Corso di base JAVA

Mauro Donadeo
mail: mauro.donadeo@gmail.com

Scrivere i nostri primi programmi

HT Lab

Introduzione

Cosa tratteremo

In questa parte tenteremo di andare un po' più a fondo scrivendo diversi programmi e tenteremo di capire alcune parti specifiche del Java. Alcune cose sono specifiche del linguaggio Java, ma la maggior parte sono comuni a tutti i linguaggi di programmazione.

Un programma che elabora numeri

```
public class Coins1{
    public static void main(String[] args){
        int lit = 15000; //lire italiane
        double euro = 2.35 //euro;

        //calcola il valore totale
        double totalEuro = euro + lit / 1936.27;

        //Stampa il valore totale
        String outMessage = "Valore totale in euro";
        System.out.print(outMessage);
        System.out.println(totalEuro);
    }
}
```

Le Variabili

- Ogni programma fa uso di variabili;
- le **variabili** sono spazi di memoria, identificate da un **nome**, che possono contenere **valori** di qualsiasi tipo
- ciascuna variabile deve essere **definita**, indicandone il tipo ed il nome;
- Una variabile può contenere soltanto valori dello **stesso tipo**.
- Nella definizione di una variabile è possibile assegnarle un valore iniziale.

Un programma può benissimo risolvere i vari problemi anche senza l'utilizzo di variabili

```
public class Coins2{
    public static void main(String[] args){
        System.out.print("Valore totale in euro");
        System.out.println(2.35 +15000/1936.27);
    }
}
```

Un programma può benissimo risolvere i vari problemi anche senza l'utilizzo di variabili

```
public class Coins2{
    public static void main(String[] args){
        System.out.print("Valore totale in euro");
        System.out.println(2.35 +15000/1936.27);
    }
}
```

ma sarebbe molto meno comprensibile e modificabile con difficoltà

La scelta dei nomi delle variabile è molto importante ed è bene scegliere nomi che descrivono adeguatamente la funzione della variabile

Alcune regole

- deve iniziare con una lettera;
- non può essere una una parola riservata o un simbolo riservato del linguaggio;
- non può contenere spazi;

La scelta dei nomi delle variabile è molto importante ed è bene scegliere nomi che descrivono adeguatamente la funzione della variabile

Alcune regole

- deve iniziare con una lettera;
- non può essere una una parola riservata o un simbolo riservato del linguaggio;
- non può contenere spazi;

Java è case sensitive

Definizione di una variabile

Sintassi

nomeTipo nomeVariabile;
nomeTipo nomeVariabile = espressione;

Scopo: definire la nuova variabile **nomeVariabile**, di tipo **nomeTipo**, ed eventualmente assegnarle il valore iniziale **espressione**

Alcune convenzioni

- i nomi di variabili e metodi **iniziano con la lettera** minuscola;
- i nomi di classi **iniziano con una lettera** maiuscola;

Quando si pensa ad un computer che memorizza una lettera ad esempio la lettera J esso in realtà memorizza la sequenza 01001010, ogni cosa all'interno di un computer è una sequenza di 0 e 1, più comunemente sequenze di bit.

01001010

Questa sequenza inoltre può assumere altri significati:

- Come detto precedentemente la lettera J
- ma anche il numero intero 74;
- $1.036960863003646x10^{-43}$

Il computer distingue il tipo della sequenza utilizzando il concetto di tipo. Il tipo di una variabile è il range di valori che può assumere.

La parola `double` `int` sono esempi in Java (anche come conosciuti *tipi semplici*).

Tipi primitivi di Java

Tipo	Che valore rappresentano	Range di valori
`byte`	`(byte)42`	-128 a 127
`short`	`(short)42`	-32768 a 32767
`int`	`42`	–2147483648 a 2147483647
`long`	`42L`	–9223372036854775808 a 9223372036854775807
`float`	`42.0F`	$-3.4x10^{38} a 3.4x10^{38}$
`double`	`42.0`	$-1.8x10^{308} a 1.8x10^{308}$
`char`	`'A'`	Centinaia di caratteri, simboli
`boolean`	`true`	true, false

Il tipo dati stringa

- Nella programmazione i tipi di dati più importanti sono i **numeri** e le **stringhe**.
- Una **stringa** è una sequenza di caratteri che in Java e molti altri linguaggi è racchiusa tra virgolette **"Hello"**
  - le virgolette non hanno fanno parte della stringa
- Possiamo **dichiarare** e inizializzazione **variabili di tipo stringa**
  - String name = **"John"**
- E' possibile **assegnare un valore** ad una variabile di tipo stringa
  - name = **"Michael"**

Assegnazione

```
public class Coins3{
    public static void main(String[] args){
        int lit = 15000; // lire italiane
        double euro = 2.35; // euro
        double dollars = 3.05; // dollari
        /* calcola il valore totale
       sommando successivamente i contributi*/
       double totalEuro = lit / 1936.27;
       totalEuro = totalEuro + euro;
       totalEuro = totalEuro + dollars * 0.72;
       System.out.print("Valore totale in euro ");
       System.out.println(totalEuro);
    }
}
```

In questo caso il valore della variabile totalEuro **cambia** durante l'esecuzione del programma

- per prima cosa la variabile vine **inizializzata** contestualmente alla sua definizione:

```
double totalEuro = lit /1936.27;
```

- poi la variabile viene **incrementata**, due volte:

```
totalEuro = totalEuro + euro;
totalEuro = totalEuro + dollars * 0.79;
```

Alcune note sintattiche

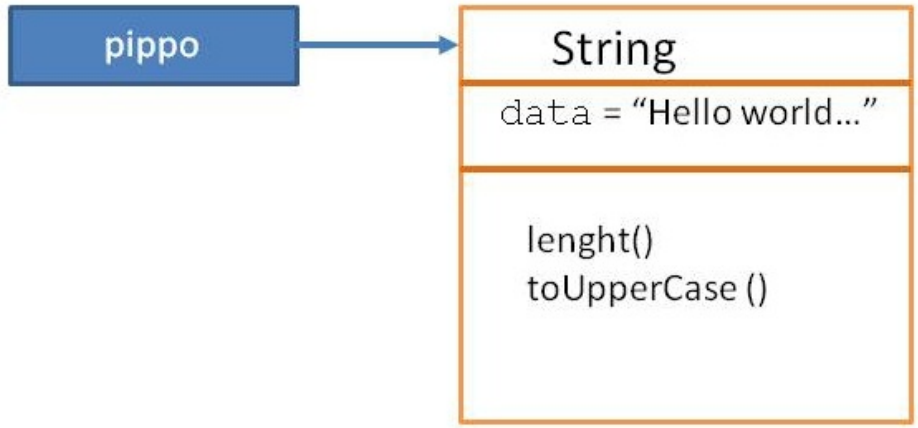
- L'operatore che indica la divisione è /, quello che indica la moltiplicazione è: *;
- Quando si descrivono i numeri in virgola mobile, bisogna utilizzare il **punto** come separatore decimale invece della virgola
- non c'è bisogno di utilizzare il punto per indicare il separatore di migliaia
- i numeri in virgola mobile si possono anche esprimere in notazione esponenziale. Ad esempio:
  - 1.93E3 vale $1.93x10^3$

Oggetti, classi, metodi

Oggetti

Un **oggetto** è un'entità che può essere manipolata in un programma mediante l'invocazione dei suoi **metodi**

- pippo è un oggetto;
- appartiene alla classe **String**
- si può manipolare ad esempio mediante i suoi metodi
  - ad esempio **toUppercase**



Per il momento consideriamo un oggetto come una **black box** dotata di una **interfaccia pubblica**, che definisce il comportamento dell'oggetto, e una sua realizzazione **nascosta** (il codice dei metodi ed i loro dati)

Una classe

- è una **fabbrica di oggetti**;
  - gli oggetti che si creano sono esemplari
- specifica i metodi che si possono invocare per gli oggetti che sono esemplari di tale classe.
- definisce i particolari della realizzazione dei metodi
- è un contenitore di :
  - metodi statici;
  - oggetti statici;

I metodi:

Costituiscono l'interfaccia pubblica di una classe:

- Istruzioni valide:
  - `String pippo = ``Hello world'';`
  - `int n = pippo.lenght();`
  - `String river = ``Missisipi'';`
  - `String BigRiver = river.toUpperCase();`
- Istruzione non valida (il metodo non appartiene alla classe):
  - `System.out.lenght();`

Metodi, parametri espliciti/impliciti

Alcuni metodi necessitano di valori in ingresso che specificano l'operazione da svolgere.

```
System.out.println(pippo)
```

pippo in questo caso rappresenta un parametro esplicito.

Altri metodi invece non necessitano di alcun parametro. Tutte le informazioni sono memorizzate nell'oggetto corrispondente.

```
int n = pippo.length();
```

pippo in questo caso è il parametro implicito.

Definizione dei metodi

public void println(String output)
public String replace(String target,String replace)

La **definizione di un metodo** inizia sempre con la sua **intestazione**, composta da:

- uno specificare di accesso:
  - in questo caso è **public**, ma esiste anche la clusula **private**;
- il tipo di dati restituito dal metodo (String, void, int, double...
- il nome del metodo (println, replace, length)
- un elenco di parametri, eventualmente vuoto, chiuso tra parentesi tonde
  - di ogni parametro si indica il tipo e nome
  - più parametri sono separati da una virgola.

Variabili oggetto

Una variabile oggetto conserva non l'oggetto stesso, ma informazioni sulla sua posizione nella memoria del computer. Sostanzialmente è un riferimento all'oggetto.

Per definire una variabile oggetto si indica il nome della **classe** a cui l'oggetto farà riferimento la variabile, seguito dal nome della variabile stessa.

NomeClasse nomeOggetto

La definizione di una variabile oggetto crea un riferimento **non inizializzato**, cioè la variabile non fa riferimento ad alcun oggetto.

Costruire oggetti: l'operatore new

Per **creare un oggetto** di una classe si usa l'operatore new seguito dal nome della classe e da una coppia di parentesi tonde

```
new NomeClasse(parametri)
```

L'operatore new **crea un nuovo oggetto e ne restituisce un riferimento, che può essere assegnato ad una variabile** del tipo appropiato.

NomeClasse nomeVar = new **NomeClasse(parametri)**

Stringhe = Oggetti

- Diversamente dai numeri, **le stringhe sono oggetti**;
  - infatti, il tipo di dati String inizia con la **maiuscola**
  - invece, int e double iniziano con la minuscola
- Una variabile di tipo stringa quindi può essere utilizzata per invocare i metodi della classe String;
  - per esempio, il metodo length restituisce la lunghezza di una stringa, cioè **il numero di caratteri** presenti in essa.

```
String name = "John";
int n = name.length(); // n = 4
```

ESERCIZIO

Creare un rettangolo descritte dalle coordinate (x,y) del suo vertice in alto a sinistra e dalla larghezza e altezza.

Creare un secondo rettangolo con le stesse caratteristiche del primo, e successivamente traslarlo di (15,20).

Stampare le caratteristiche sia del primo che del secondo rettangolo.

Potete importare la classe Rectangle, presente nel pacchetto in java.awt.Rectangle

Programmi di controllo

Sono utilizzati per collaudare il funzionamento di una classe.

- Definire una nuova classe;
- Definire in essa un nuovo metodo main;
- Costruire oggetti all'interno del metodo main;
- Applicare metodi agli oggetti
- Visualizzare i risultati delle invocazioni dei metodi.

bisogna importare le classi che si vuole utilizzare.

Tutte le classi della libreria standard sono raccolte in **pacchetti** e sono organizzate in pacchetto o finalità. `java.lang` (al quale appartengono **System** e **String**) viene importata automaticamente.

L'uso delle costanti

Un programma per il cambio di valuta

```
public class Convert1{
    public static void main(String[] args){
    double dollars = 2.35;
        double euro = dollars * 0.72;
    }
}
```

Cosa rappresenta il **numero magico** 0.72 usato per convertire i dollari in euro...

Come è possibile definire le variabili, è opportuno definire **nomi simbolici** anche alle **costanti** utilizzate nei programmi.

```
public class Convert2{
    public static void main(String[] args){
        final double EURO_PER_DOLLAR = 0.72;
        double dollars = 2.35;
        double euro = dollars * EURO_PER_DOLLAR;
        double dollars2 = 3.45;
        double euro2 = dollars2 * EURO_PER_DOLLAR;
    }
}
```

Due vantaggi principali aumento della leggibilità; se il valore della costante deve cambiare il valore cambia in un solo punto.

Definizione di costante

Sintassi

```
final nomeTipo NOME_COSTANTE = espressione;
```

Scopo

definire la costante **NOME_COSTANTE** di tipo **nomeTipo**
assegna il valore di **espressione**, che non potrà più essere modificato

Operazioni aritmetiche

- L'operatore di**moltiplicazione** va sempre indicato **esplicitamente**, non può essere **sottinteso**.
- Le operazioni di moltiplicazione e divisione **hanno la precedenza** sulle operazioni di addizione e sottrazione, cioè vengono eseguite prima.
- È possibile usare **coppie di parentesi tonde** per indicare in quale oridne valutare le sotto-espressioni.

Divisione tra interi

- Quando entrambi gli operandi sono numeri **interi**, la **divisione** calcola il quoziente intero, scartando il resto
- Per avere il resto della divisione tra numeri interi è possibile utilizzare l'operatore %.

Combinare assegnazioni e aritmetica

Abbiamo già visto come in Java sia possibile combinare in un unico enunciato un'assegnazione ed un'espressione aritmetica che coinvolge la variabile a cui si assegnerà

```
totalEuro = totalEuro + dollars*0.72
```

L'espressione di sopra è tanto comune che Java mette a disposizione delle **scorciatoie**. Infatti si può tradurre come:

```
totalEuro += dollars*0.72
```

che esiste per tutti gli operatori aritmetici

```
x = x * 2 ------> x *= 2
```

Incremento di una variabile

Incremento

È l'operazione che consiste di aumentare di uno il valore di una variabile

```
int counter = 0;
counter = counter + 1;
```

Scorciatoie

Anche per questo tipo di operazione Java mette a disposizione delle scorciatoie e precisamente fornisce un operatore chiamato **incremento/decremento**

```
counter++;
counter--;
```