

# JAVA

## Introduzione

Java è un linguaggio di alto livello e orientato agli oggetti, creato dalla **Sun Microsystem** nel 1995.

Le motivazioni, che guidarono lo sviluppo di Java, erano quelle di creare un linguaggio semplice e familiare. Esso è stato pensato anche per essere integrato in Internet (applet java: piccole applicazioni eseguibili nelle pagine Web).

Le caratteristiche del linguaggio di programmazione Java sono:

- ✚ La tipologia di linguaggio **orientato agli oggetti** (ereditarietà, polimorfismo, ...)
- ✚ la **gestione della memoria** effettuata automaticamente dal sistema, il quale si preoccupa dell'allocazione e della successiva deallocazione della memoria (il programmatore viene liberato dagli obblighi di gestione della memoria come nel Pascal e nel linguaggio C++).
- ✚ la **portabilità**, cioè la capacità di un programma di poter essere eseguito su piattaforme diverse senza dover essere modificato e ricompilato.

Con il termine piattaforma si intende l'insieme formato dall'architettura hardware del computer e il sistema operativo utilizzato (Windows, Linux, ...).

Se si esegue la compilazione di un programma C/C++ su una piattaforma Windows, quello che si ottiene è un codice eseguibile legato alla piattaforma su cui è stato compilato. Questo perché contiene delle istruzioni eseguibili solo da quel particolare processore e sfrutta le utilità di quello specifico sistema operativo. Un programma compilato in questo modo girerà sui sistemi Windows ma non su piattaforme Linux.

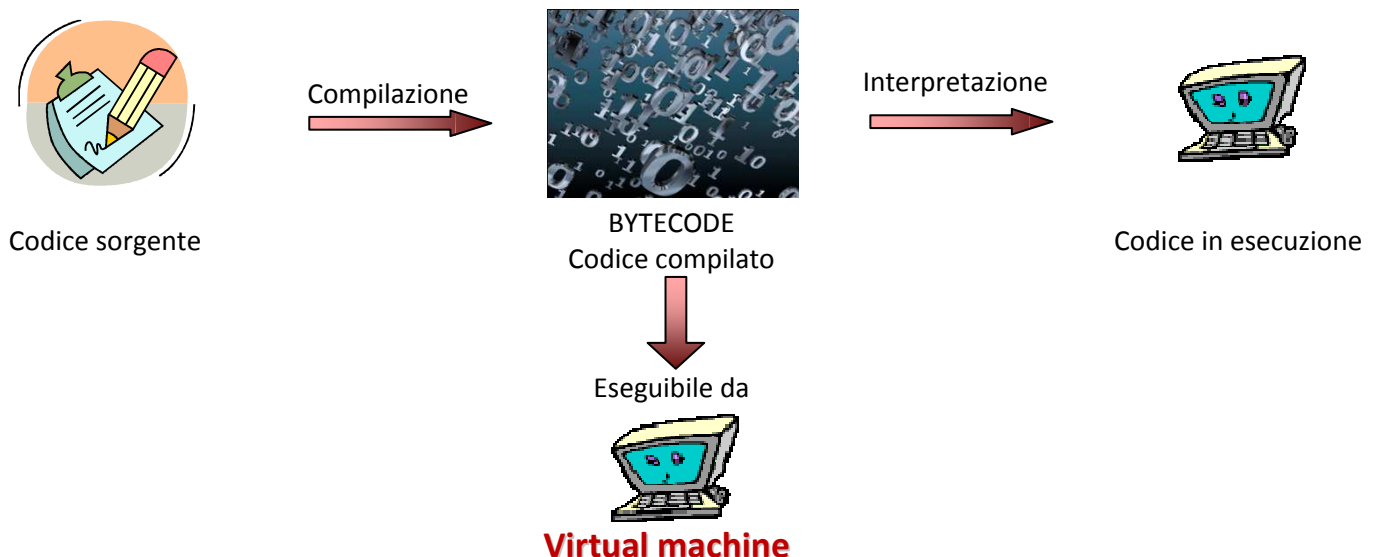
Per far eseguire il programma anche sul sistema operativo Linux occorre ricompilare il programma con un compilatore che sia in grado di generare un codice eseguibile Linux. Durante questa operazione possono essere necessarie delle modifiche al codice sorgente.

Un programma che richiede tutte queste modifiche per poter essere eseguito su una piattaforma diversa, è detto non portabile.

Al contrario un'applicazione Java, una volta scritta e compilata, non ha bisogno di subire le operazioni di portabilità. Tale applicazione può essere eseguita senza modifiche su sistemi operativi diversi ed elaboratori con architetture hardware diverse.

## Funzionamento della portabilità di Java

La portabilità di Java opera nel seguente modo:



Il codice sorgente scritto in Java è inizialmente compilato in un formato intermedio, indipendente dall'architettura del computer, denominato **bytecode**.

Questo codice tuttavia, non è un codice eseguibile da un computer reale, come un qualsiasi altro codice eseguibile. Esso può essere eseguito solo da una particolare macchina astratta, detta **Virtual Machine**.







Per poter essere eseguito su una macchina reale, il bytecode deve essere prima interpretato. Questa operazione viene effettuata dalla Virtual Machine, la quale traduce ogni istruzione nella corrispondente istruzione macchina del computer che si sta utilizzando.

A seconda dell'elaboratore su cui il programma viene eseguito, si ha un'interpretazione diversa del bytecode.

Per tali motivi il Java è considerato un linguaggio interpretato (anche se inizialmente il bytecode è realizzato con un'operazione di compilazione).

L'ambiente di programmazione Java include un insieme di librerie, in continuo aggiornamento, contenenti classi e metodi di varia utilità per facilitare lo sviluppo delle applicazioni.





Le principali librerie sono:

-  **java.lang**: collezione delle classi di base (sempre inclusa)
-  **java.io**: libreria per la gestione degli accessi ai file e ai flussi di input e output
-  **java.awt**: libreria contenente le classi per la gestione dei componenti grafici (colori, font, bottoni, finestre)
-  **java.net**: supporto per creare applicazioni che si scambiano dati attraverso la rete
-  **java.util**: classi di utilità varie (array dinamici, gestione della data, struttura di dati stack)
-  ...

Esistono numerosi strumenti e ambienti di sviluppo per il linguaggio Java: il più famoso è il **JDK** (**J**ava **D**evelopment **K**it), distribuito gratuitamente dalla Sun Microsystem, con il quale si possono scrivere e compilare applet e applicazioni Java.

Il JDK è un ambiente senza interfacce grafiche: il compilatore e l'interprete sono eseguiti da riga di comando. Con il sistema operativo Windows, per compilare ed eseguire un'applicazione Java, occorre ricorrere al **Prompt** di **MS-DOS**.

Per poter programmare in Java è necessario occorre possedere un adatto ambiente di sviluppo. Gli strumenti fondamentali sono:

-  un editor di testi (per esempio Blocco note di Windows)
-  un compilatore e un interprete Java (compresi nel JDK)
-  un browser Web abilitato ad eseguire le applet
-  la documentazione contenente la specificazione delle API (Application Programming Interface), cioè le librerie di classi che sono fornite insieme con il JDK.

Se si installa il JDK, per esempio, nella directory **C:\Programmi\Java\jre7**, il compilatore e l'interprete verranno posizionati nella directory **C:\Programmi\Java\jre7\bin**.

Per evitare di fare sempre riferimento a questa directory ogni volta che si chiede l'esecuzione del compilatore, bisogna modificare la variabile di ambiente PATH delle Variabili d'ambiente (vedi dispensa al seguente link: <http://www.mimmocorrado.it/inf/programmazione/java/dispense/installare%20java.dos.pdf>).

Per programmare in Java in maniera più spedita occorre utilizzare un **IDE** (**I**ntegrated **D**evelopment **E**nvironment), cioè un ambiente che facilita lo sviluppo dei programmi attraverso un'interfaccia grafica e una modalità di programmazione visuale. Uno di questi software è TextPad, disponibile all'indirizzo:

<http://www.mimmocorrado.it/inf/programmazione/java/TextPad.Ita.542.exe>

Per l'installazione di Textpad leggere la breve guida disponibile all'indirizzo:

<http://www.mimmocorrado.it/inf/programmazione/java/dispense/installare%20java.pdf>

## Struttura di un programma Java

Un'applicazione può essere costituita da una o più classi. Tra tutte le classi che compongono un'applicazione, una è indispensabile, perché contiene il metodo **main**. L'esecuzione di un'applicazione Java inizia con questo metodo.

Un semplice programma Java, formato da una sola classe, assume la seguente struttura:

```
class NomeClasse          // NomeClasse deve coincidere con il nome del file memorizzato sul disco
{
    public static void main(String args[])
    {
        // dichiarazioni di variabili
        // istruzioni
    }
}
```

le parentesi graffe delimitano l'inizio e la fine di un blocco di istruzioni.

### Esempio n°1 – Saluto

```
class Saluto
{
    public static void main(String args[])          //INIZIO class
    {
        System.out.print("Buongiorno a tutti,");    //INIZIO main
        System.out.println("iniziamo a programmare in Java"); //Stampa e non va a capo
        System.out.println("");                   //Stampa e va capo
        System.out.println("Buon lavoro !!!");
    }
}
//FINE main
//FINE class
```

Il metodo main è stato dichiarato usando le parole chiave **public**, **static** e **void** che specificano alcune sue proprietà:

- 🔧 public indica che il metodo è pubblico ed è visibile;
- 🔧 void indica che non ci sono valori di ritorno;
- 🔧 static indica che il metodo è associato alla classe e non può essere richiamato dai singoli oggetti della classe

Dopo il nome del metodo, tra parentesi, sono indicati i parametri. Il metodo main possiede come parametro un array di stringhe, **args [ ]**, che corrisponde ai parametri passati dalla riga di comando quando viene eseguita l'applicazione.

Java è **case-sensitive**. Questo significa che le stringhe Base, base e BASE identificano tre variabili diverse.

In un programma Java ogni istruzione termina con un **punto e virgola**.

Per facilitare la comprensione del programma è conveniente utilizzare l'indentazione e il commento delle istruzioni.

L'indentazione si attua diversificando l'incolonnando delle istruzioni.

Per inserire un commento su una sola riga occorre far precedere il commento dalla doppia barra **//**.

Per inserire un commento su più righe occorre inserire ad inizio commento **/\*** e a fine commento **\*/**.

## Esecuzione di un programma Java

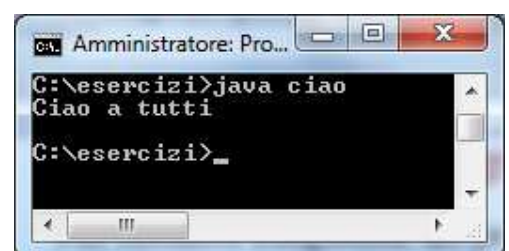
Per rendere eseguibile un programma scritto in Java occorre:

1. editare il codice Java con un editor di testi qualsiasi (Blocco note, Wordpad, ...)
2. salvare il codice sorgente editato assegnando:
  - il nome della classe (il nome inserito dopo la parola `class`)
  - l'estensione `.java`.
3. compilare il programma, digitando dal Prompt del DOS il comando: `javac nomeClasse.java`

Se la fase di compilazione ha avuto successo viene generato un file di nome `nomeClasse.class` (esso rappresenta il bytecode).

Per eseguire il programma occorre digitare dal Prompt del DOS il comando: `java nome Classe`

In quest'ultima fase l'interprete Java prende il codice compilato (bytecode) e lo traduce in codice macchina, quindi lo esegue.



## Gli identificatori

Gli identificatori sono i nomi che possono essere attribuiti alle variabili, ai metodi e alle classi.

Un identificatore è composto da una sequenza di lettere e di numeri. Il primo carattere deve essere una lettera.

Per facilitare la leggibilità di un programma è buona norma utilizzare degli identificatori significativi.

Le convenzioni che solitamente si utilizzano per i nomi degli identificatori sono:

- ✚ gli identificatori che si riferiscono al nome di attributi, di metodi e di oggetti cominciano con la prima lettera minuscola
- ✚ gli identificatori che specificano il nome delle classi iniziano con la lettera maiuscola
- ✚ se gli identificatori sono formati da più parole, queste vengono unite e ogni parola successiva alla prima ha l'iniziale maiuscola (per esempio, area Rettangolo, conta Numeri, ...)

Gli identificatori non possono corrispondere ai nomi delle parole chiave del linguaggio.

PAROLE CHIAVE DI JAVA					
abstract	const	finally	int	public	throw
boolean	continue	float	interface	return	throws
break	default	for	long	short	transient
byte	do	goto	native	static	try
case	double	if	new	super	void
catch	else	implementa	package	switch	volatile
char	extends	import	private	synchronized	while
class	final	instanceof	protected	this	

## Tipi di dato numerici

Il tipo di dato stabilisce qual è l'insieme di valori e operazioni accettabili da una variabile.

### Tipi interi

TIPO	DIMENSIONE	INTERVALLO VALORI
byte	8 bit	-128 , 127
short	16 bit	-32768 , 32767
Int	32 bit	-2 147 483 648 , -2 147 483 647
long	64 bit	$-2^{63}$ , $2^{63} - 1$

### Tipi razionali

TIPO	DIMENSIONE	INTERVALLO VALORI
float	32 bit	$\pm 1,4 \dots \cdot 10^{-45}$ , $\pm 3,4 \dots \cdot 10^{+38}$ (da 6 a 7 cifre significative)
double	64 bit	$\pm 4,94 \dots \cdot 10^{-324}$ , $\pm 1,79 \dots \cdot 10^{+308}$ (da 14 a 15 cifre significative)

## Variabili e costanti

Le variabili sono i contenitori dove si memorizzano i dati.

Gli elementi che caratterizzano una variabile sono: il tipo, il nome e la sua visibilità.

La dichiarazione di una variabile ha la seguente sintassi:

Dichiarazione di una variabile	
Sintassi	Esempio
<code>tipo nomeVariabile</code>	<code>int base</code> <code>byte numeroLotto</code> <code>double altezza=1,83</code>

Una variabile può essere dichiarata in qualsiasi punto del programma, ma a seconda della sua posizione, cambia la sua visibilità. Il campo di visibilità di una variabile è costituito dal blocco dentro al quale è stata dichiarata. All'esterno di questo blocco non viene riconosciuta.

Non è possibile dichiarare altre variabili con lo stesso nome, anche all'interno di blocchi annidati.

Per leggibilità conviene raggruppare le dichiarazioni di tutte le variabili nella parte iniziale del programma.

Una costante, a differenza della variabile, può assumere un solo valore durante tutta l'esecuzione del programma.

Per dichiarare una costante occorre far precedere la dichiarazione dalla parola chiave **final**.

Dichiarazione di una costante	
Sintassi	Esempio
<code>final tipo nome = valore</code>	<code>final double PI_GRECO = 3.14</code>

## Il casting

Il casting è un'operazione che converte un tipo di dato in un altro.

Il casting si realizza antepoendo, tra parentesi tonde, al valore da convertire, il tipo di dato in cui convertire.

### Esempio 1

```
int numeratore = 5;
int denominatore = 3;
frazione = (double) numeratore / denominatore
```

### Esempio 2

```
pigreco = 3.14;
parte_intera_pigreco = (int) pigreco
```

## Fase di Input in Java

Le primitive per l'input da tastiera di java sono abbastanza complesse.

Per semplificare la gestione dell'input si definiscono nuove funzioni per l'input, che nascondono quelle primitive.

Ad esempio il seguente programma, che utilizza le primitive di Java, può semplificarsi molto definendo ed utilizzando un nuovo Package Java.

### Esempio

```
import java.io.*;

class RettangoloBufferedReader
{
    public static void main(String[] args) throws IOException
    {
        BufferedReader stdin = new BufferedReader
            (new InputStreamReader(System.in));

        int base, altezza, area;

        System.out.print("Base = ");
        System.out.flush();
        base = Integer.parseInt(stdin.readLine());

        System.out.print("Altezza = ");
        System.out.flush();
        altezza = Integer.parseInt(stdin.readLine());

        area = base*altezza;

        System.out.println("Area = "+area);
    }
}
```

Utilizzando il **package system**, creato all'UNICAL, il programma si semplifica nel seguente modo:

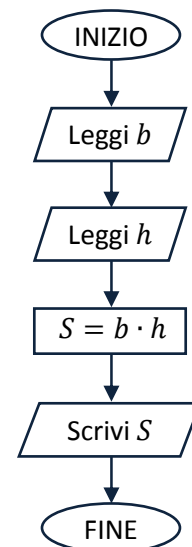
```
import system.IO;

public class AreaRettangolo
{
    public static void main(String[] args)
    {
        int base, altezza, area;

        System.out.print("Base = ");
        base=IO.in.readInt();

        System.out.print("Altezza = ");
        altezza = IO.in.readInt();

        area = base*altezza;
        System.out.print("Area = "+area);
    }
}
```



## Package system

FUNZIONE	SIGNIFICATO	ESEMPIO	RISULTATO
NomeVariabile = IO.in.readInt ( )	INPUT numero intero	base = IO.in.readInt ( )	base riceve un n° intero
NomeVariabile = IO.in.readLong ( )	INPUT numero long	base = IO.in.readLong ( )	base riceve un n° long
NomeVariabile = IO.in.readByte ( )	INPUT numero byte	base = IO.in.readByte ( )	base riceve un n° byte
NomeVariabile = IO.in.readShort ( )	INPUT numero short	base = IO.in.readShort ( )	base riceve un n° short
NomeVariabile = IO.in.readFloat ( )	INPUT numero decimale	base = IO.in.readDouble ( )	base riceve un n° decimale
NomeVariabile = IO.in.readDouble ( )	INPUT numero decimale	base = IO.in.readDouble ( )	base riceve un n° decimale
NomeVariabile = IO.in.readBoolean ( )	INPUT valore booleano	SiNo = IO.in.readBoolean ( )	SiNo riceve un booleano
NomeVariabile = IO.in.readChar ( )	INPUT carattere	Car = IO.in.readChar ( )	car riceve un carattere
NomeVariabile = IO.in.readString ( )	INPUT stringa	nome = IO.in.readString ( )	nome riceve una stringa

## Operatori di confronto

OPERATORE	SIGNIFICATO	ESEMPIO	RISULTATO
==	Uguale	a == 3	a = 3
!=	Diverso	a != 3	a ≠ 3
<	Minore	a < 3	a < 3
<=	Minore o uguale	a <= 3	a ≤ 3
>	Maggiore	a > 3	a > 3
>=	Maggiore o uguale	a >= 3	a ≥ 3
%	Resto della divisione intera	23 % 5	3

## Operatori booleani

OPERATORE	SIGNIFICATO	ESEMPIO	RISULTATO
&&	AND	(a > 3) && (a < 5)	(a > 3) e (a < 5)
	OR	(a < 3)    (a > 5)	(a < 3) oppure (a > 5)
!	NOT	a != 3	a ≠ 3

## Operatori di incremento / decremento

OPERATORE	SIGNIFICATO	ESEMPIO	RISULTATO
++	x++ equivale x = x+1	x = 5; x++	x = 6
--	x-- equivale x = x-1	x = 5; x--	x = 4
+=	x += y equivale x = x + y	x = 5; x += 2	x = 7
-=	x -= y equivale x = x - y	x = 5; x -= 2	x = 3
*=	x* = y equivale x = x * y	x = 5; x *= 2	x = 10
/=	x/ = y equivale x = x / y	x = 6; x /= 3	x = 2
%=	x% = y equivale x = x % y	x = 5; x %= 3	x = 2

## Classe Math

import java.lang.Math;

FUNZIONE	SIGNIFICATO	ESEMPIO
sin (x)	Seno di un angolo	Math.sin(30)
cos (x)	Coseno di un angolo	Math.cos(60)
tan (x)	Tangente di un angolo	Math.tan(45)
asin (x)	Arcoseno di un angolo	Math.asin(0.5)
acos (x)	Arcocoseno di un angolo	Math.acos(0.5)
atan (x)	Arcotangente di un angolo	Math.atan(0.5)
exp(x)	Esponenziale di un numero	Math.exp(2)
log(x)	Logaritmo di un numero	Math.log(5)
pow(x,n)	Potenza di un numero	Math.pow (2, 3)
round (x)	Approssimazione di un numero	Math.round (3,6)
abs (x)	Valore assoluto di un numero	Math.abs (-3)
random ()	Numero reale casuale in [0..1[	Math.random()
max (x,y)	Massimo fra x e y	Math.max(3,5)
min (x,y)	Minimo fra x e y	Math.min(3,5)

## Classe String

FUNZIONE	SIGNIFICATO	ESEMPIO	RISULTATO
s1.equals(s2)	uguaglianza tra stringhe	boolean x ; x = S1.equals(S2);	Restituisce VERO se Stringa1 = Stringa2
s1.compareTo(s2)		S1.compareTo(S2>0)	
length (s)	lunghezza di una stringa	int x; x = Stringa1.length();	Restituisce la lunghezza della Stringa1
charAt(n)	Carattere alla posizione n	String Stringa1 = "Ciao"; char x = Stringa1.charAt(2);	Restituisce il terzo carattere, cioè a (indice 2)
indexOf (s)	indica la posizione iniziale di una sottostringa	String Stringa1 = "Pippo" int x = Stringa1.indexOf("po")	La variabile x conterrà 3. Se non viene trovato nulla la variabile conterrà -1
startsWith (s)	indica se la stringa inizia con la parola o lettera indicata	String Stringa1 = "Pippo"; Boolean x = Stringa1.startsWith("Pi");	Restituisce VERO perché Stringa1 inizia con Pi
endsWith (s)	indica se la stringa termina con la parola o lettera indicata	String Stringa1 = "Pippo"; Boolean x = Stringa1.endsWith("po");	Restituisce VERO perché Stringa1 termina con po
substring (n)	restituisce la sottostringa puntata da n	String Stringa1 = "Martina"; String x = Stringa1.substring(3);	Restituisce la stringa tina
s1.concat(s2) Oppure S1 + S2	concatenazione tra stringhe	String S1="Ciao"; String S2="Mondo"; String S3=S1.concat(S2); (oppure String S3 = S1 + S2;)	Restituisce la stringa S3 = "Ciao Mondo"
s1.toUpperCase()	trasforma la stringa da minuscola a maiuscola	String S1="Ciao"; S2=s1.toUpperCase();	Restituisce la stringa S2 = "CIAO"
s1.toLowerCase()	trasforma la stringa da maiuscola a minuscola	String S1="CIAO"; S2=s1.toLowerCase();	Restituisce la stringa S2 = "ciao"
IO.in.readLine()	Riceve la stringa inserita	S1=IO.in.readLine()	

NOTA

System.out.println (x+"Risultato"+y) **NO**

System.out.println (""+x+"Risultato"+y)

**SI**



## La struttura di selezione

In un algoritmo si può verificare che le istruzioni da eseguire siano diverse a seconda dei dati elaborati.

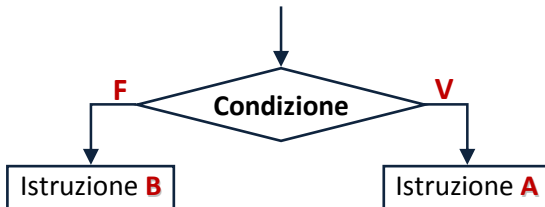
L'esecutore deve effettuare una scelta fra due alternative. Per effettuare tale scelta l'esecutore controlla il grado di verità della *condizione* posta in un test.

Se la condizione risulta *Vera* l'esecutore esegue l'istruzione A.

Se la condizione risulta *Falsa* l'esecutore segue l'istruzione B.

### Struttura di selezione

#### Flow-chart



#### Pseudolinguaggio

*Se* Condizione  
*allora*  
Istruzione A  
*altrimenti*  
Istruzione B

### Codifica in linguaggio Java

#### Selezione a due uscite

*if* (Condizione = Vera) Istruzione A;  
*else* Istruzione B;

#### Selezione a una uscita

*if* (Condizione = Vera) Istruzione A;

## Esempio – *Struttura di selezione*

Dati due numeri  $a$  e  $b$ , calcola il maggiore.

Max fra due numeri								
Flow-chart		Pseudolinguaggio		Trace table (Input: a=3 ; b=5)				
<pre> graph TD     INIZIO([INIZIO]) --&gt; LeggiA[/Leggi a/]     LeggiA --&gt; LeggiB[/Leggi b/]     LeggiB --&gt; Decision{a &gt; b}     Decision -- F --&gt; ScriviB[Scrivi b]     Decision -- V --&gt; ScriviA[Scrivi a]     ScriviB --&gt; FINE([FINE])     ScriviA --&gt; FINE                     </pre>		n°	Istruzione	n°	a > b	a	b	
		1	INIZIO	1		INIZIO		
		2	Leggi il numero $a$	2			3	
		3	Leggi il numero $b$	3				5
		4	Se $a > b$	4	Falso			
		5	<i>allora</i> Scrivi $a$	5				
		6	<i>altrimenti</i> Scrivi $b$	6				<b>5</b>
7	FINE	7				FINE		

## Linguaggio Java

```

import system.IO;
class max
{
    public static void main (String args[])
    {
        int a, b;

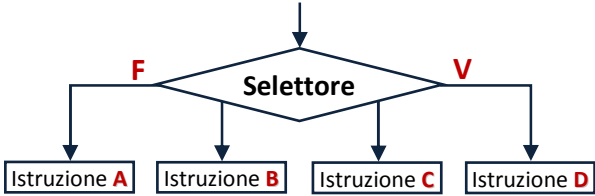
        System.out.print("Introduci il primo numero: ");
        a = IO.in.readInt();
        System.out.print("Introduci il secondo numero: ");
        b = IO.in.readInt();

        if (a > b)
            System.out.println("Il MAX e' " +a);
        else
            System.out.println("Il MAX e' " +b);
    }
}
    
```

## La struttura di selezione multipla

In un algoritmo può capitare di effettuare una scelta fra più di due alternative. Oltre all'utilizzo di istruzioni di selezione annidati, è possibile utilizzare la struttura di selezione multipla.

Il selettore multiplo controlla il valore ricevuto ed esegue l'istruzione ad esso relativo.

Struttura di selezione multipla	
Flow-chart	Pseudolinguaggio
	<p><i>Se</i> Selettore</p> <p>Valore 1: Istruzione A</p> <p>Valore 2: Istruzione B</p> <p>Valore 3: Istruzione C</p> <p>...</p> <p><i>altrimenti</i> Istruzione X</p> <p><i>FineSelettore</i></p>

Codifica in linguaggio Java
<pre><i>switch</i> ( &lt;selettore&gt; ) {     <i>case</i> &lt;valore1&gt;: &lt;istruzioneA&gt; <i>break</i>;     <i>case</i> &lt;valore2&gt;: &lt;istruzioneB&gt; <i>break</i>;     <i>case</i> &lt;valore3&gt;: &lt;istruzioneC&gt; <i>break</i>;     ...     <i>default</i>: &lt;istruzioniN&gt; <i>break</i>; }</pre>

### Nota

I valori che possono essere sottoposti al controllo del selettore switch sono costanti intere, char o enumerative.

### Esempio – *Struttura di selezione multipla*

```
import system.IO;
class giornoSettimanale
{
    public static void main (String args[])
    {
        int numeroGiorno;
        String giorno;

        System.out.print("Quale giorno (in numero) della settimana e' oggi ? ");
        numeroGiorno = IO.in.readInt();

        switch ( numeroGiorno )
        {
            case 1: giorno = "Lunedì"; break;
            case 2: giorno = "Martedì"; break;
            case 3: giorno = "Mercoledì"; break;
            case 4: giorno = "Giovedì"; break;
            case 5: giorno = "Venerdì"; break;
            case 6: giorno = "Sabato"; break;
            case 7: giorno = "Domenica"; break;
            default: giorno = null; // in tutti gli altri casi
        }
        System.out.println("Oggi e' "+giorno);
    }
}
```

## La struttura di iterazione

In un algoritmo può capitare che alcune istruzioni debbano essere ripetute, in modo identico, più volte.

La ripetizione di un insieme di istruzioni prende il nome di *iterazione* o *ciclo* (loop).

Il gruppo di istruzioni ripetute prende il nome di *corpo del ciclo*.

L'istruzione di iterazione può essere definita o indefinita.

L'iterazione è detta *definita* quando è noto a priori il numero di ripetizioni.

L'iterazione è detta *indefinita* quando il ciclo viene ripetuto un numero di volte sconosciuto a priori e termina quando si verifica una determinata condizione.

### Esempi

- 🔧 Ripeti 10 volte la preghiera "Atto di dolore" (Iterazione definita)
- 🔧 Bevi un bicchiere di vino finché non cadi a terra ubriaco (Iterazione indefinita)

L'iterazione indefinita è detta *precondizionale* (o iterazione per vero) se il controllo per l'arresto dell'iterazione è posto prima del gruppo di istruzioni da ripetere.

L'iterazione indefinita è detta *postcondizionale* (o iterazione per falso) se il controllo per l'arresto dell'iterazione è posto dopo del gruppo di istruzioni da ripetere.

Strutture di Iterazione		
<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
Mentre Condizione = Vera esegui Istruzioni	Ripeti Istruzioni finché Condizione = Falsa	Ripeti Istruzioni N volte

Codifica in linguaggio Java		
<i>Iterazione indefinita precondizionale</i>	<i>Iterazione indefinita postcondizionale</i>	<i>Iterazione definita enumerativa</i>
<b>while</b> (Condizione = Vera) { Istruzioni }	<b>do</b> { Istruzioni } <b>while</b> (Condizione = Falsa)	<b>for</b> (contatore=1; contatore < N°Iterazioni; contatore++) { Istruzioni }

## Esempio 1 - Iterazione indefinita precondizionale

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss  $S_n = \frac{n(n+1)}{2}$ .

Somma dei primi N numeri naturali						
Flow-chart	Pseudolinguaggio	Trace table (Input N=5)				
	n°	Istruzione	n°	Cont < N	Contatore	Somma
	1	INIZIO	3		1	
	2	Leggi il numero <i>N</i>	4			0
	3	Assegna al <i>Contatore</i> il valore <i>1</i>	5	Vero		
	4	Assegna alla <i>Somma</i> il valore <i>0</i>	7			0+1=1
	5	<i>Mentre</i> il <i>Contatore</i> <= <i>N</i> <i>fai</i>	8		2	
	6	<i>Inizio</i>	5	Vero		
	7	Aggiungi alla <i>somma</i> il <i>contatore</i>	7			1+2=3
	8	Incrementa di <i>1</i> il <i>contatore</i>	8		3	
	9	<i>Fine</i>	5	Vero		
	10	Scrivi <i>Somma</i>	7			3+3=6
11	FINE	8		4		
		5	Vero			
		7			6+4=10	
		8		5		
		5	Vero			
		7			10+5=15	
		8		6		
		5	Falso			
		10			<b>15</b>	
		11	FINE			

## Linguaggio Java

```
import system.IO;
class sommaNumeri_While
{
    public static void main (String [] args)
    {
        int contatore, somma, n;

        IO.out.println("                SOMMA DEI PRIMI N NUMERI NATURALI");

        IO.out.print("Introduci un numero naturale: ");
        n = IO.in.readInt();

        somma = 0;
        contatore = 1;

        while (contatore <= n)
        {
            somma = somma + contatore;
            contatore++;
        }

        IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
    }
}
```

## Esempio 2 - Iterazione indefinita postcondizionale

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss  $S_n = \frac{n(n+1)}{2}$ .

Somma dei primi N numeri naturali						
Flow-chart	Pseudolinguaggio	Trace table (Input N=5)				
<pre> graph TD     INIZIO([INIZIO]) --&gt; LeggiN[/Leggi N/]     LeggiN --&gt; Contatore0[Contatore = 0]     Contatore0 --&gt; Somma0[Somma = 0]     Somma0 --&gt; SommaAdd[Somma = Somma + Contatore]     SommaAdd --&gt; ContatoreInc[Contatore = Contatore + 1]     ContatoreInc --&gt; ContatoreLeN{Contatore &lt;= N}     ContatoreLeN -- V --&gt; SommaAdd     ContatoreLeN -- F --&gt; ScriviSomma[/Scrivi Somma/]     ScriviSomma --&gt; FINE([FINE])                     </pre>	n°	Istruzione	n°	Cont < N	Contatore	Somma
	1	INIZIO	1	INIZIO		
	2	Leggi il numero <i>N</i>	2			
	3	Assegna al <i>Contatore</i> il valore <i>0</i>	3		0	
	4	Assegna alla <i>Somma</i> il valore <i>0</i>	4			0
	5	<i>Ripeti</i>	7			0+0=0
	6	<i>Inizio</i>	8		1	
	7	Aggiungi alla <i>somma</i> il <i>contatore</i>	10	Vero		
	8	Incrementa di <i>1</i> il <i>contatore</i>	7			1+2=3
	9	<i>Fine</i>	8		2	
	10	<i>Finchè Contatore &lt;= N</i>	10	Vero		
	11	Scrivi <i>Somma</i>	7			3+3=6
12	FINE	8		3		
		10	Vero			
		7			6+4=10	
		8		4		
		10	Vero			
		7			10+5=15	
		8		5		
		10	Falso			
		11			<b>15</b>	
		12		FINE		

## Linguaggio Java

```

import system.IO;
class sommaNumeri_Do
{
    public static void main (String [] args)
    {
        int contatore, somma, n;

        IO.out.println("                SOMMA DEI PRIMI N NUMERI NATURALI");

        IO.out.print("Introduci un numero naturale: ");
        n = IO.in.readInt();

        contatore=0;
        somma = 0;

        do
        {
            somma = somma + contatore;
            contatore++;
        }
        while (contatore <= n);

        IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
    }
}
                    
```

### Esempio 3 - Iterazione definita enumerativa

Calcola la **Somma** dei primi **N** numeri naturali, senza utilizzare la formula di Gauss  $S_n = \frac{n(n+1)}{2}$ .

Somma dei primi N numeri naturali					
Flow-chart	Pseudolinguaggio	Trace table (N=5)			
<pre> graph TD     INIZIO([INIZIO]) --&gt; LeggiN[/Leggi N/]     LeggiN --&gt; Somma0[Somma = 0]     Somma0 --&gt; Contatore[Contatore = 1, N]     Contatore --&gt; LoopStart(( ))     LoopStart --&gt; SommaAdd[Somma = Somma + Contatore]     SommaAdd --&gt; LoopStart     LoopStart --&gt; Decision{ }     Decision --&gt; SommaAdd     Decision --&gt; ScriviSomma[/Scrivi Somma/]     ScriviSomma --&gt; FINE([FINE])                     </pre>	n°	Istruzione	n°	Contatore	Somma
	1	INIZIO	3		0
	2	Leggi il numero <i>N</i>	4	1	
	3	Assegna alla <i>Somma</i> il valore <i>0</i>	6		0+1=1
	4	Ripeti con <i>Contatore da 1 a N</i>	4	2	
	5	<i>Inizio</i>	6		1+2=3
	6	Aggiungi alla <i>somma</i> il <i>contatore</i>	4	3	
	7	<i>Fine</i>	6		3+3=6
	8		4	4	
	9	Scrivi <i>Somma</i>	6		6+4=10
10	FINE	4	5		
		6		10+5=15	
		9		<b>15</b>	
		10		FINE	

### Linguaggio Java

```

import system.IO;
class sommaNumeri_For
{
    public static void main (String [] args)
    {
        int contatore, somma, n;

        IO.out.println("                SOMMA DEI PRIMI N NUMERI NATURALI");

        IO.out.print("Introduci un numero naturale: ");
        n = IO.in.readInt();

        somma = 0;

        for (contatore=1; contatore <= n; contatore++)
        {
            somma = somma + contatore;
        }

        IO.out.println("La somma dei primi "+n+" numeri naturali e' "+somma);
    }
}
    
```