
1.1 Introduzione Alla Programmazione

Iniziamo con la domanda “*Che cosa significa programmare?*”

Programmare significa creare dei programmi, scrivere dei programmi. *Ma cosa sono i programmi?*

I programmi sono una sequenza di istruzioni che il computer deve eseguire.

Quando scriviamo le istruzioni da far eseguire al computer per risolvere un determinato problema, dobbiamo considerare tutti i possibili casi che il computer deve affrontare. Questo perché dobbiamo sempre tenere a mente che i computer sono fondamentalmente stupidi, o meglio sanno eseguire compiti semplici ad una velocità mostruosa ma non sanno decidere da soli, quindi dobbiamo essere noi a dirgli precisamente cosa devono fare in tutte le possibili situazioni.

Inoltre essi in non posso eseguire tutti i compiti, ma solo quelli privi di ambiguità.

Ad esempio immaginiamo di mettere un computer per aprire le porte di un locale quando arrivano le persone. Se questo compito fosse svolto da una persona gli potremmo dire di fare entrare solo le persone che non sembrano dei pochi di buono. Ma un computer non riesce a capire quando una persona potrebbe essere un poco di buono, è un concetto ambiguo, abbastanza soggettivo.

Se invece decidessimo di far entrare solo le persone più alte di 1 metro e 50, per il computer sarebbe compito perfetto.

Ora abbiamo più o meno una vaga idea di cosa possa essere un programma ma con quale linguaggio possiamo comunicare con il computer?

Il pc, in realtà, capisce solo un linguaggio composto da 0 e 1, che in pratica, semplificando molto, consiste nel mandare corrente elettrica, nel caso di 1, e toglierla, nel caso di 0. 0 e 1 sono anche noti come **sistema binario**.

Con una sequenza di 0 e 1, e codificando in modo opportuno, possiamo dire qualsiasi cosa al pc.

Ad esempio la parola “ciao” la potremmo codificare come la sequenza 010111010101001.

Ma, per una persona, programmare in 0 e 1 è veramente difficile. Per fortuna sono stati inventati vari linguaggi di programmazione che si avvicinano, chi più chi meno, al linguaggio umano.

Dopo questa breve introduzioni, iniziamo con il dare una serie di definizioni essenziali e preparare il nostro pc con i software necessari per iniziare a programmare.

1.2 Algoritmi e Programmi

Un **algoritmo** è insieme di istruzioni elementari che indicano come eseguire dei compiti.

Esempio di Algoritmo Preparazione Pasta

1. Versa 500 ml di acqua in una pentola
2. Accendi il fuoco
3. Metti una pentola sopra il fuoco
4. Aspetta 10 minuti che l'acqua bolla
5. Pesa 500 grammi di la pasta
6. Aggiungi il sale all'acqua
7. Aggiungi la pasta nella pentola
8. Aspetta 10 minuti per la cottura
9. Al termine la scoli
10. Aggiungi al sugo.
11. Mettila nel piatto

Da notare che l'algoritmo si può scrivere anche in italiano.

Seguendo un algoritmo chiunque dovrebbe riuscire a portare a termine il compito.

Esempio - Trovare la chiave giusta in un mazzo di chiavi

- 1) Selezionare una chiave dal mazzo e marcarla con un pennarello
- 2) Tentare di aprire il lucchetto con la chiave appena marcata; se funziona, andare al passo 7)
- 3) Altrimenti, selezionare una chiave dal mazzo
- 4) Se non è marcata, marcarla e tornare al passo 2)
- 5) Se non ci sono chiavi non marcate, la chiave giusta non è presente nel mazzo e andare al passo 7)
- 6) La chiave apre il lucchetto
- 7) Fine della ricerca

Un altro esempio di algoritmo può essere qualsiasi ricetta per cucinare un piatto.

A questo punto ci si potrebbe chiedere cosa c'entrino queste pseudo-istruzioni con i pc. In qualche modo c'entrano perché i **programmi** che eseguono i computer non sono che una sequenza di istruzioni, ovviamente non scritte in italiano ma con un linguaggio di programmazione che il pc può capire.

Quindi, per far svolgere un compito al nostro pc, dobbiamo prima creare un algoritmo, in italiano, e dopo tradurlo nel nostro linguaggio di programmazione, in questo caso java.

Questo inizialmente, poi, man mano che si prende confidenza con la programmazione, si può scrivere direttamente l'algoritmo in java, senza passare prima dall'italiano.

A prima vista può sembrare che la parte più difficile sia tradurre l'algoritmo in java, in realtà è vero il contrario. Trovare l'algoritmo che risolve un problema può essere molto complesso.

Un approccio che aiuta molto è la tecnica del “**dividi et impera**”, ossia se un problema è troppo complesso per essere risolto, lo si divide in sotto-problemi e si cerca di risolvere quelli. Se il sotto-problema è ancora complicato lo si divide ulteriormente, fino a raggiungere qualcosa che è risolvibile. Alla fine si mette insieme tutto e si ottiene la soluzione complessiva.

La descrizione su come si deve risolvere un problema, quindi il suo algoritmo, a parole sembra semplice. Nella realtà, spesso, è proprio l'ostacolo più grande che incontra chi inizia a programmare.

E purtroppo, questa capacità, non la si impara leggendo solamente un libro MA facendo molti, molti, molti esercizi. Non c'è nessun altro modo. Chi dice il contrario mente sapendo di mentire.

Imparare a programmare è come correre una maratona; si può essere esperti di regolamento, del funzionamento del corpo umano, di tutta la teoria sulla corsa ma, se non ci si allena duramente, non si arriverà da nessuna parte.

Ricapitolando, abbiamo visto questi due importanti concetti:

- **Algoritmo** = insieme di istruzioni elementari che indicano come svolgere operazioni complesse.
- **Programma** = algoritmo scritto in un linguaggio “comprensibile” (nel nostro caso Java) dal computer.

1.3 Preparazione Ambiente di Sviluppo

Teoricamente, per poter sviluppare in java è sufficiente installare il tool di java e un semplice editor di testo, come ad esempio notepad, ma spesso i progetti sono composti da centinaia di file e gestirli diventa complicato. Ci vengono in aiuto degli ambienti di sviluppo grafico, chiamati IDE (Integrated Development Enviroment).

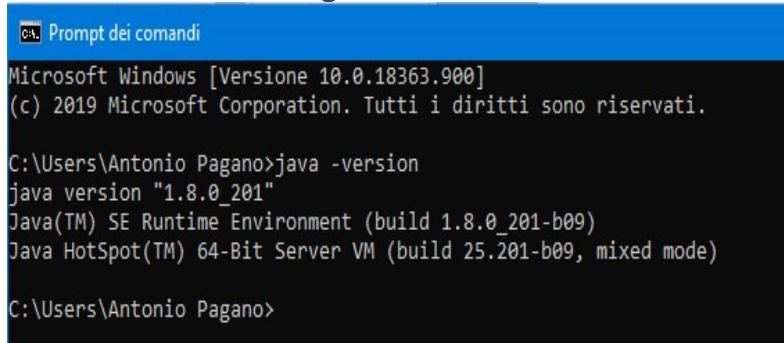
Esistono tantissimi IDE, spesso gratuiti. In questo testo useremo **Eclipse**.

Detto ciò, scarichiamo e configuriamo gli strumenti per iniziare a sviluppare in Java, che sono essenzialmente due:

- **JDK**: E' un tool che ci fornisce una serie di strumenti per lavorare in java. Scaricarlo dal seguente link:

<http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

e cliccare due volte per installarlo. Durante l'installazione premere sempre sì. Se tutto è avvenuto correttamente, apriamo un prompt MSDOS e digitiamo il comando `java -version`



```
Prompt dei comandi
Microsoft Windows [Versione 10.0.18363.900]
(c) 2019 Microsoft Corporation. Tutti i diritti sono riservati.

C:\Users\Antonio Pagano>java -version
java version "1.8.0_201"
Java(TM) SE Runtime Environment (build 1.8.0_201-b09)
Java HotSpot(TM) 64-Bit Server VM (build 25.201-b09, mixed mode)

C:\Users\Antonio Pagano>
```

vedremo cos' la versione di java installata (in questo caso la 8). In caso di errore disinstallare e ripetere la procedura di installazione.

- Eclipse: fare il download dal seguente link, prestando attenzione a scaricare la versione Eclipse EE (enterprise edition)

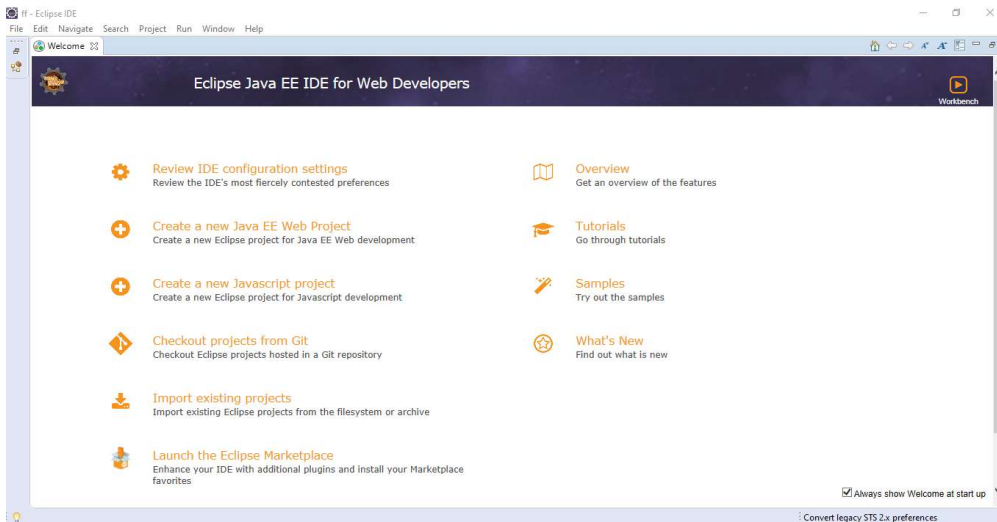
<https://www.eclipse.org/downloads/>

Bene, ora siamo pronti per partire, abbiamo tutti gli strumenti necessari, non ci rimane che configurarli:

1) Lanciamo Eclipse: posizionamoci nella cartella in cui è stato scaricato e cliccare due volte sul file "eclipse.exe".

Durante l'avvio, selezionare il **workspace**, ossia la cartella in cui salvaremo i nostri progetti.

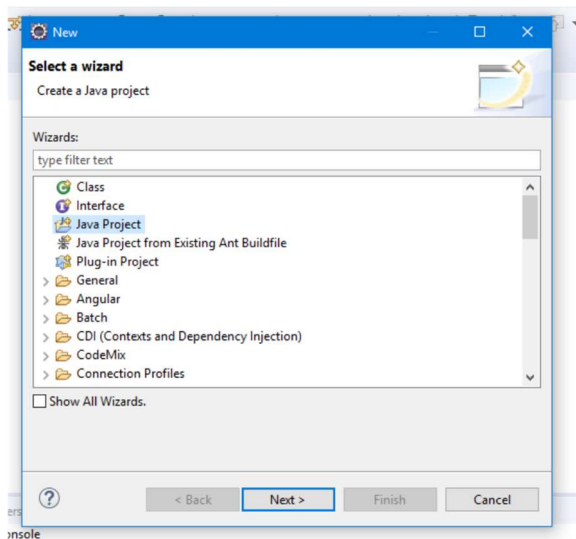
Se tutto è andato a buon fine, vedremo una finestra del genere:



2) Creiamo il progetto che servirà da base per le esercitazioni: dal menù, selezionare

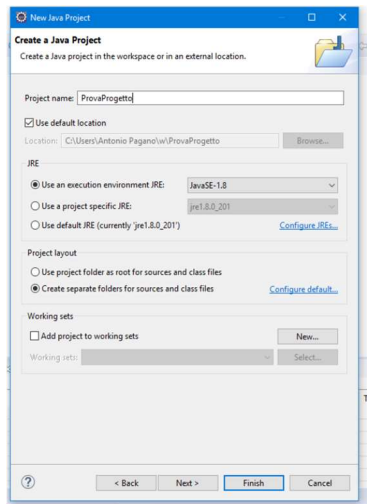
File → New → Other

Dalla seguente finestra:

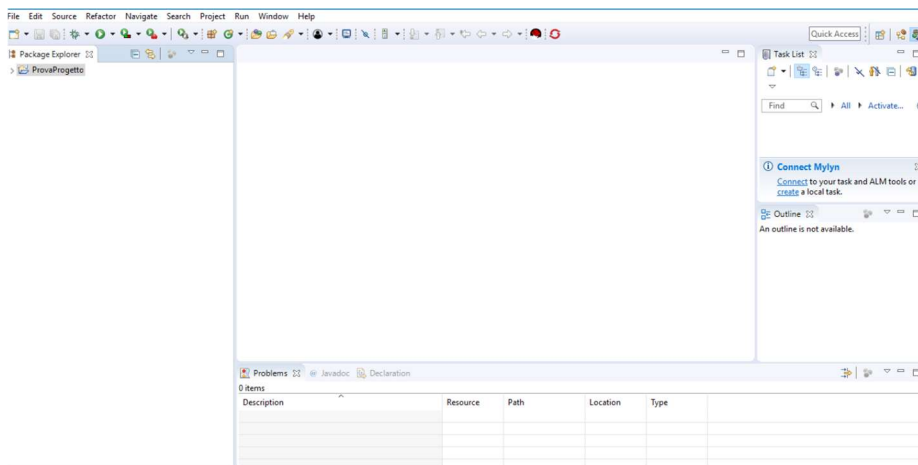


Cliccare ***“Next”***.

Nella nuova schermata inserire nel campo ***“Project Name”*** il valore ***“ProvaProgetto”*** e premere ***“Finish”***.



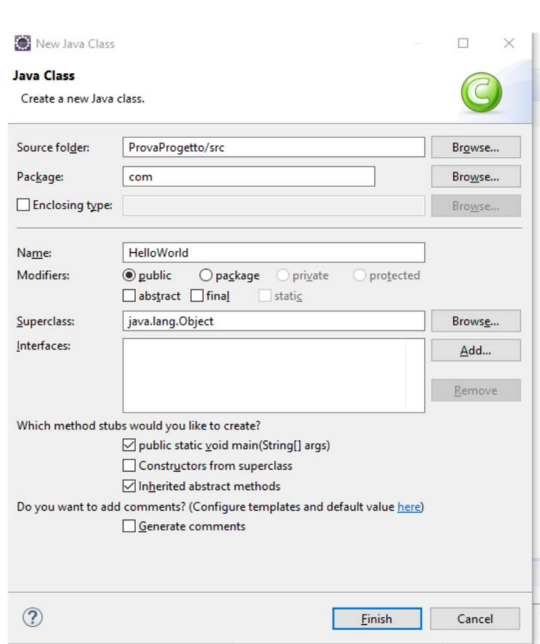
Se tutto è andato a buon fine, sul lato sinistro dovrebbe apparire il nostro primo progetto java, di nome “ProvaProgetto”.



3) Creazione classe di prova: selezionare il progetto “ProvaProgetto”, e dal menù cliccare su:

File→New→Class

Si aprirà una finestra così fatta:



Inserire i seguenti valori:

Package = com

Name = HelloWorld

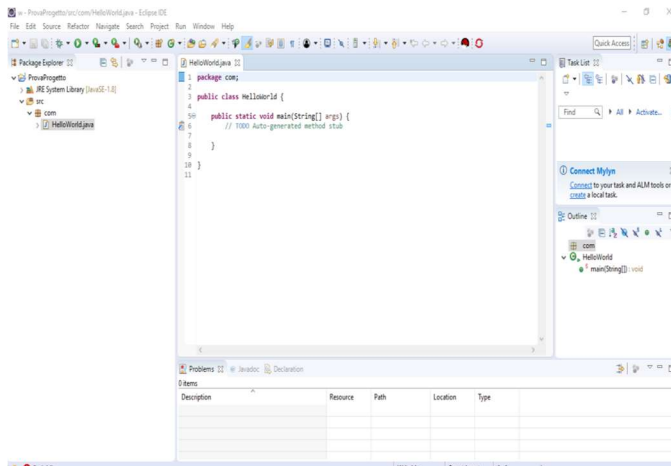
Selezionare la checkbox:

“public static void main(String[] args)”

Premere **“Finish”**.

Se tutto è andato a buon fine abbiamo creato un nuovo File chiamato “HelloWorld.java”.

Eclipse dovrebbe avere questo aspetto:



Useremo il file appena creato per fare moltissimi esercizi, come vedremo nei capitoli successivi.

1.4 Le variabili

Per scrivere applicazioni Java, la prima cosa che si deve sapere è creare e usare le variabili. Ma cosa sono le variabili?

Una variabile è una piccola area di memoria in cui viene salvato un determinato valore. La possiamo vedere come se fosse una scatola in cui possiamo memorizzare dei valori, ad esempio possiamo inserirci dei numeri o del testo.

Per essere utilizzata, deve essere prima **dichiarata** e poi **inizializzata**.

Per la dichiarazione ha la seguente sintassi:

```
<tipo_variabile> <nome_variabile>;
```

Il <tipo_variabile> indica che cosa si deve memorizzare nella variabile, ad esempio un numero, un numero con virgola, un testo etc. (più avanti vedremo i tipi dati in java).

Esempio

```
int x;
```

con questa istruzione abbiamo creato una variabile con nome x che può contenere dei numeri, in particolare interi, senza virgola. I numeri in java sono di tipo **int** (più avanti li vedremo nel dettaglio).

Nota Bene: ogni istruzione Java termina con un punto e virgola “;”.

Ora che la variabile è stata creata, abbiamo la possibilità di inserire dei valori, che devono essere dello stesso tipo della variabile; ad esempio se la variabile è di tipo intero possiamo inserire solo degli interi.

Per poter fare questo, la sintassi è:

<nome_variabile> = <valore>

Esempio:

```
x = 10;
```

In questo modo si scrive il valore 10 all'interno della variabile chiamata "x".

L'operazione di dichiarazione e inizializzazione può essere fatta sulla stessa riga.

Esempio

```
int x = 20;
```

Una volta che la variabile è stata inizializzata, possiamo inserire nuovi valori, sovrascrivendo i vecchi, quante volte vogliamo.

Esempio

x = 10;

x = 30;

x=40;

1.5 Tipi Primitivi

Java ha un insieme di tipi di dati pronti all'uso, detti **tipi primitivi**:

- **byte**: numeri interi da -128 a 127 incluso. In memoria occupa 8 bit.
- **short**: da -32768 a 32767 incluso. 16 bit
- **int**: da -2147483648 a 2147483647 incluso. 32 bit
- **long**: da -9223372036854775808 a 9223372036854775807 incluso. 64 bit. In genere si aggiunge il suffisso l o L. es. long prova = 200L;
- **float**: numeri con la virgola, fino a 7 cifre dopo la virgola. Occupano 32 bit. In genere si aggiunge il suffisso f o F.
- **double**: numeri con la virgola, fino a 16 cifre dopo la virgola. Occupano 64 bit. In genere si aggiunge il suffisso d o D. Se non è specificato, un numero con la virgola è considerato un double.
- **boolean**: rappresenta i valori vero (true) o falso (false);
- **char**: per i caratteri singoli.

Esempio

```
boolean flag = true;
flag = false;
byte range;
range = 124;
short temperature;
temperature = -200;
```

```
long r = -42332200000L; // i numeri long finiscono con la lettera L
double number = -42.3;
float number1 = -42.3f; // i numeri long finiscono con la lettera f

char ch = 'a';
```