette modificata. Le regole sono le seguenti:

- Vengono estratte delle palline che hanno un colore (`rosso`, `giallo` o `verde`) e un numero da 1 a 9 estremi compresi
- Il giocatore deve indicare un numero e un colore
- Il punteggio viene assegnato così:
  - Colore giusto: `+1`
  - Un punteggio che varia da 0 a 4 in base alla distanza del numero previsto da quello estratto, ad esempio, detto $n_e$ il numero *estratto* e $n_p$ il numero *previsto*
    $$\left(1 - \frac{|n_e - n_p|}{8}\right) \cdot 4$$
  - Se colore e numero sono uguali: `+1`

Il programma deve chiedere in input all'utente un numero e un colore ed estrarre una pallina casualmente, per poi stampare in ouput il punteggio ottenuto. Si usi una classe `ball` e una classe `game` per gestire le partite.

Il programma deve partire chiedendo all'utente il numero $n$ di round che vuole giocare. Dopodiché, verranno chieste $n$ previsione all'utente; per ognuna deve essere stampato il punteggio e alla fine degli $n$ round deve essere visualizzato il punteggio totale

### Esercizio 3: *Geometria 1 (geometry1)*

Creare una classe per ciascuno dei seguenti oggetti geometrici: `Punto`, `Rettangolo`, `Cerchio`. Creare una classe `PianoCartesiano` che possa contenere le tre classi elencate precedentemente. Una volta inserita una classe nel piano cartesiano verrà ritornato un `id`, tramite il quale possiamo accedervi (l'istanza dell'oggetto aggiunto rimane scollegata da quella inserità nel piano cartesiano). Creare un metodo `closer` che prenda un `id` esistente e ritorni l'id della forma geometrica più vicina (usare il centro delle figure per confrontare la distanza)

### Esercizio 4: *Geometria 1 (geometry1)*

Creare una classe per ciascuno dei seguenti oggetti geometrici: `Punto`, `Rettangolo`, `Cerchio`. Creare una classe `PianoCartesiano` che possa contenere le tre classi elencate precedentemente. Una volta inserita una classe nel piano cartesiano verrà ritornato un `id`, tramite il quale possiamo accedervi (l'istanza dell'oggetto aggiunto rimane scollegata da quella inserità nel piano cartesiano). Creare un metodo `closer` che prenda un `id` esistente e ritorni l'id della forma geometrica più vicina (usare il centro delle figure per confrontare la distanza)

### Esercizio 5: *Montecarlo (montecarlo)*

Il casinò di Montecarlo ha richiesto una nuova slot machine; lo scopo del gioco è quello di simulare la pesca di *cinque carte* da una mazzo standard (con quattro semi, le carte da 2 a 10, l'asso(1) le tre figure (11,12,13) e niente jolly)

Le opportunità di vincita (con quattro delle 5 carte estratte) sono le seguenti:

| Combinazione | vincita |
|---|---|
| Poker di figure | 1000 € |
| Poker d'assi | 1000 € |
| [a] Colore (5 carte dello stesso seme) | Somma delle carte uscite × 8 |
| [b] Scala colore(5 carte dello stesso seme) | Somma delle carte uscite × 10 |

La presenza di un'altra carta non intacca la vincita.

Contare quante slotmachine sono presenti nel casino

Si scrivano le classi opportune per implementare il gioco e un classe `Casino` dove create diverse `Slot`.

Fare in modo che la giocata possa essere efffetutata solo dopo aver inseirto un certo importo;

Introdurre successivamente anche i *jolly* tra le carte e fare in modo che la presenza del jolly come carta estratta raddoppi la vincita;

[a] Es: 1-5-10-13 → vincita = 224 €
[b] Es: 3-4-5-6 → vincita = 180 €