

Programmazione I

A.A. 2002-03

linguaggio Java

(*Lezione X, Parte I*)

Il primo programma

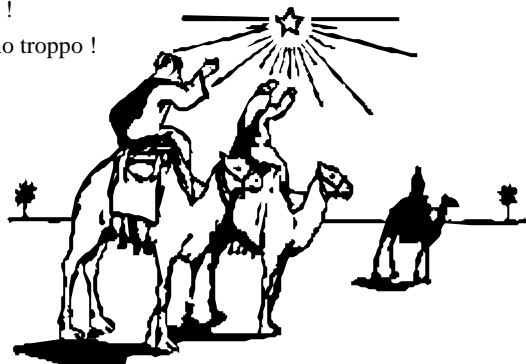
Prof. Giovanni Gallo
Dr. Gianluca Cincotti

Dipartimento di Matematica e Informatica
Università di Catania

e-mail : { [gallo](mailto:gallo@dmf.unict.it), [cincotti](mailto:cincotti@dmf.unict.it) } @dmf.unict.it

La velocità di una carovana...

- ... è quella del cammello più lento;
 - Il ritmo delle lezioni cerca di mediare tra diverse esigenze.
 - Gli impazienti pazientino !
 - I rallentatori non rallentino troppo !



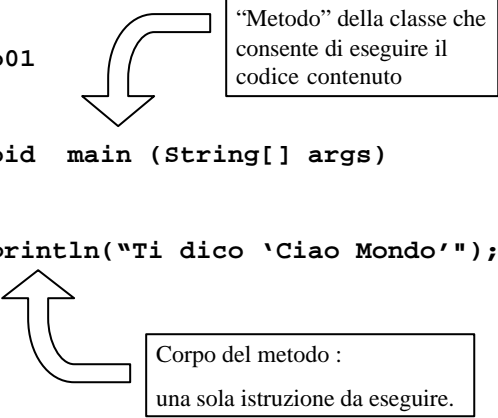
Per iniziare ...

➤ Osservazioni di carattere generale:

- Java è case-sensitive :
 - cioè distingue tra maiuscole e minuscole.
- Le parentesi graffe contengono le parti (i “*blocchi*”) di programma.
- Tutto in Java è dentro una classe.
- Ogni istruzione termina con un punto e virgola.

Il primo programma in Java !

```
public class Esempio01
{
    public static void main (String[] args)
    {
        System.out.println("Ti dico 'Ciao Mondo'");
    }
}
```



“Metodo” della classe che consente di eseguire il codice contenuto

Corpo del metodo :
una sola istruzione da eseguire.

Alcune osservazioni

- La parola riservata **“public”** si dice *modificatore di accesso*.
- Il nome di ogni classe in Java inizia con maiuscola per convenzione.
 - Il nome del programma (Esempio01) deve essere uguale al nome con cui si salva il codice fatto seguire da “.java”
- L'esecuzione di un programma inizia sempre dal metodo **“main”**.
 - Occhio alle parentesi graffe !

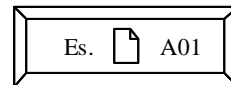
Standard output

- L'oggetto **“system.out”** si riferisce allo standard output.
 - Il metodo **“println”** oppure **“print”** può essere invocato per esso, in modo da visualizzare quanto contenuto tra le parentesi.
 - La sintassi della chiamata ad un metodo contenuto in un oggetto è sempre del tipo
 - ❖ <nome oggetto>.<nome metodo> (parametri);
 - Anche se un metodo non accetta argomenti le parentesi vanno messe comunque: “()” .

Commenti

➤ Esistono tre metodi per inserire *commenti* all'interno dei programmi:

- Commenti di riga :
 - // ...
- Commenti lunghi (anche su più righe!) :
 - /* ...
 - ...
 - */
- Commenti di documentazione :
 - /** ... */



Precauzioni nell'uso dei commenti

È consuetudine tra i programmatori isolare una parte di codice che non si intende eseguire momentaneamente inglobandolo tra comandi di commento. Questo non è auspicabile perché la sintassi di Java non consente annidamento di commenti del tipo lungo.

```
public static void main (String[] args)
{
    System.out.println("Uno");
    /* System.out.println(" e Due");
        /* visualizza la seconda stringa */
    */
}
```

Librerie di classi

- Una *libreria* (o package) è una collezione di classi che possono essere usate nei programmi
- La *libreria standard* “java.lang” fa parte di ogni sistema di sviluppo Java
 - La classe **System** e la classe **String** sono parte di essa.
 - Altre librerie possono essere prodotte da terze parti o sviluppate da voi stessi.

Librerie di classi (cont.)

<u>Package</u>	<u>Ambito</u>
<code>java.lang</code>	Supporto generale allo sviluppo <i>(importato sempre automaticamente)</i>
<code>java.applet</code>	Applets per il web
<code>java.awt</code>	Abstract Windowing Toolkit
<code>javax.swing</code>	Graphical User Interface (GUI)
<code>java.net</code>	Comunicazione di rete
<code>java.util</code>	Utilità varie
<code>java.text</code>	Testo formattato
<code>java.math</code>	Operazioni matematiche

Utilizzare i pacchetti

- Per usare una classe di un pacchetto, la si deve prima importare:

```
import <package>.<Classe>;  
import java.util.Random;
```

- Per importare tutte le classi di un pacchetto si usa il carattere asterisco “*”:

```
import java.util.*;
```

Fine

Programmazione I

A.A. 2002-03

Linguaggio Java

(*Lezione X, Parte II*)

Tipi di dati primitivi

Prof. Giovanni Gallo

Dr. Gianluca Cincotti

Dipartimento di Matematica e Informatica

Università di Catania

e-mail : { [gallo](mailto:gallo@dmf.unict.it), [cincotti](mailto:cincotti@dmf.unict.it) } @dmf.unict.it

Le variabili

- I programmi elaborano informazioni di molti tipi differenti.
 - Differenti informazioni richiedono un numero differente di celle di RAM per la memorizzazione.
- Ogni volta che si introduce una **VARIABILE**
 - (cioè un nome simbolico per indicare una locazione di memoria in cui sarà conservato un valore di interesse)occorre “dire” al compilatore quanta memoria deve essere associata a tale variabile.

Dichiarazione di variabili

- Una *variabile* è un *dato* individuato da un *identificatore*, che rappresenta l'indirizzo della cella di memoria in cui il dato è archiviato.
 - Una variabile deve essere *dichiarata*, specificandone l'identificatore e il tipo di informazione che deve contenere

Tipo del dato Nome della variabile

```
int totale;  
int contatore, temp, risultato;
```

- Quando in un programma si richiama una variabile si usa il suo valore

Identificatori

- Gli *identificatori* sono le parole usate dal programmatore
 - Un identificatore è composto
 - da lettere, cifre, il carattere underscore '_' ed il simbolo '\$'
 - non può iniziare con una cifra
 - Java è sensibile alle maiuscole, *case sensitive*
 - **Totale** e **totale** sono identificatori diversi
- Alcuni identificatori speciali sono detti *parole riservate* e hanno un significato prestabilito
 - Una parola riservata non può essere ridefinita

Parole riservate

<code>abstract</code>	<code>default</code>	<code>goto</code>	<code>operator</code>	<code>synchronized</code>
<code>boolean</code>	<code>do</code>	<code>if</code>	<code>outer</code>	<code>this</code>
<code>break</code>	<code>double</code>	<code>implements</code>	<code>package</code>	<code>throw</code>
<code>byte</code>	<code>else</code>	<code>import</code>	<code>private</code>	<code>throws</code>
<code>byvalue</code>	<code>extends</code>	<code>inner</code>	<code>protected</code>	<code>transient</code>
<code>case</code>	<code>false</code>	<code>instanceof</code>	<code>public</code>	<code>true</code>
<code>cast</code>	<code>final</code>	<code>int</code>	<code>rest</code>	<code>try</code>
<code>catch</code>	<code>finally</code>	<code>interface</code>	<code>return</code>	<code>var</code>
<code>char</code>	<code>float</code>	<code>long</code>	<code>short</code>	<code>void</code>
<code>class</code>	<code>for</code>	<code>native</code>	<code>static</code>	<code>volatile</code>
<code>const</code>	<code>future</code>	<code>new</code>	<code>super</code>	<code>while</code>
<code>continue</code>	<code>generic</code>	<code>null</code>	<code>switch</code>	

Variabili con il nome corto o lungo ?

➤ Le seguenti regole sono **CONSIGLI** per un codice più standard e leggibile ... non obbligatorie ma molto utili :

- Una variabile usata spesso ma solo in una piccola zona del programma può avere un nome corto e non significativo:
 - j, k, h, ...
- Una variabile usata dappertutto in un programma di grande complessità dovrebbe avere un nome più lungo ed espressivo:
 - totaleSpeseOrdinarie, uovaNelPaniere, minimoIntero, ...

Inizializzazione di variabili

➤ Formato generale :

<tipo della variabile> <nome della variabile> = <valore iniziale>;

```
int somma = 0;  
int base = 32, max = 149;
```

- L'assegnazione di un valore iniziale è facoltativa (ma consigliabile).
- A differenza di altri linguaggi (per es. Pascal) la dichiarazione di una variabile può essere fatta in qualsiasi momento dentro il programma.

Tipi di dati e Java

➤ Java è un linguaggio *fortemente tipizzato*

- Il tipo di ogni variabile o espressione può essere identificato leggendo il programma ed è già noto al momento della compilazione
- È obbligatorio dichiarare il tipo di una variabile prima di utilizzarla

Tipi di dato

Ogni tipo di dato ha:

- Un *nome*
 - **Es.:** `int`, `double`, `char`
- Un *insieme di valori* letterali possibili
 - **Es.:** `3`, `3.1`, `'c'`
- Un *insieme di operazioni* lecite
 - **Es.:** `+`, `*`
- In Java ci sono:
 - Tipi primitivi (pre-definiti)
 - Tipi di oggetti o riferimenti a oggetti

Tipi di dati primitivi

- Numeri interi
 - `byte`, `short`, `int`, `long`
- Numeri decimali in virgola mobile
 - `float`, `double`
- Caratteri
 - `char`
- Valori booleani
 - `boolean`

Tipi di dati numerici

- La differenza tra i diversi tipi di dato per rappresentare numeri consiste nella occupazione di memoria
 - E quindi nei valori che possono rappresentare:

<u>Tipo</u>	<u>Memoria in byte</u>	<u>Valore min</u>	<u>Valore max</u>
byte	1	-128	127
short	2	-32,768	32,767
int	4	-2,147,483,648	2,147,483,647
long	8	$\sim 10^{19}$	$\sim 10^{19}$
float	4	+/- 3.4×10^{38} con 7 cifre significative	
double	8	+/- 1.7×10^{308} con 15 cifre significative	

Assegnamento

- Un' *istruzione di assegnamento* modifica il valore di una variabile
- L'operatore di assegnamento è indicato con =

```
totale = 55;  
  ↑     ↓
```

- L'espressione alla destra del carattere = è valutata ed il suo valore viene assegnato alla variabile a sinistra
- L'eventuale valore precedente di `totale` è sovrascritto
- Si possono assegnare solo valori compatibili con il tipo dichiarato

Notazioni numeriche

- Numeri:
 - senza parte frazionaria (es. 301);
 - con parte frazionaria (es. 13.12) o in notazione scientifica (es. 1.5e3);
- REGOLA : Un numero intero è di tipo **int**, una costante in virgola mobile di tipo **double**. Per alterare questa regola si usano dei suffissi:
 - l, L = long, f, F = float, d, D = double;
- Esempi:
 - long 76376446544L
 - float 223.75F , 2.5e-2f
 - double 1322D , 1.26D

In generale :

5 è diverso da 5.0

Notazioni numeriche (cont.)

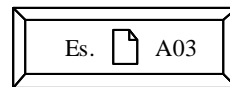
- I numeri interi possono essere introdotti con varie notazioni:
 - decimale: nessun prefisso;
 - ottale: prefisso 0;
 - esadecimale: prefisso 0x.
- Possono essere sia int che long (con suffisso l o L).
- Es.: 102, 022L, 0xFF, 0xA0

Es.  A02

Problemi con numeri “floating point”

Esistono valori speciali per indicare *l'infinito positivo* e *l'infinito negativo*, oltre al *valore non definito*. Tali valori “compaiono” quando si eseguono operazioni mal definite tra numeri non interi (es. : divisioni per 0)

- Double.POSITIVE_INFINITY
- Double.NEGATIVE_INFINITY
- Double.NaN (Not a Number)



Ciò che non è variabile...

➤ ... è **costante**, cioè un identificatore il cui valore non può essere modificato dopo la sua dichiarazione iniziale.

- Il compilatore segnala un errore se si cerca di modificare una costante.

➤ Si dichiara con il modificatore **final**

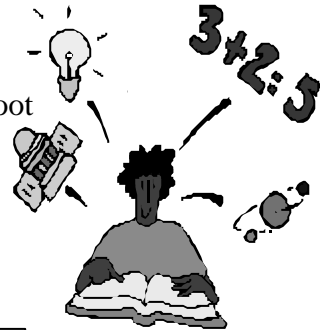
```
final int MAX = 100;
```

- E' convenzione scrivere il nome delle costanti con caratteri TUTTI MAIUSCOLI.
- Facilitano la manutenzione e prevengono errori involontari.

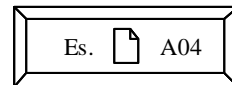
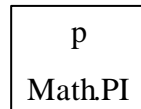
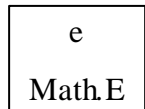
La matematicaaaaa !!!

Le funzioni matematiche meno banali sono definite nella classe “*Math*” del package “*java.math*”

```
double x = 4.0;
double y = Math.sqrt (x); // Square root
double z = Math.sin (x);
double w = Math.cos (x);
double t = Math.tan (x);
double s = Math.pow (x,y); // xy
```



Idem per exp e log ...



Il tipo carattere

- Una variabile `char` contiene un singolo carattere dell'insieme dei caratteri **Unicode**
 - i caratteri ASCII sono i caratteri UNICODE con codice tra 0 e 255.
- L'insieme **Unicode** usa 2 byte per rappresentare un carattere, consentendo quindi 65,536 caratteri differenti
- I caratteri sono delimitati dai singoli apici ' ' ,

'a' 'x' '7' '\$' ','

- È necessario usare il singolo apice per i caratteri e il doppio apice per le stringhe.
 - 'h' e "h" sono dati diversi, l'uno è un carattere e l'altro è una stringa di lunghezza 1.

Notazione caratteri UNICODE

- Sono codificati tra i valori `\u0000` e `\uffff`
 - La sequenza di escape `\u` indica che il numero che segue è un carattere UNICODE e i numeri sono espressi in esadecimale.
- Inoltre, ci sono alcuni caratteri ASCII speciali:
 - `'\b'` vuol dire backspace
 - `'\n'` vuol dire salto riga
 - `'\''` vuol dire apice `'`
 - `'\"'` vuol dire `"`
 - `'\\'` vuol dire `\`



Il tipo booleano

- Una variabile di tipo `boolean` può assumere solamente uno tra i seguenti valori:

`true`, `false`.
- Esempio: `boolean pronto = false;`
- Nota per i programmatori di C :
 - A differenza del C, in Java non si può usare:
 - zero come equivalente a `"false"` e diverso da zero come equivalente a `"true"`.

Un'anteprima ...

Il tipo *stringa* non è un tipo primitivo.

Le stringhe sono dei veri e propri oggetti !

```
// Questo programma visualizza una variabile stringa
public class EsempioStringa
{
    public static void main (String[] args)
    {
        String frase = "Ecco una frase !";
        System.out.println (frase);
    }
}
```

Fine