

Eco-design Digitale di Base per i servizi ICT

Programmazione in Java

Massimo Giaccone, Luglio 2025

Cos'è un linguaggio di programmazione?

*“Un **linguaggio di programmazione** è un sistema di notazione per la scrittura di **programmi** per **computer**. La maggior parte dei linguaggi di programmazione sono **linguaggi formali** basati su testo...”*

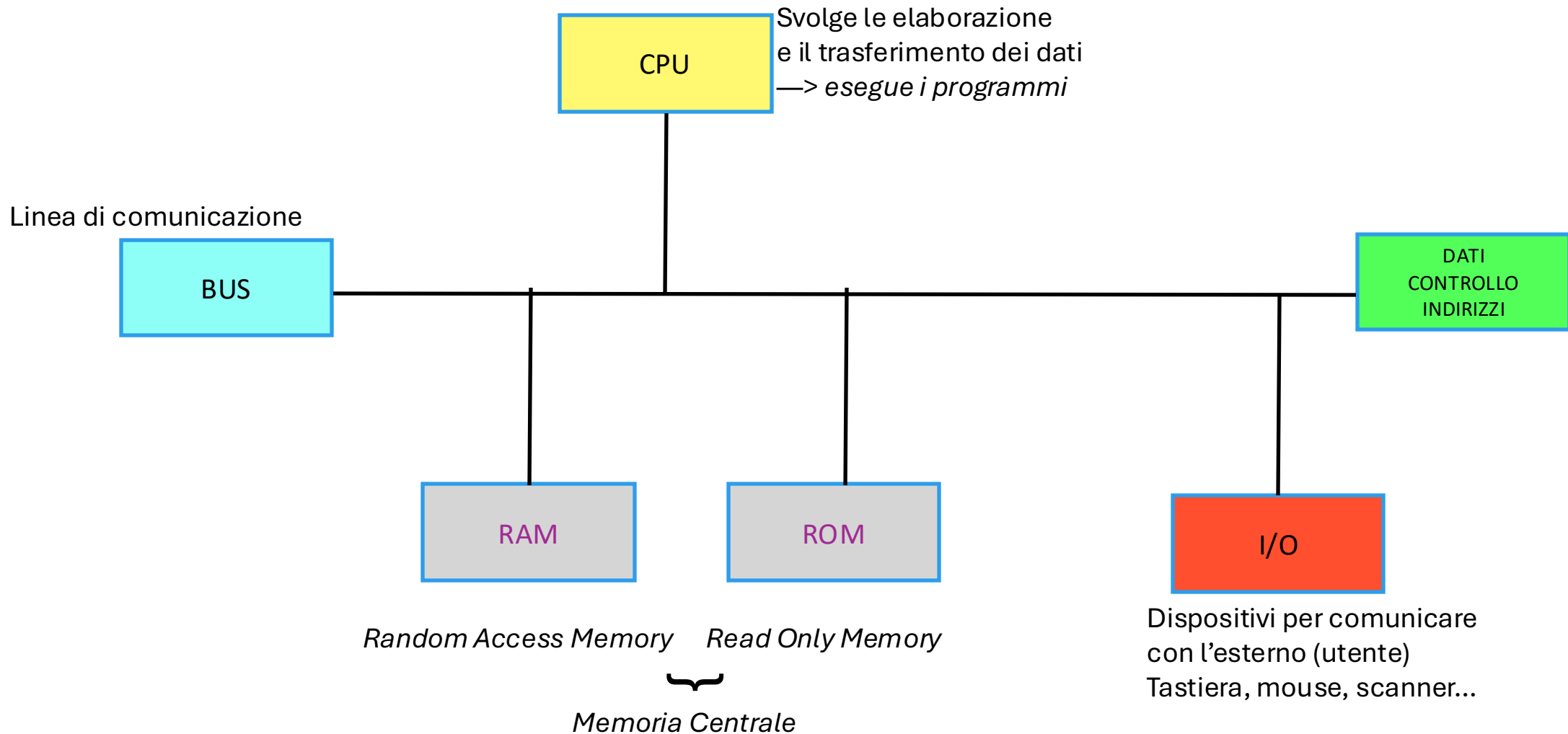
[Fonte: Wikipedia]

Programma: insieme ordinato di istruzioni scritte in un linguaggio interpretabile da un computer per esprimere un algoritmo in grado di risolvere dei problemi.

```
0h init:      LHI R30, 0x4000      ; set R30 = 0x40000000h
4h           SW R29, 0x0000(R30)   ; save R29 in 0x40000000h (RAM)
8h           SW R28, 0x0004(R30)   ; save R28 in 0x40000004h (RAM)
Ch           LHI R29, 0xC000      ; set R29 = 0xC0000000h (STARTUP address)
10h          LBU R28, 0x0000(R29)   ; read STARTUP signal into R28
14h          BEQZ R28, handler     ; if STARTUP == 0 then jump to (interrupt) handler
18h          SB R0, 0x0004(R29)     ; set STARTUP = 0
1Ch          J main                ; jump to main:
20h handler:  LHI R29, 0x3000      ; set R29 = 0x30000000h (INPUT_PORT address)
24h          LBU R28, 0x0004(R29)   ; read interrupt INPUT_PORT signal into R28
28h          BNEZ R28, input_port   ; if INT_I != 0 then jump to (interrupt) input_port
2Ch          LHI R29, 0x9000      ; set R29 = 0x90000000h (LED address)
30h          SB R0, 0x0004(R29)     ; switch LED signal
34h          LW R28, 0x0004(R30)    ; restore R28 value from memory (RAM)
38h          LW R29, 0x0000(R30)    ; restore R29 value from memory (RAM)
3Ch          RFE
40h
44h
48h main:     ADDI R1,R0,0x0000     ; Fibonacci sequence
4Ch          ADDI R2,R0,0x0001     ; set R1 = 0
50h          ADDI R3,R0,0x0001     ; set R2 = 1
54h          ADDI R4,R0,0x0014     ; set R3 = 1
58h loop:     ADD R1,R2,R0          ; set counter R4 = 0x14
5Ch          ADD R2,R3,R0          ; copy R2 into R1
60h          ADD R3,R2,R1          ; copy R3 into R2
64h          SUBI R4,R4,0x0001     ; R3 = R2 + R1
68h          BEQZ R4, loop         ; decrease R4 by 1
```

Esempio di codice Assembly

Computer oggi



Un po' di storia...

The Evolution Of Computer Programming Languages



Hex



Assembler



C



Fortran



C++



Java



Ruby

Paradigmi di programmazione

- **Procedurale:** Il codice è suddiviso in funzioni/procedure. (*divide et impera*)
- **Strutturata:** ogni algoritmo può essere scritto con sequenze, selezioni e iterazioni.
- **Orientata agli oggetti (OOP):** Si basa su *classi* e *oggetti* (istanze delle classi).
- **Dichiarativa:** si concentra sul “cosa” fare più che sul “come”.
 - **Programmazione funzionale:** funzioni matematiche pure.
 - **Programmazione logica:** basata sulla logica del primo ordine.

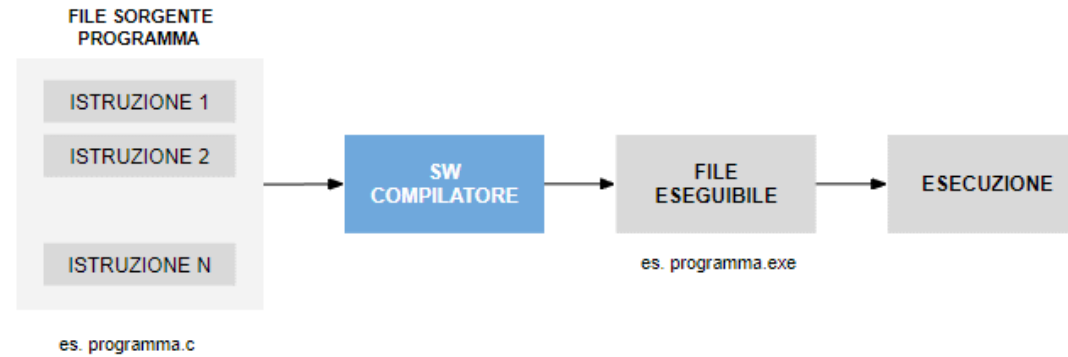
Obiettivi del corso

- Sintassi di base e strutture di controllo
- Programmazione orientata agli oggetti: classi, oggetti, ereditarietà, polimorfismo
- Gestione delle eccezioni e input/output
- Strutture dati: liste, mappe, set
- Best practice per la scrittura di codice manutenibile

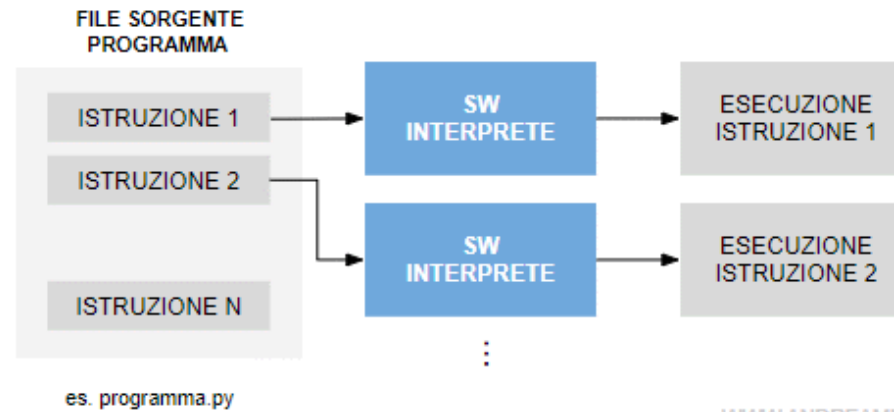
Struttura del corso

- 2 luglio dalle ore 14:00 alle ore 18:00
- 7 luglio dalle ore 14:00 alle ore 18:00
- 15 luglio dalle ore 14:00 alle ore 18:00
- 21 luglio dalle ore 14:00 alle ore 18:00
- 23 luglio dalle ore 14:00 alle ore 18:00
- 25 luglio dalle ore 14:00 alle ore 18:00

Compilato vs Interpretato



WWW.ANDREAMININI.COM



WWW.ANDREAMININI.COM

Linguaggio Java

- Cos'è?

Un linguaggio di programmazione ad alto livello, **orientato agli oggetti con una forte tipizzazione statica**.

- **Tipizzazione statica:** Il tipo delle variabili è noto a tempo di compilazione e non può cambiare.
- **Portabilità:** WORA (Write Once, Run Everywhere)

Architettura Java

- Java è progettato per essere un linguaggio **indipendente dalla piattaforma**, grazie a una **architettura multilivello**:
 1. **Sorgente Java (.java)**: scritto dal programmatore
 2. **Compilazione**: il compilatore javac traduce in **bytecode** (.class)
 3. **JVM**: esegue il bytecode, rendendo Java portabile
- **“Write once, run anywhere”**

Significa che lo stesso .class può essere eseguito su qualsiasi macchina dotata di **Java Virtual Machine**.

JVM (Java Virtual Machine)

La **JVM** è un'interprete + ambiente di esecuzione per il bytecode Java. È responsabile di:

- **Caricamento** delle classi (ClassLoader)
- **Verifica** del bytecode (sicurezza)
- **Esecuzione** (interprete e/o compilazione Just-In-Time - JIT)
- **Gestione della memoria** (heap, stack)
- **Garbage Collection** automatica
- **Gestione delle eccezioni** in tempo di esecuzione

JVM (Java Virtual Machine)

Componenti principali:

- **Class Loader Subsystem**

- Carica classi dal file system, rete, JAR, ecc.
- Evita di caricare due volte la stessa classe (class caching)

- **Runtime Data Areas**

- **Heap:** oggetti e dati dinamici
- **Stack:** chiamate ai metodi, variabili locali
- **Method Area:** metadati di classi e metodi
- **Program Counter (PC) Register:** traccia l'istruzione corrente

JVM (Java Virtual Machine)

Componenti principali:

- **Execution Engine**

- Interprete (linea per linea)
- JIT Compiler (ottimizzazione runtime)

- **Garbage Collector**

- Rimuove oggetti non più referenziati automaticamente

Ciclo di vita programma Java

- **1. Editing:** il programmatore scrive il codice usando un editor
- **2. Compilazione:** il compilatore converte il codice in bytecode generando un file .class

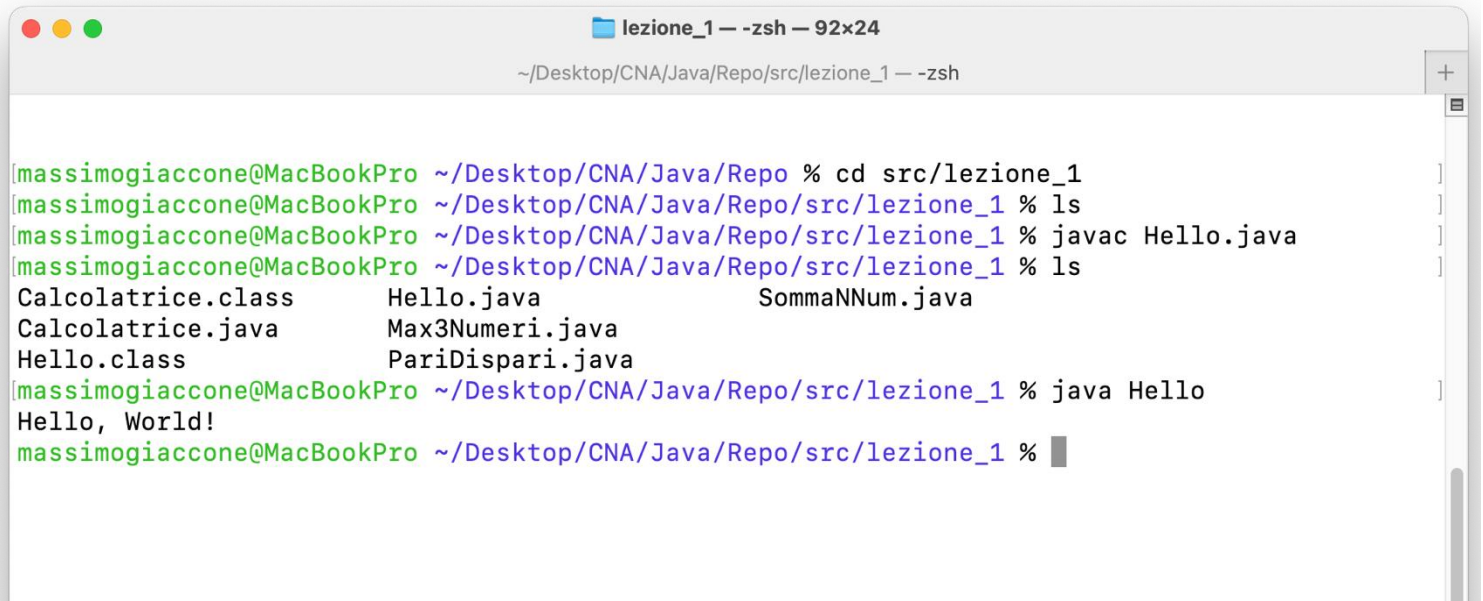
Ciclo di vita programma Java

- **3. Caricamento in JVM:** Il ClassLoader carica il file in memoria
- **4. Verifica del bytecode:** la JVM garantisce sicurezza

Ciclo di vita programma Java

- **5. Esecuzione:** Interpreti legge e esegue istruzioni bytecode: il **JIT compiler** traduce blocchi usati spesso in **macchina nativa**
- **6. Raccolta automatica dei rifiuti (GC)**
Libera memoria non più usata

Esempio:



```
lezione_1 — -zsh — 92x24
~/Desktop/CNA/Java/Repo/src/lezione_1 — -zsh

[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo % cd src/lezione_1
[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo/src/lezione_1 % ls
[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo/src/lezione_1 % javac Hello.java
[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo/src/lezione_1 % ls
Calcolatrice.class      Hello.java              SommaNNum.java
Calcolatrice.java       Max3Numeri.java
Hello.class             PariDispari.java
[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo/src/lezione_1 % java Hello
Hello, World!
[massimogiaccone@MacBookPro ~/Desktop/CNA/Java/Repo/src/lezione_1 % ]
```


Vantaggi utilizzo JVM

- **Portabilità**
Bytecode identico su ogni piattaforma (Windows, Linux, macOS)
- **Sicurezza**
La verifica del bytecode impedisce molte vulnerabilità (buffer overflow, accessi illegali)
- **Gestione automatica della memoria**
Lo sviluppatore non deve gestire malloc o free, riducendo i bug
- **Ottimizzazione runtime**
Grazie al JIT compiler, le performance migliorano durante l'esecuzione
- **Supporto multilinguaggio**
La JVM può eseguire anche linguaggi diversi da Java (Scala, Kotlin, Groovy)

JVM, JDK e JRE

- **JDK (Java Development Kit)**: strumenti per compilare e sviluppare
- **JRE (Java Runtime Environment)**: librerie + JVM
- **JVM (Java Virtual Machine)**: esegue bytecode .class

Installiamo l'editor

Utenti Windows/Mac

- Installare prima il JDK (noi useremo 17) successivamente installare l'IDE:
- Eclipse (più completo, può gestire più file insieme)
- Visual Studio Code (più leggero, con compilazione da terminale ed estensioni)

Come testiamo il compilatore?

1. Aprire Terminale/Prompt dei comandi
2. Digitare `java --version`
3. Creare un file con estensione `.java`
4. Compilare con il comando:
`javac nome_del_file.java`
5. Lanciare da terminale il comando:
`java nome_del_file`

IDE/Compilatori online

- <https://www.onlinegdb.com/>
- <https://www.mycompiler.io/it/>

Ciao mondo!

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

Tipi primitivi Java

Tipo	Dimensione	Range	Esempio
byte	8 bit	da -128 a 127	byte b = 100;
short	16 bit	da -32.768 a 32.767	short s = 30000;
int	32 bit	da -2^{31} a $2^{31} - 1$ ($\sim \pm 2$ miliardi)	int i = 123456;
long	64 bit	da -2^{63} a $2^{63} - 1$	long l = 9_000_000_000L;

Tipi primitivi Java

Tipo	Dimensione	Range approx.	Precisione	Esempio
float	32 bit	$\sim \pm 3.4e38$	~7 cifre decimali	float f = 3.14f;
double	64 bit	$\sim \pm 1.7e308$	~15 cifre decimali	double d = 3.14159;

Tipo	Dimensione	Descrizione	Esempio
char	16 bit	Carattere Unicode (0 - 65.535)	char c = 'A';
boolean	1 bit*	Valori true / false	boolean b = true;

Lessico Java

- I **commenti** servono a documentare il codice e non vengono eseguiti.

```
1- /*****
2
3  Welcome to GDB Online.
4  GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
5  C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite,
6  Code, Compile, Run and Debug online from anywhere in world.
7
8  *****/
9  public class Main
10 {
11     public static void main(String[] args) {
12         System.out.println("Hello World");
13     }
14 }
15
```

Lessico Java

- Una **costante** è un valore che non cambia durante l'esecuzione del programma.

```
1- /*****
2
3 Welcome to GDB Online.
4 GDB online is an online compiler and debugger tool for C, C++, Python, Java, PHP, Ruby, Perl,
5 C#, OCaml, VB, Swift, Pascal, Fortran, Haskell, Objective-C, Assembly, HTML, CSS, JS, SQLite, Prolog.
6 Code, Compile, Run and Debug online from anywhere in world.
7
8 *****/
9
10- public class Main{
11
12     final static int numero = 10;|
13-     public static void main(String[] args) {
14         System.out.println(numero);
15     }
16 }
17
```

Lessico Java

- **Assegnazione di variabili:**

- `int x = 10`
- `String nome = "Alice"`
- `float prezzo = 4.99`
- `bool is_active = True`

Lessico Java

- **Operatori aritmetici:**
 - **$a + b$ # addizione**
 - **$a - b$ # sottrazione**
 - **$a * b$ # moltiplicazione**
 - **a / b # divisione**
 - **$a ** b$ # elevamento a potenza**
 - **$a \% b$ # modulo (resto della divisione)**

Lessico Java

- **Operatori logici:**
 - **true && false # AND logico**
 - **true || false # OR logico**
 - **!true # NOT logico**
- **Operatori di confronto:**
 - **a == b # uguale**
 - **a != b # diverso**
 - **a > b # maggiore**
 - **a >= b # maggiore o uguale**

Lessico Java

```
int eta = 18;  
if (eta >= 18){  
    System.out.println("Maggiorenne");  
    System.out.println("Può votare");}  
else  
    System.out.println("Minorenne");
```

Lessico Java

for (int i = 0; i < 5; i++) {...}

while (condizione) {...}

Lessico Java

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

Java vs C vs Python

Concetto	Java	C	Python
Tipizzazione	Statica, forte	Statica, debole	Dinamica
Gestione memoria	Garbage collector	Manuale (malloc/free)	Automatica
OOP	Centrale	Non nativa	Sì, ma flessibile
Sintassi	Verbosa	Compatta	Molto compatta

Esempio

```
public class Somma {  
    public static void main(String[] args) {  
        int a = 5, b = 3;  
        int risultato = somma(a, b);  
        System.out.println("Risultato: " + risultato);  
    }  
  
    public static int somma(int x, int y) {  
        return x + y;  
    }  
}
```