Impariamo a programmare in Java™

Lezione 2

Gr.it

Tipo di programmazione

- ***Istruzione Macchina***

L'insieme di istruzioni macchina (***instruction set***) è **specifico** di una particolare CPU: quello di un Pentium™ è diverso da quello di uno SPARC™

Per eseguire un programma in un computer è necessario **scrivere all'interno della memoria primaria le configurazioni di bit corrispondenti alle istruzioni macchina del programma.** Per fare ciò è necessario conoscere tutti i codici numerici delle istruzioni macchina, Questa operazione lunga e noiosa (che veniva eseguita agli albori dell'informatica) è stata presto automatizzata da *un programma in esecuzione sul computer stesso*, detto ***assemblatore*** (***assembler***) → ***Linguaggio a Basso Livello***

Tipo di programmazione

- ***Linguaggi ad Alto Livello***
  - Negli anni ‘50 furono inventati i primi linguaggi di programmazione ***ad alto livello***
    FORTRAN: primo “vero” linguaggio

    BASIC, **COBOL**
    Anni ‘60 e ‘70: programmazione strutturata

    Pascal (Niklaus Wirth, 1968)

    C (Brian Kernigham e Dennis Ritchie, 1970-75)
    Anni ‘80 e ‘90, programmazione orientata agli oggetti

    C++ (Bjarne Stroustrup, 1979)

    Java (James Gosling e Patrick Naughton, 1991)

Tipo di programmazione

- Linguaggi ad Alto Livello
  - Un programma, detto ***compilatore***, legge il programma in linguaggio ad alto livello e genera il corrispondente programma nel linguaggio macchina di una specifica CPU
  - I linguaggi ad alto livello sono

indipendenti dalla CPU

ma il prodotto della compilazione (***codice eseguibile***), non è indipendente dalla CPU

  - occorre compilare il programma con un diverso compilatore per ogni CPU sulla quale si vuole eseguire il programma stesso

Linguaggio Compilato o Interpretato

Un programma viene sviluppato ***scrivendo il codice sorgente***

- Linguaggio Compilato

Linguaggi come il C, C++, Delphi, Visual Basic sono **linguaggi compilati** e seguono questi passi: si scrive il codice sorgente in un editor che viene poi controllato per verificare che non ci siano errori e poi viene compilato, ovvero ogni istruzione viene trasformata nel corrispondente codice in linguaggio macchina che può essere, così, eseguito dal processore

- Linguaggio Interpretato

I linguaggi interpretati, invece, seguono una strada diversa, il codice sorgente viene, appunto, interpretato al volo e vengono, quindi, eseguite le istruzioni così come descritte nel codice sorgente; un esempio su tutti è il PHP il cui codice viene "elaborato" e restituisce una pagina html pura

e Java?

- Un **linguaggio che è a metà strada tra queste metodologie è Java** che è sia compilato che interpretato; il codice sorgente viene compilato in un formato intermedio (chiamato ***bytecode***), il quale a sua volta viene interpretato dalla Java Virtual Machine (JVM), che ha il compito di interpretare “al volo” le istruzioni bytecode in istruzioni per il processore; la JVM viene sviluppata per ogni Sistema Operativo che rende, in pratica, JAVA altamente portabile.

“HELLO WORLD!”

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

Occorre fare molta attenzione

- il testo va inserito esattamente come è presentato (per il momento…)
- maiuscole e minuscole sono considerate distinte
- il file deve chiamarsi Hello.java

“HELLO WORLD!”

Il nostro primo programma Java

A questo punto compiliamo il programma

```
javac  Hello.java
```

ed il compilatore genera il file Hello.class

Ora eseguiamo il programma

```
java  Hello
```

ottenendo la visualizzazione del messaggio di saluto sullo schermo

Hello, World!

“HELLO WORLD!”

Ecco la procedura di compilazione ed esecuzione nel dettaglio:
Aprire il «Prompt dei comandi» da Windows (digita cdm dopo aver cliccato su start)

```
Prompt dei comandi
Microsoft Windows [Versione 10.0.22621.608]
(c) Microsoft Corporation. Tutti i diritti riservati.

C:\Users\gregu>cd c:/java

c:\java>javac Hello.java

c:\java>java Hello
Hell, World!
```

Al posto di «*java*» va il nome della cartella dove avete salvato i vostri codici sorgenti

Al posto di «*Hello*» va il nome del file sorgente che dovete compilare ed eseguire

ANALIZZIAMO IL NOSTRO PROGRAMMA

La prima riga

```
public class Hello
```

definisce una nuova ***classe***

- Le classi sono ***fabbriche di oggetti*** e rappresentano un concetto fondamentale in Java, che è un linguaggio di programmazione ***orientato agli oggetti*** (**OOP**, ***O**bject-**O**riented **P**rogramming*)

- Per il momento, consideriamo gli ***oggetti*** come ***elementi da manipolare in un programma Java***

ANALIZZIAMO IL NOSTRO PROGRAMMA

La ***parola chiave*** **public** indica che la classe **Hello** può essere utilizzata da tutti (in seguito vedremo **private**…)

```
public class Hello
```

- Una parola chiave è una parola riservata del linguaggio che va scritta esattamente così com'è e che non può essere usata per altri scopi
- La parola chiave **class** indica che inizia la **definizione** di una ***classe***
- Ciascun ***file sorgente*** (parte di un programma Java) può contenere ***una sola classe pubblica***, il cui nome ***deve coincidere*** con il nome del file

ANALIZZIAMO IL NOSTRO PROGRAMMA

- Per ora non usiamo le classi come fabbriche di oggetti, ma come ***contenitori di metodi***
- Un metodo serve a definire una sequenza di istruzioni o ***enunciati*** che ***descrive come svolgere un determinato compito*** (in altri linguaggi i metodi si chiamano *funzioni* o *procedure*)
- Un metodo ***deve*** essere inserito in una classe, quindi le classi rappresentano il meccanismo principale per l'organizzazione dei programmi

ANALIZZIAMO IL NOSTRO PROGRAMMA

La costruzione

```
public static void main(String[] args)
{    …
}
```

Definisce il metodo **main** *(principale)*

- Un'applicazione Java ***deve*** avere un metodo **main**
- Anche qui, **public** significa ***utilizzabile da tutti***
- Invece, **static** significa che il metodo **main** ***non esamina e non modifica gli oggetti della classe*** **Hello** a cui appartiene

PROGRAMMA SEMPLICE

Sintassi

```
public class NomeClasse
{public static void main(String[] args)
   { enunciati }
}
```

- Scopo: eseguire un programma semplice, descritto da ***enunciati*** e contenuto nel file ***NomeClasse*.java**
- Nota: la parte in **blu** viene per ora considerata una ***infrastruttura necessaria***, approfondita in seguito

PROGRAMMA SEMPLICE

- Gli enunciati del ***corpo*** di un metodo (*gli enunciati contenuti tra le parentesi graffe*) vengono eseguiti uno alla volta ***nella sequenza in cui sono scritti***
- Ogni enunciato termina con il carattere `;`
- Il metodo **main** del nostro esempio ha un solo enunciato che visualizza una riga di testo

```
{ System.out.println("Hello World!"); }
```

- Ma ***dove*** la visualizza? Un programma può inserire testo in una finestra, scriverlo in un file o anche inviarlo ad un altro computer attraverso Internet…

PROGRAMMA SEMPLICE

```
{ System.out.println("Hello World!"); }
```

Nel nostro caso la destinazione è *l'output standard* (o "flusso di uscita standard"): è una proprietà di ciascun programma che dipende dal sistema operativo del computer

In Java, l'output standard è rappresentato da un *oggetto* di nome out come ogni metodo, *anche gli oggetti devono essere inseriti in classi*: out è inserito nella classe System della libreria standard, che contiene oggetti e metodi da utilizzare per accedere alle *risorse di sistema*. Per *usare* l'oggetto out della classe System si scrive

```
System.out
```

PROGRAMMA SEMPLICE

Quando si usa un oggetto, bisogna specificare

cosa si vuol fare con l'oggetto stesso

in questo caso vogliamo usare un metodo dell'oggetto out, il metodo println, che stampa una riga di testo

per usare il metodo println dell'oggetto System.out si scrive

```
System.out.println(parametri)
```

la coppia di parentesi tonde racchiude le informazioni necessarie per l'esecuzione del metodo (***parametri***)

A volte il carattere ***punto*** significa "usa un oggetto di una classe", altre volte "usa un metodo di un oggetto": dipende dal contesto...

PROGRAMMA SEMPLICE

Sintassi:

```
Oggetto.nomeMetodo(parametri)
```

Scopo: invocare il metodo ***nomeMetodo*** dell'***oggetto***, fornendo ***parametri*** se sono richiesti

Nota: se non sono richiesti parametri, le parentesi tonde devono essere indicate ugualmente

Nota: se ci sono due o più parametri, essi vanno separati l'uno dall'altro con una ***virgola***, all'interno delle parentesi tonde

PROGRAMMA SEMPLICE

Diversamente da altri linguaggi (es. FORTRAN) il linguaggio Java consente una ***disposizione del testo a formato libero***: gli spazi e le interruzioni di riga ("andare a capo") non sono importanti, tranne che per separare parole

Sarebbe quindi possibile scrivere

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

Bisogna però ***fare attenzione alla leggibilità!***