

# AccademyJava

## Campari Mirko - Bruno Vincenzo Java



**Benvenuti**

---

**Benvenuti a tutti**

**Inizieremo con:  
Presentazione e test d'ingresso**

**LA PAUSA FINISCE A E 22**

---

# Cosa vedremo

---

- TEST D'INGRESSO
  - INTRODUZIONE A JAVA
  - INTRODUZIONE DI HTML E CSS
  - INTRODUZIONE AL CONCETTO BASE DI DATI
  - INTRODUZIONE AI DATABASE
  - CHE COS'È UN DATABASE
  - CHE COS'È UN DBMS
  - INTRODUZIONE VERSIONAMENTO
  - INTRODUZIONE CONCETTO DI ASTRAZIONE
  - TEST DI FINE INTRODUZIONE
-

# Cosa ci servirà

**Un IDE. ( VSC )**

**Java installato nella nostra macchina**

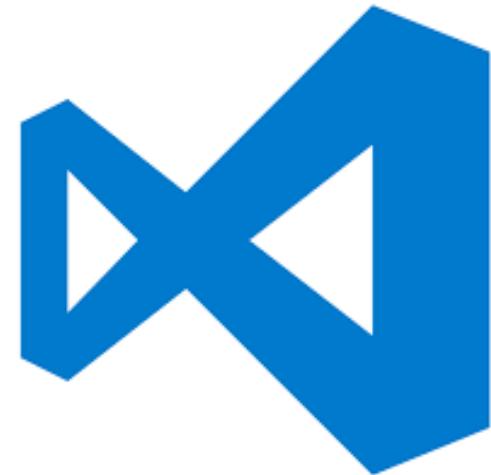
**Alcune estensioni per HTML e JAVA che  
vedremo più avanti**

**La documentazione che verrà fornita**

## Che cosa si intende per IDE?

***Integrated Development Environment (IDE)***

Un IDE, o ambiente di sviluppo integrato, è un software progettato per la realizzazione di applicazioni che aggrega strumenti di sviluppo comuni in un'unica interfaccia utente grafica.



Noi useremo **VISUAL STUDIO CODE**

<https://code.visualstudio.com/download>

In realtà se dovessimo essere pignoli **VSC** non è  
propriamente un IDE ma una light version

**Per chi non ha eseguito l'istallazione di JAVA  
andremo ora a seguire questo tutorial ufficiale**

**[https://www.java.com/it/download/help/windows\\_manual\\_download.html](https://www.java.com/it/download/help/windows_manual_download.html)**

**e l'estensione**

**Extension Pack for Java**

**Un IDE è: Un ambiente di sviluppo integrato, un software che quando siamo nella fase di programmazione ci aiuta nello sviluppo e nel debugging/testing del nostro codice**

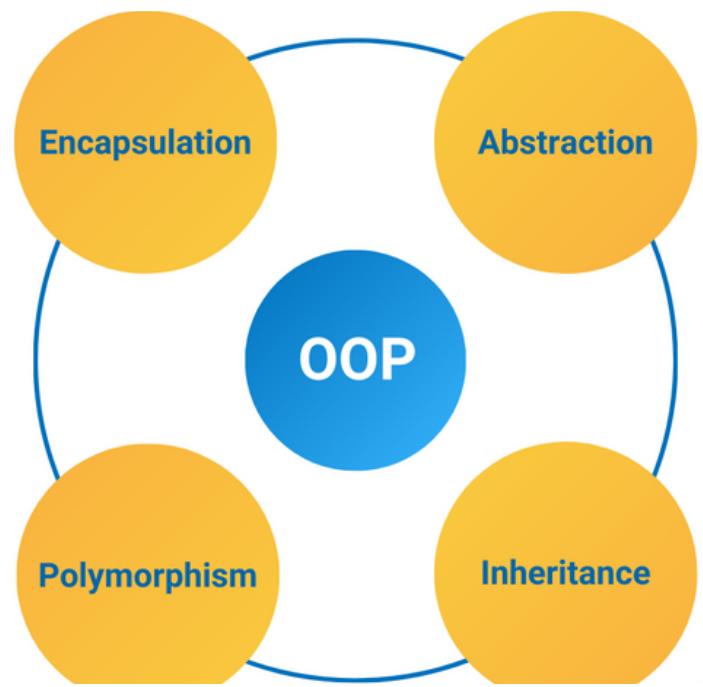
Ne esistono molteplici che differiscono l'uno dall'altro per specificità e caratteristiche rispetto ai linguaggi o alle tecnologie, noi per ora andremo ad installare VSC per tutti, in seguito ciò verrà lasciato alla vostra libera scelta

**Cosa faremo :**

- Scaricheremo e Installeremo l'IDE ( VSC = Visual Studio Code )
- Installeremo le Librerie fondamentali dei Linguaggi JAVA

# OOP

In informatica, la programmazione orientata agli oggetti (in inglese object-oriented programming, in acronimo OOP), a volte chiamata semplicemente programmazione ad oggetti, è un paradigma di programmazione che permette di definire oggetti software in grado di interagire gli uni con gli altri attraverso lo scambio di messaggi.



Particolarmente adatta nei contesti in cui si possono definire delle relazioni di interdipendenza tra i concetti da modellare (contenimento, uso, specializzazione), un ambito che più di altri riesce a sfruttare i vantaggi della programmazione ad oggetti è quello delle interfacce grafiche.

## Alcuni dei vantaggi della programmazione OOP:

- essa fornisce un supporto naturale alla modellazione software degli oggetti del mondo reale o del modello astratto da riprodurre;
- permette una più facile gestione e manutenzione di progetti di grandi dimensioni;
- l'organizzazione del codice sotto forma di classi favorisce la modularità e il riuso di codice.

**La programmazione ad oggetti prevede di raggruppare in alcune parti circoscritte del codice sorgente, chiamate classi, la dichiarazione delle strutture dati e delle procedure che operano su di esse. Le classi, quindi, costituiscono dei modelli astratti, che a tempo di esecuzione vengono invocate per istanziare o creare oggetti software relativi alla classe invocata. Questi ultimi sono dotati di attributi (dati) e metodi (procedure) secondo quanto definito/dichiarato dalle rispettive classi.**

**La parte del programma che fa uso di un oggetto si chiama client.**

---

**Un linguaggio di programmazione è definito ad oggetti quando permette di implementare tre meccanismi usando la sintassi nativa del linguaggio**

***Incapsulamento;***

***Ereditarietà;***

***Polimorfismo.***

---

- L'**incapsulamento** consiste nella separazione della cosiddetta **interfaccia** di una classe dalla corrispondente **implementazione**, in modo che i client di un oggetto di quella classe possano utilizzare la prima, ma non la seconda.
- L'**ereditarietà** permette essenzialmente di definire delle classi a partire da altre già definite.
- Il **polimorfismo** permette di scrivere un client che può servirsi di oggetti di classi diverse, ma dotati di una stessa interfaccia comune; nel tempo di esecuzione tale client potrà attivare comportamenti diversi senza conoscere a priori il tipo specifico dell'oggetto che gli viene dato in ingresso.

**ORA VEDEREMO UNO DEGLI ELEMENTI PRINCIPALI DELL'OOP:**

## **LA CLASSE**

**Le classi definiscono dei tipi di dato e permettono la creazione degli oggetti secondo le caratteristiche definite nella classe stessa.**

**Grazie alle relazioni di ereditarietà, è possibile creare nuove classi a partire da quelle esistenti, estendendole con caratteristiche aggiuntive.**

## La classe è composta da:

---

- attributi, ossia variabili e/o costanti che definiscono le caratteristiche o proprietà degli oggetti instanziabili invocando la classe; i valori inizializzati degli attributi sono ottenuti attraverso il cosiddetto costruttore;
- metodi, ossia procedure che operano sugli attributi.

## La classe è composta da:

Un paragone (impreciso) con la matematica è il seguente: si può pensare che una classe definisca un insieme in modo intensivo, ovvero indicandone le caratteristiche invece che elencandone gli elementi, e che gli oggetti siano gli elementi di quell'insieme. Tuttavia, in matematica, il numero degli elementi è una caratteristica intrinseca dell'insieme stesso, e risulta definito nel momento in cui si definisce l'insieme, mentre in programmazione è possibile istanziare una classe un numero di volte arbitrario (teoricamente da zero ad infinito, in pratica da zero fino a esaurimento della memoria di lavoro del calcolatore) e che dipende dall'esecuzione del programma. Per questo motivo, è preferibile considerare la classe come un modello astratto istanziabile.

In altri termini, una classe è paragonabile al progetto di un'infrastruttura che può poi essere messa in opera/esercizio ovvero realizzata o meno con l'istanziazione dei suoi oggetti tutti con le medesime caratteristiche, ovvero gli attributi (con valori diversi), su cui opereranno i metodi o funzioni..

# Ma cos'è un oggetto

---

**Un oggetto è una istanza di una classe.**

**Esso è dotato di tutti gli attributi e i metodi della classe, ed agisce come un fornitore di "messaggi" (i metodi) che il codice eseguibile del programma (procedure o altri oggetti) può attivare su richiesta.**

**Inviare un messaggio ad un oggetto significa, in gergo, invocare un metodo su quell'oggetto. Il metodo riceve come parametro l'oggetto su cui è stato invocato, che può essere referenziato tramite una parola-chiave o una sintassi apposita, anche se è passato come parametro implicito; per esempio, in C++, in Java, e in C# si usa la parola-chiave `this`, mentre in Smalltalk, in Objective-C, Python e in Ruby si usa la parola-chiave `self`.**

## Ma cos'è un oggetto

---

Dal punto di vista del calcolatore, ogni oggetto è identificato da una certa zona di memoria, nella quale sono memorizzati gli attributi, e il valore di questi ultimi determina lo stato interno dell'oggetto. Istanziare un oggetto vuol dire allocare memoria ed eventualmente inizializzarla secondo le specifiche definite dalla classe.

Molti linguaggi forniscono un supporto per l'inizializzazione automatica di un oggetto, con uno o più metodi speciali, detti costruttori. Analogamente, la fine della vita di un oggetto può essere gestita con un metodo detto distruttore.

## Ma cos'è un oggetto

---

Il codice eseguibile del programma accede a tale zona di memoria sempre e solo secondo le modalità date dalla classe.

Secondo il principio noto come **information hiding**, l'accesso ai campi di un oggetto deve essere permesso solo tramite metodi invocati su quello stesso oggetto. Il vantaggio principale è che il controllo completo sullo stato interno viene assegnato ad una zona ristretta del codice eseguibile del programma (la classe), perché il codice esterno non è autorizzato a modificarlo. In tal caso, risulta possibile imporre dei vincoli sui possibili valori che la tupla degli attributi può o non può assumere, e anche sulle possibili transizioni tra questi stati. Un oggetto quindi può essere visto come una macchina a stati finiti.

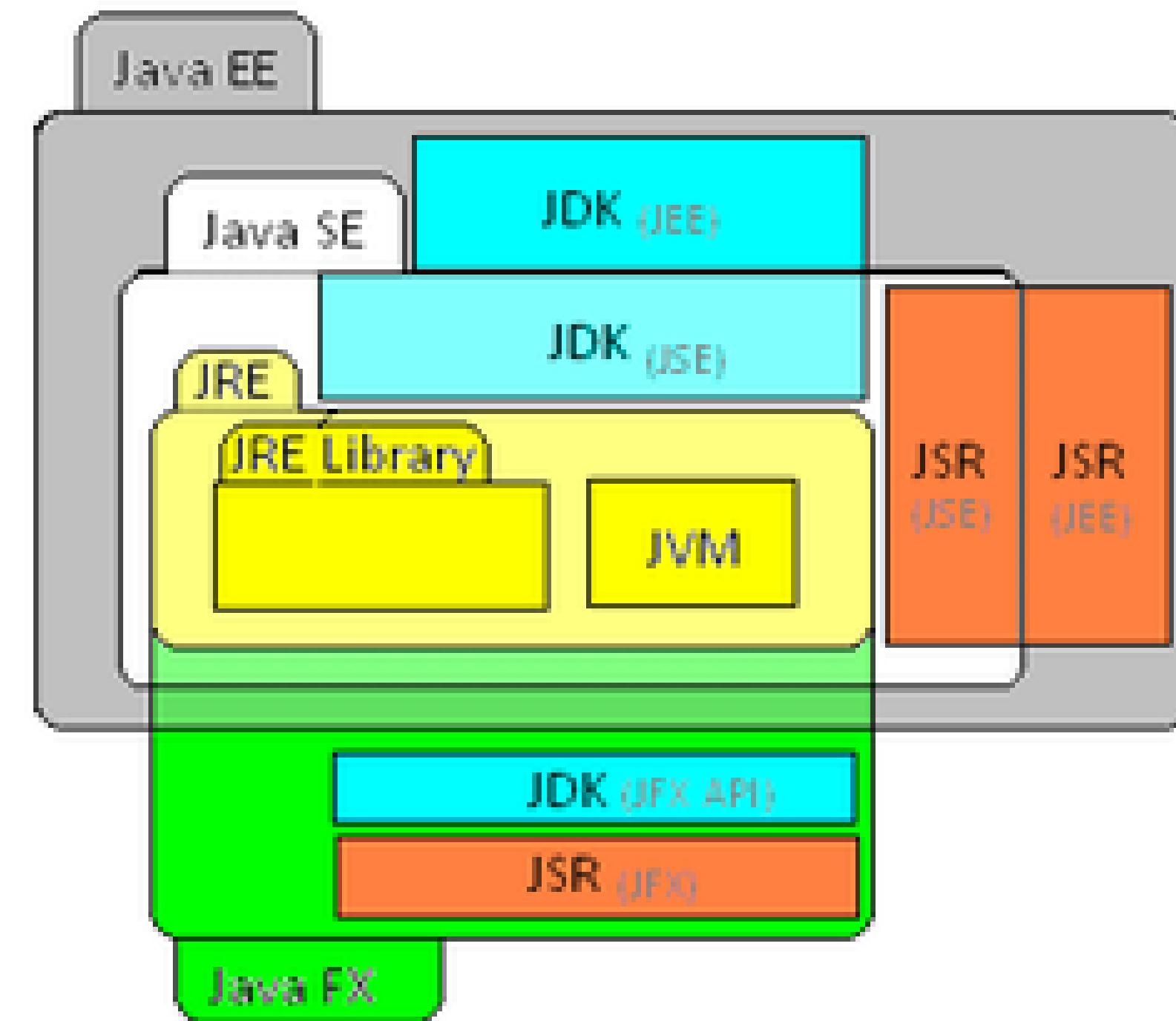


**Java™**

**In informatica la piattaforma Java è una piattaforma software (API - application programming interface), sviluppata su specifiche e implementazioni di Sun Microsystems (acquisita nel gennaio 2010 dalla Oracle Corporation) ovvero l'ambiente di esecuzione necessario per l'esecuzione di programmi scritti in linguaggio java.**

**Tale piattaforma ha come caratteristica il fatto di rendere possibile scrittura ed esecuzione di applicazioni indipendenti dall'hardware di esecuzione, che risulta così virtualizzato dalla piattaforma stessa, rendendo così il linguaggio java e i relativi programmi portabili su piattaforme hardware diverse**

# Architettura di JAVA



# Cos'è Java?

---

**Java è un popolare linguaggio di programmazione, creato nel 1995.  
È di proprietà di Oracle e più di 3 miliardi di dispositivi eseguono  
Java.**

**È usato per:**

- **Applicazioni mobili (specialmente app Android)**
  - **Applicazioni desktop**
  - **Applicazioni web**
  - **Server Web e server applicativi**
  - **Giochi**
  - **Connessione alla banca dati**
  - **E molto, molto altro!**
-

# Perché usare Java?

---

- **Java funziona su diverse piattaforme (Windows, Mac, Linux, Raspberry Pi, ecc.)**
- **È uno dei linguaggi di programmazione più diffusi al mondo**
- **Ha una grande richiesta nell'attuale mercato del lavoro**
- **È facile da imparare e semplice da usare**
- **È open-source e gratuito**
- **È sicuro, veloce e potente**
- **Ha un enorme supporto da parte della comunità (decine di milioni di sviluppatori)**
- **Java è un linguaggio orientato agli oggetti che dà una struttura chiara ai programmi e consente il riutilizzo del codice, abbassando i costi di sviluppo**
- **Poiché Java è vicino a C++ e C#, rende facile per i programmatore passare a Java o viceversa**

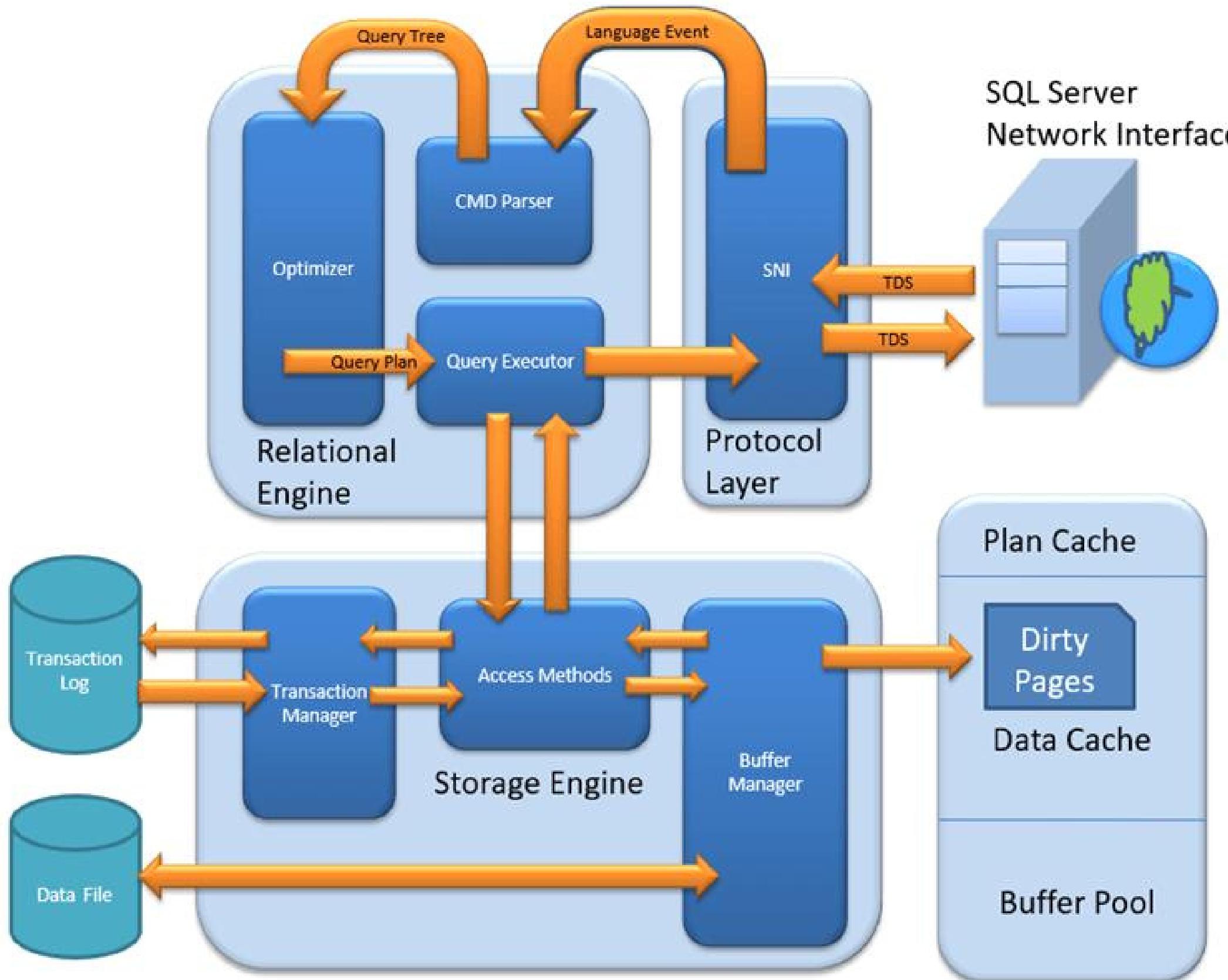


**Structured Query Language (SQL) è un linguaggio di interrogazione (query) utilizzato per creare, modificare e gestire i dati in un database relazionale.**

**Si tratta nello specifico di un linguaggio specifico di dominio (DSL) usato per comunicare con i sistemi di gestione di database relazionali (RDBMS).**

---

# **SQL (Structured Query Language)**



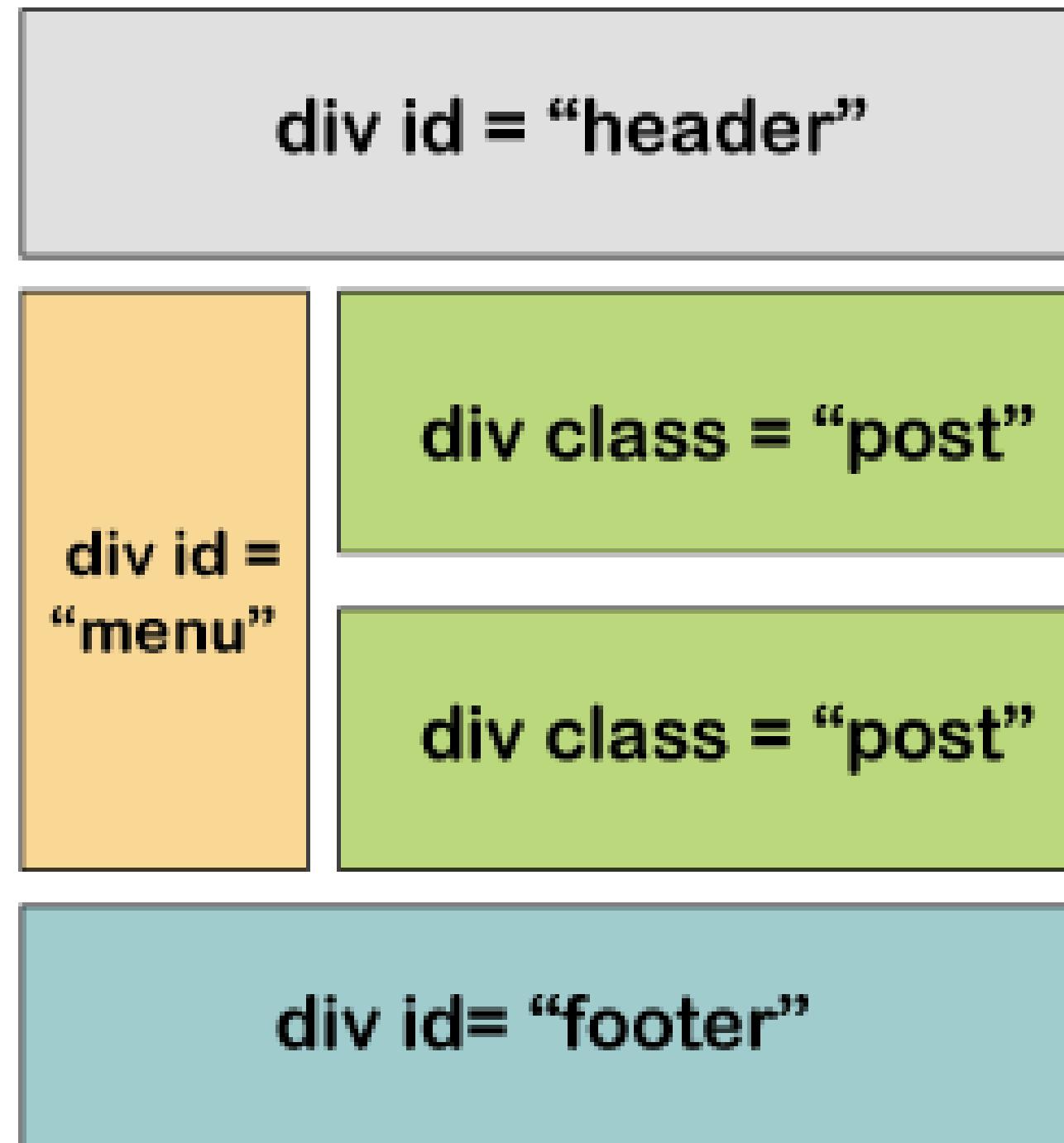
**HTML**



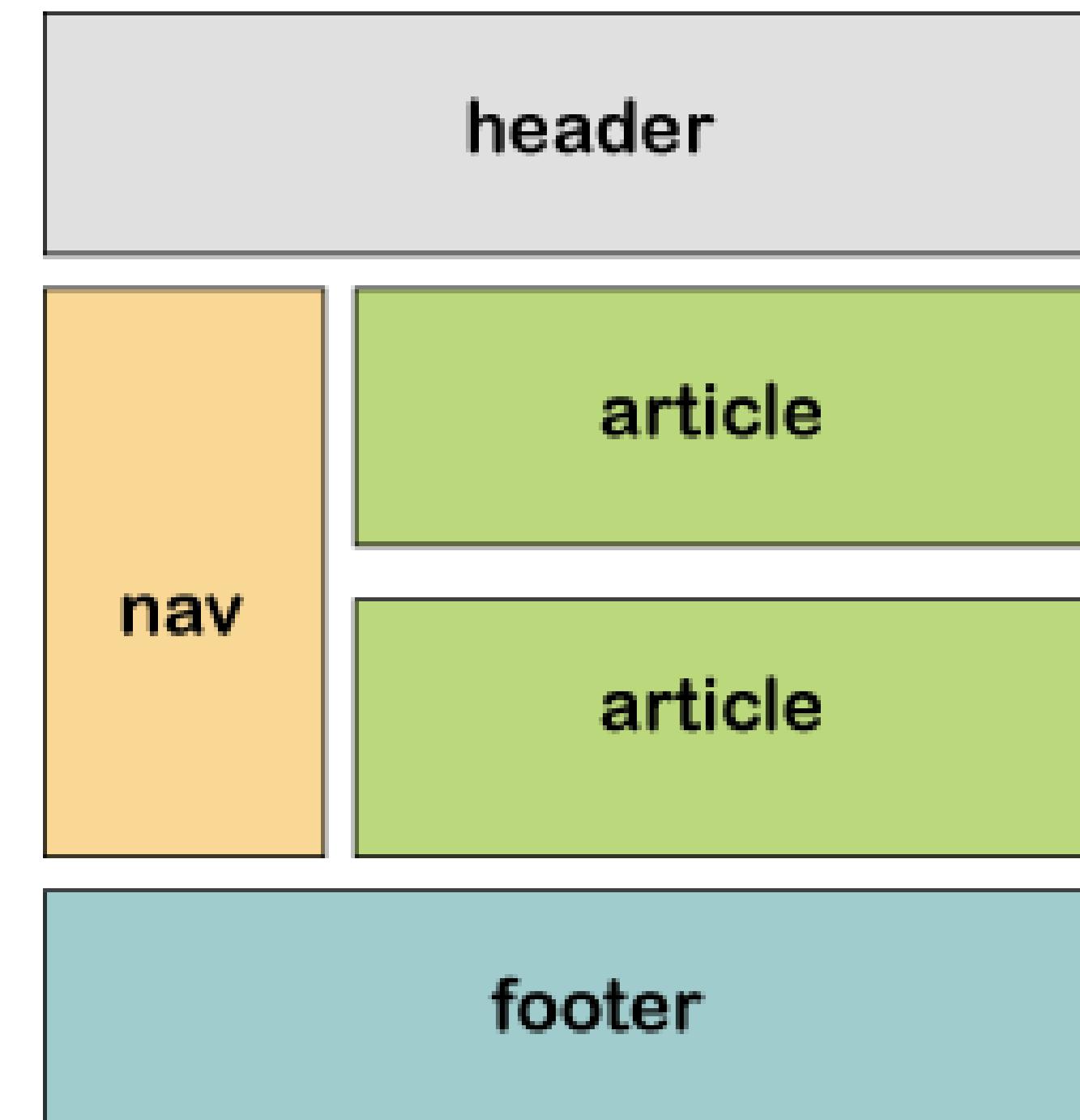
**In informatica l'HyperText Markup Language (traduzione letterale: linguaggio a marcatori per ipertesti), comunemente noto con l'acronimo HTML, è un linguaggio di markup.**

**Nato per la formattazione e impaginazione di documenti ipertestuali disponibili nel web 1.0, oggi è utilizzato principalmente per il disaccoppiamento della struttura logica di una pagina web (definita appunto dal markup) e la sua rappresentazione, gestita tramite gli stili CSS per adattarsi alle nuove esigenze di comunicazione e pubblicazione all'interno di Internet.**

## Html4 Structure



## Html5 Structure

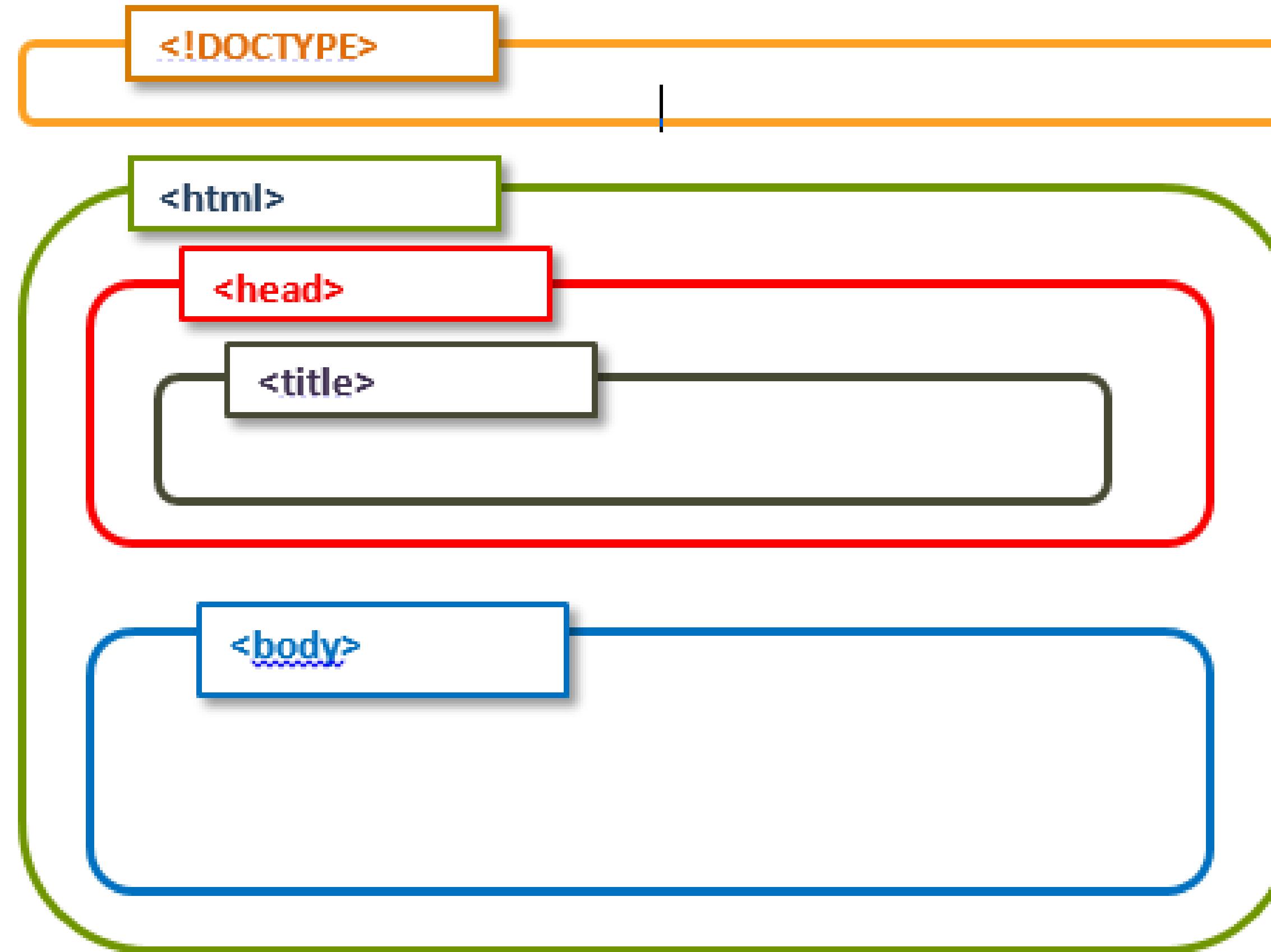


VS

**L'HTML è un linguaggio di pubblico dominio, la cui sintassi è stabilita dal World Wide Web Consortium (W3C).**

**È derivato dall'SGML, un metalinguaggio finalizzato alla definizione di linguaggi utilizzabili per la stesura di documenti destinati alla trasmissione in formato elettronico. La versione attuale, la quinta, è stata rilasciata dal W3C nell'ottobre 2014.**

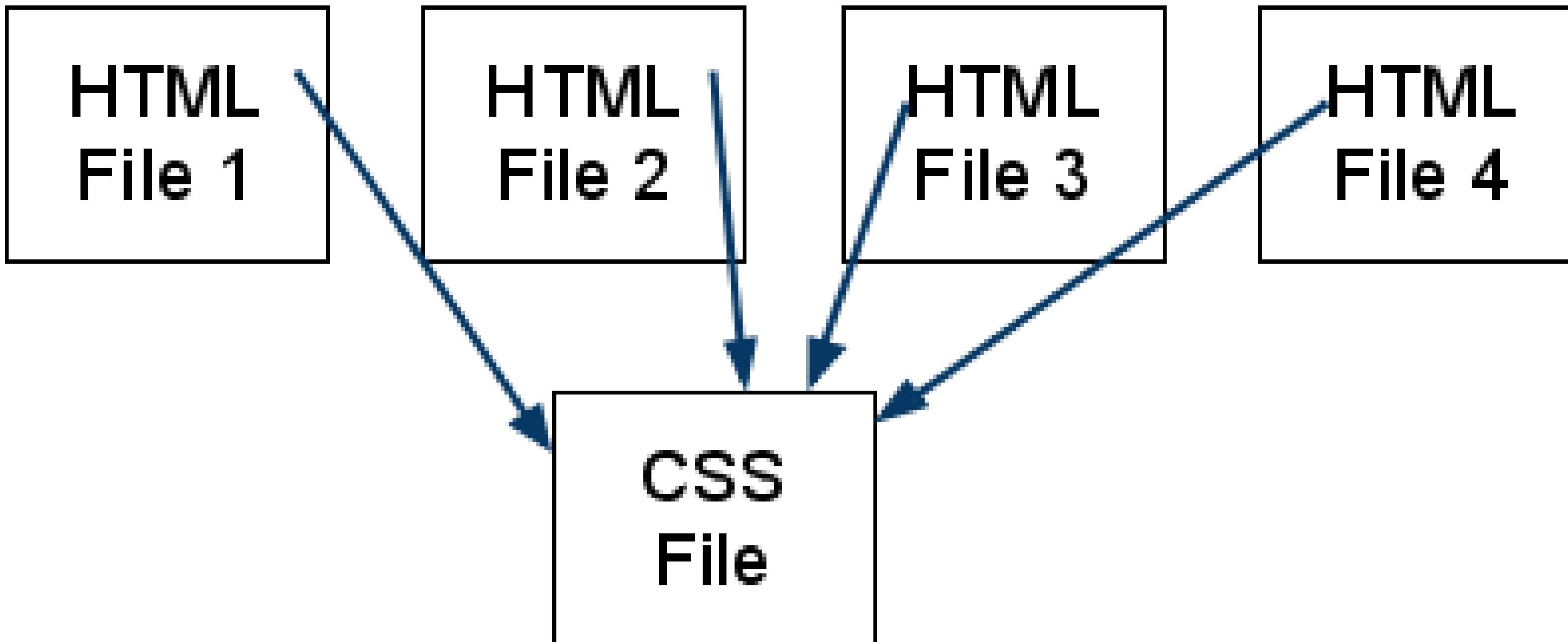
**Il motivo principale che ha spinto il W3C e i suoi membri a sviluppare HTML5 è stata la necessità di fornire direttamente le funzionalità che in precedenza erano fruibili tramite estensioni proprietarie all'esterno dei browser, come Adobe Flash e simili. Un secondo obiettivo che gli sviluppatori si erano prefissati era quello di garantire una maggiore compatibilità tra i diversi browser, indipendentemente dalla piattaforma software utilizzata, e principalmente mirata all'espansione dei dispositivi mobili.**



**Il CSS (sigla di Cascading Style Sheets, in italiano fogli di stile a cascata), in informatica, è un linguaggio usato per definire la formattazione di documenti HTML, XHTML e XML, ad esempio i siti web e relative pagine web. Le regole per comporre il CSS sono contenute in un insieme di direttive (Recommendations) emanate a partire dal 1996 dal W3C.**



**L'introduzione del CSS si è resa necessaria per separare i contenuti delle pagine HTML dalla loro formattazione o layout e permettere una programmazione più chiara e facile da utilizzare, sia per gli autori delle pagine stesse sia per gli utenti, garantendo contemporaneamente anche il riutilizzo di codice ed una sua più facile manutenzione.**



## A cosa serve SQL

---

**Come detto, SQL (Structured Query Language) è un linguaggio standardizzato per database basati sul modello relazionale (RDBMS), progettato per le seguenti operazioni:**

- **creare e modificare schemi di database (DDL = Data Definition Language);**
- **inserire, modificare e gestire dati ( Data Manipulation Language);**
- **interrogare i dati memorizzati (DQL = Data Query Language);**
- **creare e gestire strumenti di controllo e accesso ai dati (DCL = Data Control Language).**

**A dispetto del nome, non si tratta perciò di un semplice linguaggio di interrogazione: alcuni suoi sottoinsiemi, infatti, permettono di creare, gestire e amministrare database.**

---



## Esempio CIAO MONDO

**Andiamo a creare un file Java chiamato Main.java, e andiamo ad usarlo tramite il seguente codice per stampare "Ciao Mondo " a schermo:**

### Main.java

```
1. public class Main {  
2.     public static void main(String[] args) {  
3.         System.out.println("Ciao Mondo "); } }
```

## Esempio Spiegato

**Ogni riga di codice che viene eseguita in Java deve trovarsi all'interno di un file class.**

**Nel nostro esempio, abbiamo chiamato la classe Main . Una classe dovrebbe sempre iniziare con una prima lettera maiuscola.**

**Nota: Java fa distinzione tra maiuscole e minuscole: "MyClass" e "myclass" hanno significati diversi. Il nome del file java deve corrispondere al nome della classe. Quando si salva il file, salvarlo utilizzando il nome della classe e aggiungere ".java" alla fine del nome del file. Per eseguire l'esempio sopra sul tuo computer, assicurati che Java sia installato correttamente: vai al capitolo Iniziare per sapere come installare Java. L'output dovrebbe essere:**

Hello World

# Il metodo MAIN

**Il main() metodo è sempre richiesto e lo vedrai in ogni programma Java che andrai ad utilizzare o scrivere:**

## 1. **public static void main(String[] args)**

**Qualsiasi codice all'interno del metodo main() verrà eseguito.**  
**Non preoccuparti delle parole chiave prima e dopo main.**  
**Li conoscerai poco a poco durante il corso.**  
**Per ora ricordiamoci solo che ogni programma Java ha un classname che deve corrispondere al nome del file, e che ogni programma deve contenere il metodo main().**

# System.out.println()

**All'interno del main() metodo, possiamo utilizzare il println() metodo per stampare una riga di testo sullo schermo:**

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

**Nota: le parentesi graffe {} segnano l'inizio e la fine di un blocco di codice.**

**System è una classe Java nativa che contiene elementi utili, come out, che è l'abbreviazione di "output". Il metodo println(), abbreviazione di "print line", viene utilizzato per stampare un valore sullo schermo (o su un file).**

**Non preoccuparti troppo di System, out e println(). Sappi solo che ti servono per stampare cose sullo schermo.**

**Si noti inoltre che ogni istruzione di codice deve terminare con un punto e virgola ( ;).**

**Hai imparato dal capitolo precedente che puoi utilizzare il `println()` metodo per generare valori o stampare testo in Java:**

**1. `System.out.println("Hello World!");`**

**Quando lavori con il testo, anche detto string, deve essere racchiuso tra virgolette doppie """. Se dimentichi le doppie virgolette, si verifica un errore:**

**1. `System.out.println("TUTTO OK!");`**  
**2. `System.out.println(sbagliato!);`**

**C'è anche un metodo `print()`, che è simile a `println()`.**

**L'unica differenza è che non inserisce una nuova riga alla fine dell'output:**

**1. `System.out.print("Hello World! ");`**  
**2. `System.out.print("I will print on the same line.")`**

**È inoltre possibile utilizzare il `println()` metodo per stampare i numeri.  
Tuttavia, a differenza del testo, non mettiamo i numeri tra virgolette:**

- 1. `System.out.println(3);`**
- 2. `System.out.println(358);`**
- 3. `System.out.println(50000);`**

**Puoi anche eseguire calcoli matematici all'interno del metodo `println()`:**

- 1. `System.out.println(3 + 3);`**
- 2. `System.out.println(3 * 2);`**

# COMMENTI in JAVA

---

**Il commento, nell'ambito dei linguaggi di programmazione, è una parte del codice sorgente che ha il solo scopo di descriverne le caratteristiche funzionali, ovvero di spiegare il funzionamento delle successive linee di codice, e che non fa parte dell'algoritmo risolutivo codificato in linguaggio di programmazione.**

**Durante il processo di compilazione queste istruzioni sono ignorate e di conseguenza non pesano computazionalmente sulla grandezza dell'eseguibile prodotto. Queste particolari notazioni hanno molta importanza, soprattutto se il programma è sviluppato da persone diverse e in tempi diversi, aumentandone la leggibilità/intelligenza al lettore e favorendone così la manutenibilità.**

**I commenti vengono anche usati per inibire temporaneamente l'esecuzione di alcune porzioni di codice, con lo scopo di non eseguire alcune parti del codice senza tuttavia rimuoverle dal codice sorgente stesso. Quest'ultima pratica può essere utilizzata in fase di debugging del programma.**

---

**Il commentare in JAVA e in tutti i linguaggi è fondamentale possiamo usarli per spiegare il codice, per renderlo più leggibile oltreché il semplice essere utilizzarlo per impedire l'esecuzione durante il test del codice alternativo o semplicemente non utilizzarlo.**

**I commenti a riga singola iniziano con due barre ( //).**  
**qualsiasi testo compreso tra // e la fine della riga viene ignorato da Java e non verrà eseguito. Questo esempio utilizza un commento a riga singola prima di una riga di codice:**

- 1.// questo è un commento**
- 2.System.out.println("Hello World");**
- 3.System.out.println("Hello World"); // questo è un commento**

**Commenti multilinea Java SONO commenti su più righe CHE iniziano con /\*e terminano con \*/.**

**Qualsiasi testo compreso tra /\*e \*/ verrà ignorato da Java.**

**Questo esempio utilizza un commento su più righe (un blocco di commento) per spiegare il codice:**

- 1./\* QUESTO CODICE**
- 2.è UN ESEMPIO DI COMMENTO MULTILINEA \*/**
- 3.System.out.println("Ciao MONDO");**

# Esercizio

---

# LE VARIABILI

---

**La variabile è un identificatore V associato a un insieme prefissato di possibili valori che definiscono il tipo della variabile.**

**L'insieme dei possibili valori definisce il range dei valori che V può assumere durante l'esecuzione di un programma.  
Definendo il tipo e la rappresentazione della variabile oltre al range di valori, vengono definite anche le operazioni possibili con la variabile stessa. Durante l'esecuzione di un programma ciascuna variabile ha un valore corrente.**

---

# LE VARIABILI

---

**Le variabili consentono l'archiviazione dei dati.**

**In Java, ci sono diversi tipi di variabili, le principali sono:**

- **char - contiene singoli caratteri, come 'a' o 'B'.**
  - **String - è una serie di char quindi contiene del testo, come "Ciao".**
  - **int - contiene numeri interi, come 123 o -123.**
  - **float - contiene numeri a virgola mobile ovvero con i decimali, come 19,99 o -19,99.**
  - **boolean - contiene valori con due stati: vero o falso.**
-

# LE VARIABILI

***type variableName = value;***

**Type è uno dei tipi di Java e variableName è il nome della variabile  
(rappresenta la dichiarazione della variabile).**

**Il segno uguale viene utilizzato per assegnare un valore alla variabile  
(rappresenta l'inizializzazione).**

- 1. int provaNumero = 12;**
- 2. String provaTesto = "Ciao Mondo";**
- 3. boolean provaBool = true;**

**Dichiarazione e inizializzazione possono avvenire in momenti differenti:**

- 1. boolean provaBool;**
- 2. provaBool = false;**
- 3. provaBool = true;**

**Per non far sovrascrivere il valore della variabile, bisogna usare la parola chiave **final** dichiarando la variabile come costante (significa immutabile e di sola lettura).**

- 1. final int provaNumero = 15;**
- 2. provaNumero = 20;**

**Questo esempio genera un errore ovvero:  
"error: cannot assign a value to a final variable"**

**È possibile usare `println` per stampare a schermo le variabili.**

- 1. `String testo = "Prova"`**
- 2. `System.out.println(testo);`**

**Si può combinare il testo con una variabile utilizzando il carattere +**

- 1. `testo = "Mondo";`**
- 2. `System.out.println("Ciao " + testo);`**

**È possibile farlo anche all'interno di una variabile e anche tra variabili**

- 1. `String firstPart = "Ciao ";`**
- 2. `String lastPart = "Mondo";`**
- 3. `String fullPart = firstPart + lastPart;`**
- 4. `System.out.println(fullPart);`**

**Per i valori numerici il + funziona come un operatore matematico.**

- 1. int x = 5;**
- 2. int y = 6;**
- 3. System.out.println(x + y);**

**Per dichiarare più variabili dello stesso tipo è possibile utilizzare un elenco separato da virgolette:**

- 1. int x = 5, y = 6, z = 50;**
- 2. System.out.println(x + y + z);**

**È inoltre possibile assegnare lo stesso valore a più variabili in una riga:**

- 1. int x, y, z;**
- 2. x = y = z = 50;**
- 3. System.out.println(x + y + z);**

**Tutte le variabili in Java devono essere identificate con nomi univoci. Questi nomi univoci sono chiamati identificatori, possono essere nomi brevi (come x e y) o nomi più descrittivi (anni, somma, totale).**

**Nota: si consiglia di utilizzare nomi descrittivi per scrivere codice comprensibile e gestibile.**

- 1.// Buono**
- 2.int minutiPerOra = 60;**
- 3.// Non è facile da intuire**
- 4.int m = 60;**