INTRODUZIONE ALLA PROGRAMMAZIONE AD ALTO LIVELLO

IL LINGUAGGIO JAVA

- **Un programma è una formulazione testuale di un algoritmo in un particolare linguaggio di programmazione.**

PROGRAMMA = DATI + CONTROLLO

- Il "potere espressivo" di un linguaggio è caratterizzato da:
  - quali **tipi di dati** consente di rappresentare direttamente o tramite definizione dell'utente (numeri, caratteri, valori logici, stringhe, strutture, ecc.);
  - quali **istruzioni di controllo** mette a disposizione (quali operazioni e in quale ordine di esecuzione).

Perché JAVA

Caratteristiche principali di Java

- Java è un linguaggio object oriented basato su: **classi**, **oggetti**, **metodi**.
- **Un linguaggio orientato agli oggetti**: i dati sono rappresentati come **oggetti** e le operazioni come **metodi** che operano su essi.
- **Pensato per lo sviluppo di applicazioni in rete.**
  - Semplice
  - Robusto
  - Architecture neutral (Indipendente dalla piattaforma)
  - Sicuro.

Perché JAVA

- Sintassi simile a C e C++
- Elimina i costrutti più "pericolosi" di C e C++
  - aritmetica dei puntatori
  - (de)allocazione esplicita della memoria
  - strutture (struct)
  - definizione di tipi (typedef)
  - preprocessore (#define)
- Aggiunge garbage collection automatica
- Conserva la tecnologia OO di base di C++

Concetto di Classe in Java

- **Classe** : **Collezione di oggetti e metodi**

- Una **classe** in Java definisce un **insieme di oggetti con le stesse caratteristiche**.
- Ad esempio:
  - la **classe libro** : insieme degli oggetti libro.
  - la **classe matrice**: l'insieme degli oggetti matrice.
  - la **classe moneta** : l'insieme degli oggetti moneta.

- Il concetto di classe è una estensione del concetto di **tipo** dei linguaggi imperativi.

Oggetti e Metodi in Java

- **Oggetti**

elementi di una classe con uguali caratteristiche e sui quali possono operare i

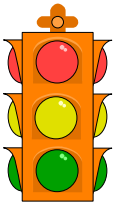
- **Metodi**

definiti per la classe a cui loro appartengono.

- Esempio: **oggetto SEMAFORO**

Funzioni (Metodi)
- **Accendi**
- **Spegni**
- **Diventa rosso**
- **...**

Dati:
- **Colori**
- **Peso**
- **Tecnologia**
- **...**



Struttura di un programma Java

- Un programma Java consiste in un **insieme di definizioni di classi**.
- In genere ogni classe è definita in un file separato (compilabile separatamente dagli altri file). Tuttavia tutto il programma può risiedere in un unico file.
- Almeno una delle classi che appartengono all'applicazione deve *esportare* un metodo `main`.
- Il programma viene eseguito chiamando l'interprete/compilatore con un parametro che specifica la classe che contiene il metodo `main`.

Primo esempio di un programma Java

- **Semplicissimo programma che stampa la stringa *Ciao*.**

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

- `PrimoProgramma` **è il nome della classe (*attenti alle maiuscole*)**

Primo esempio di un programma Java

- Il primo rigo del programma definisce una classe di nome ***PrimoProgramma***

```
public class PrimoProgramma
```

- La parola chiave `public` indica che la classe può essere utilizzata dalle altre classi.
- La parola chiave `class` indica la definizione del nome (***PrimoProgramma***) e del contenuto della classe.
- Il nome della classe deve corrispondere al nome del file che la contiene. In questo caso: **PrimoProgramma.java**

Il metodo main

- La semplice classe **PrimoProgramma** non definisce oggetti ma definisce un metodo essenziale per poter eseguire il programma : **il metodo** `main`.
- Un metodo definisce le operazioni da eseguire come una sequenza di istruzioni che eseguono uno specifico compito o calcolano un particolare risultato.
- Senza metodi non avremmo operazioni !
- Il metodo `main` deve essere utilizzabile da tutti e quindi deve essere dichiarato `public`.

Il metodo main

- La parola chiave `static` indica che il metodo non accede e non modifica i metodi della classe.
- Le parole chiave `void` e `args[]` le analizzeremo più avanti.
- Le parentesi `{ }` indicano l'inizio e la fine di una classe, di un metodo o di un blocco di operazioni.
- Le operazioni terminano con il `;`

Operazioni e invocazione di un metodo

- Il metodo `main` contiene una singola operazione

```
System.out.println("Ciao");
```

per visualizzare sullo schermo la parola Ciao.

- `System` indica una classe predefinita per operare sulle risorse del sistema che contiene oggetti e metodi.
- `out` è un oggetto che indica lo standard output (lo schermo).
- `println` è un metodo della classe System per scrivere sullo standard output e andare a capo (`print` non va a capo).

Invocazione di un metodo

- L'esecuzione di un metodo si richiede nel modo seguente:

```
oggetto.metodo(parametri);
```

- che significa esegui le operazioni definite dal `metodo` sull'`oggetto` usando i `parametri` indicati.
- I parametri possono essere assenti, ma le parentesi vanno inserite comunque. Se ci sono più parametri si separano con una virgola.
- Nel programma si vuole stampare una stringa e quindi la si indica tra apici: "Ciao", per differenziarla da un identificatore o una parola chiave.

Compilazione ed esecuzione

- Per eseguire questo programma che sta nel file **PrimoProgramma.java** usando il compilatore JDK:

COMPILAZIONE:

- `C:> javac PrimoProgramma.java`

ESECUZIONE:

- `C:> java primo`
- **Attenti agli errori di sintassi e di semantica !!**

Dati e variabili

- Nel programma che abbiamo discusso non sono state usate locazioni di memoria per conservare dei dati.
- Quando questo è necessario occorre definire delle **variabili**.
- **Una variabile è un'astrazione della cella di memoria.**
- Formalmente, una **variabile** è un **simbolo** associato a un indirizzo fisico che **denota un valore.**

Dati e variabili

- Una **variabile è un contenitore che può conservare un valore**. Essa viene realizzata tramite una o più celle di memoria (variabile semplice o strutturata).
- Ad esempio nel caso:

x → **4** *1328*

- l'indirizzo di **x** è *1328* e il suo valore è attualmente **4**.
- **Attenzione:**
  - il valore può cambiare nel corso dell'esecuzione,
  - l'indirizzo è fissato (e non cambia durante l'esecuzione).
  - Il programmatore non ha bisogno di conoscere dove la variabile sia memorizzata (il suo indirizzo).

Dichiarazione di una variabile

- La dichiarazione di una variabile introduce una nuova variabile, identificata da un **simbolo** (**nome**), e da un **tipo** che definisce le caratteristiche e le operazioni che si possono effettuare sulla variabile**.**
- ESEMPI
  - `int a, b, sum;`
  - `double num, cifra;`
  - `char ch;`
- ATTENZIONE: **definendo** la variabile, si **dichiara** il suo tipo e contemporaneamente si **alloca** il relativo spazio in memoria.

Inizializzazione di una variabile

- Opzionalmente, nella dichiarazione è possibile specificare un **valore iniziale** per una nuova variabile :
- ESEMPIO
  - `int a, b = 8, sum = 0;`
  - `float pi = 3.14;`
- La variabile verrà creata con il suo valore iniziale (che ovviamente potrà variare).

Operazione di assegnamento (=)

- L'assegnamento è una istruzione che calcola il valore di una espressione e memorizza il valore (lo assegna) **in una variabile**

```
variabile = espressione
```

- ESEMPI:
  - `i = 1;`
  - `i = j + 1 ;`
  - `x = x - y ;`
  - `z = (2*x)-5;`

Operazione di assegnamento (=)

- Il nuovo valore della variabile è quello denotato dall'espressione posta a destra dell'assegnamento.
- Quindi, se ad esempio il valore di k era 2:
  - `k = 10;` **cambia in 10 il valore di** k
  - `j = k + 1;` **cambia in 11 il valore di** j
- Questo nuovo valore può essere usato in altre espressioni.

Uso di variabili - Area di un Rettangolo

- Programma Java che calcola l'area di un rettangolo.

```
public class AreaRettangolo
{
  public static void main(String args[])
  {
    int base, altezza, area;
    base = 5;
    altezza = 8;
    area = base * altezza;
    System.out.println("Area = " + area);
  }
 }
```


Costanti

- Una costante rappresenta un dato che **non può cambiare di valore** nel corso dell'esecuzione.
- La dichiarazione di una costante **associa** ad un identificatore (nome) un valore noto a priori.
- In Java le costanti i dichiarano con la parola chiave **final**
  - `final double pi = 3.14159;`
  - `final int massimo = 10000;`
- Il valore della costante non potrà più essere modificato nel programma, ma verrà utilizzato nelle espressioni. Ad esempio se j è una variabile intera:

```
j = massimo - 1;
```