

---

---

# **PROGRAMMAZIONE I**

---

---

**LAUREA TRIENNALE IN SCIENZE INFORMATICHE**

Magliani Andrea  
Perego Luca

Università degli studi di Milano-Bicocca

A.A. 2022/2023

# INDICE

<b>Codice → Linguaggio Macchina.....</b>	<b>3</b>
1.1 Componenti.....	3
1.2 Vantaggi & Svantaggi.....	4
<b>Errori.....</b>	<b>4</b>
2.1 Errori Sintattici.....	4
2.2 Errori di Run-Time.....	4
2.3 Errori Logici.....	4
<b>Variabili.....</b>	<b>4</b>
3.1 Introduzione.....	4
3.2 Tipi.....	5
3.3 Cast.....	5
<b>String.....</b>	<b>6</b>
4.1 Definizione.....	6
4.2 Struttura.....	6
<b>Flusso di controllo.....</b>	<b>6</b>
5.1 Gestione del flusso di controllo.....	6
5.2 Blocco di codice.....	6
<b>Costrutti Condizionali.....</b>	<b>7</b>
6.1 Definizione.....	7
6.2 If.....	7
6.3 Switch.....	7
<b>Cicli.....</b>	<b>7</b>
7.1 Introduzione.....	7
7.2 Tipi di cicli.....	8
<b>Metodi.....</b>	<b>8</b>
8.1 Main.....	8
8.2 Struttura di un Metodo.....	8
8.3 Record di Attivazione.....	9
8.4 Utilizzi non canonici.....	9
<b>Array.....</b>	<b>9</b>
9.1 Introduzione.....	9
9.2 Struttura.....	9
<b>Ricorsione.....</b>	<b>10</b>
10.1 Definizione.....	10
10.2 Stack.....	10

# Codice → Linguaggio Macchina

## 1.1 Componenti

- **Compilatore:** Prende in input il codice sorgente e dà come output il codice in linguaggio macchina, da eseguire successivamente.
- **Interprete:** Prende come input il codice sorgente e lo traduce un comando alla volta, eseguendolo gradualmente.

## 1.2 Vantaggi & Svantaggi

Java è sia compilato che interpretato.

Il compilatore Java traduce il codice in **bytecode**, che viene poi eseguito dalla **Java Virtual Machine**, permettendo di eseguire un codice compilato su qualsiasi computer.

La JVM si occupa poi di tradurre il bytecode in Linguaggio Macchina.

# Errori

## 2.1 Errori Sintattici

La sintassi del linguaggio di programmazione presenta errori;

## 2.2 Errori di Run-Time

Errori che accadono durante l'esecuzione del codice;

## 2.3 Errori Logici

Il codice viene eseguito correttamente ma non dà l'output desiderato, si tratta di un errore nell'implementazione dell'algoritmo;

# Variabili

## 3.1 Introduzione

Una **variabile** è una posizione di memoria identificata da un' identificatore, che contiene valori di un unico tipo ed è utilizzabile solamente in un determinato **scope**.

La variabile deve essere **dichiarata** prima di essere utilizzata, specificando il tipo e assegnando un identificatore.

## 3.2 Tipi

Il tipo determina il valore che la variabile può assumere e le operazioni che possono essere effettuate su esso. Le variabili possono essere **primitive** o **non-primitive**.

Un valore può essere assegnato ad una variabile del suo tipo o alla sua destra in questa sequenza:

byte → short → int → long → float → double

## 3.3 Cast

è possibile cambiare il tipo di una variabile temporaneamente tramite il **typecast**.

# String

## 4.1 Definizione

Una **String** è una sequenza di char trattati come unico elemento. String è un tipo non-primitivo.

## 4.2 Struttura

String è una **classe**, ma funziona in modo caratteristico, in quanto può essere dichiarato come variabile.

In quanto classe, String dispone di diversi metodi per manipolare i suoi oggetti.

Essendo un array di char, è possibile accedere ai **singoli char** della stringa.

# Flusso di controllo

## 5.1 Gestione del flusso di controllo

Il **flusso di controllo** è l'ordine in cui un programma svolge le istruzioni.

Un'istruzione di selezione (**branching statement**) sceglie tra due o più azioni possibili.

Un ciclo (**loop**) ripete un'azione fino a che non viene soddisfatta una condizione di stop.

## 5.2 Blocco di codice

sequenza di comandi racchiusa tra parentesi. I blocchi possono essere annidati. Le variabili dichiarate all'interno di un blocco possono essere utilizzate solo al loro interno.

# Costrutti Condizionali

## 6.1 Definizione

L'istruzione `if` offre una **selezione multiramo** che permette di avere un blocco di codice eseguito solo se vengono avverate delle condizioni predefinite.

## 6.2 If

start → valutazione dell'espressione booleana → `if true`, istruzione 1, `if false` istruzione 2

Il ramo `else` può essere omesso. L'istruzione `if-else` può essere annidata in sé stessa.

## 6.3 Switch

L'istruzione `switch` offre una selezione multiramo alternativa all'`if`, utilizzando come condizione un intero, un carattere o una stringa.



# Cicli

## 7.1 Introduzione

I cicli sono costrutti che permettono di **ripetere** un blocco di codice.

Il blocco di codice è detto **body** e ogni ripetizione di quest'ultimo è detta **iterazione**.

## 7.2 Tipi di cicli

Esistono 3 costrutti che permettono cicli:

```
while (espressione) {  
    //istruzioni  
}
```

```
do {  
    //istruzioni  
} while (espressione);
```

```
for (inizializzazione; condizione; aggiornamento) {  
    //istruzioni  
}
```

# Metodi

## 8.1 Main

I metodi sono blocchi di codice eseguiti solamente se richiamati nel main. Il **main** è il metodo eseguito all'avvio della classe.

## 8.2 Struttura di un Metodo

- **Intestazione:** definisce nome, valore di return e parametri.
- **Parametri attuali:** il valore effettivo dell'argomento.
- **Parametri formali:** una o più variabili dichiarate nell'intestazione. All'invocazione ogni parametro viene inizializzato, questo processo è detto **chiamata per valore**.

## 8.3 Record di Attivazione

Il **record di attivazione** contiene tutte le informazioni relative ad un metodo.

Questa è una struttura dati che contiene: parametri, variabili locali, indirizzo di rientro e return.

Viene creato dinamicamente alla chiamata del metodo e posto in cima allo stack.

Viene gestito tramite politica **LIFO** (Last In First Out)

## 8.4 Utilizzi non canonici

**Driver:** programmi molto semplici per testare la funzionalità dei metodi.

**Stub:** prototipo semplificato del metodo da inserire nel codice per testarlo.

# Array

## 9.1 Introduzione

Un array è un **oggetto** che contiene una sequenza di variabili distinguibili tramite la loro posizione, detta **indice**.

La dichiarazione degli array avviene tramite l'operatore `new`.

## 9.2 Struttura

Una variabile array contiene l'**indirizzo di memoria** in cui l'array è memorizzato (**reference**). Tutte le operazioni sulla reference di un array utilizzano il suo indirizzo di memoria e non il suo contenuto.

Gli array possono essere **multidimensionali**, dichiarandoli come `array[riga][colonna]`

# Ricorsione

## 10.1 Definizione

È detto **ricorsivo** un algoritmo che contiene una versione ridotta dell'algoritmo completo.

## 10.2 Stack

Ogni **chiamata ricorsiva** crea un nuovo record di attivazione in cima allo **stack**.

Questo ha un limite di dimensione, che se superato porta allo **stack overflow**.