

3 - Variabili

Programmazione e analisi di dati
Modulo A: Programmazione in Java

Paolo Milazzo

Dipartimento di Informatica, Università di Pisa
<http://pages.di.unipi.it/milazzo>
milazzo@di.unipi.it

Corso di Laurea Magistrale in Informatica Umanistica
A.A. 2019/2020

Espressioni aritmetiche (1)

Nel programma HelloWorld abbiamo usato il comando `System.out.println()` per stampare una **stringa**

```
public class HelloWorld {  
    public static void main(String[] args) {  
        //visualizza un messaggio di saluto  
        System.out.println("Hello World!");  
    }  
}
```

Possiamo in realtà stampare tanti altri tipi di valori

- Ad esempio, possiamo stampare il risultato di una **espressione**

Espressioni aritmetiche (2)

Consideriamo il seguente programma che calcola l'area di un triangolo di base 5 cm e altezza 10 cm

```
public class AreaTriangolo {  
    public static void main(String[] args) {  
        System.out.println(5*10/2);  
    }  
}
```

Il programma risolve l'espressione $5 \cdot 10 / 2$ e stampa il risultato a video

- Nota: * rappresenta l'operazione di moltiplicazione

Espressioni aritmetiche (3)

Le espressioni aritmetiche più semplici sono costituite da singoli **letterali**

- **Letterali interi:** 3425, 12, -34, 0, -4, 34, -1234,
- **Letterali frazionari:** 3.4, 5.2, -0.1, 0.0, -12.45, 1235.3423,

Espressioni più complesse si ottengono utilizzando **operatori aritmetici**

- moltiplicazione * divisione /
- modulo % (resto della divisione tra interi)
- addizione + sottrazione -

Le operazioni sono elencate in ordine decrescente di **priorità**

- ossia $3+2*5$ fa 13, non 25

Le **parentesi tonde** cambiano l'ordine di valutazione degli operatori

- ossia $(3+2)*5$ fa 25

Inoltre, tutti gli operatori sono **associativi a sinistra**

- ossia $3+2+5$ corrisponde a $(3+2)+5$
- quindi $18/6/3$ fa 1, non 9

Espressioni aritmetiche (4)

L'operazione di **divisione** / si comporta diversamente a seconda che sia applicato a letterali interi o frazionari

- $25/2 = 12$ (divisione intera)
- $25\%2 = 1$ (resto della divisione intera)
- $25.0/2.0 = 12.5$ (divisione reale)
- $25.0\%2.0 = 1.0$ (resto della divisione intera)

Una operazione tra un letterale **intero** e un **frazionario** viene eseguita come tra **due frazionari**

- $25/2.0 = 12.5$
- $1.5 + (25/2) = 13.5$ (**attenzione** all'ordine di esecuzione delle operazioni)
- $2 + (25.0/2.0) = 14.5$

Variabili (1)

Modifichiamo il programma che calcola l'area di un triangolo introducendo un po' di **variabili**

```
public class AreaTriangolo2 {  
    public static void main(String[] args) {  
        int base, altezza;  
        int area;  
  
        base=5;  
        altezza=10;  
  
        area=base*altezza/2;  
  
        System.out.println(area);  
    }  
}
```

Il programma così risulta essere più chiaro:

- Si capisce meglio quali siano la base e l'altezza del triangolo
- Si capisce meglio che cosa calcola il programma

Variabili (2)

Una **variabile** è una casella di memoria identificata da un **nome**

- Il suo scopo è di contenere un valore di un certo **tipo**
- Serve per memorizzare dati durante l'esecuzione di un programma
- Il nome di una variabile (come il nome di una classe o di un metodo) è un **identificatore**
 - ▶ può essere costituito da lettere, numeri e underscore (esempi: `num1`, `i`, `risultato`, `costo_tot`, ...)
 - ▶ non deve coincidere con una parola chiave del linguaggio (esempi "sbagliati": `class`, `static`, `int`, ...)
 - ▶ è bene che sia significativo per il programma (meglio `base`, `altezza` e `area` di `X1`, `X2` e `X3`)

Variabili (3)

In Java ogni variabile deve essere **dichiarata** prima del suo uso

- Nella dichiarazione di una variabile se ne specifica il **nome** e il **tipo** (o meglio, il tipo dei valori che può contenere)

Nell'esempio, abbiamo dichiarato tre variabili con nomi base, altezza e area, tutte di tipo `int` (numeri interi)

```
int base, altezza;  
int area;
```

ATTENZIONE! Ogni variabile deve essere dichiarata **UNA SOLA VOLTA** (la prima volta che compare nel programma)

Assegnamento

Si può memorizzare un valore in una variabile tramite l'operazione di **assegnamento**

Il valore da assegnare a una variabile può essere un **letterale** o il risultato della valutazione di un'**espressione**

Esempi di assegnamenti:

```
base=5;  
altezza=10;  
area=base*altezza/2;
```

Dall'ultimo esempio si può notare che le variabili possono essere usate anche all'**interno di espressioni**

- I valori di `base` e `altezza` vengono **letti** e usati nell'espressione
- Il risultato dell'espressione viene **scritto** nella variabile `area`

Dichiarazione + Assegnamento

Prima di poter essere usata (letta) in un'espressione una variabile deve:

- essere stata dichiarata
- essere stata assegnata almeno una volta (**inizializzata**)

Si può combinare dichiarazione e assegnamento (**dichiarazione con inizializzazione**). Ad esempio:

```
int base=5;
int altezza=10;
int area=base*altezza/2
```

Esempio: Calcolo interessi (1)

```
public class Interessi {  
  
    public static void main(String[] args) {  
  
        double capitale=100000;  
        double tasso=0.05;  
        double interesse;  
  
        //calcola l'interesse maturato dopo un anno  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        //calcola l'interesse maturato dopo due anni  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        //calcola l'interesse maturato dopo tre anni  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        // stampa il risultato  
        System.out.print("Capitale maturato: ");  
        System.out.println(capitale);  
  
    }  
}
```

Esempio: Calcolo interessi (2)

Risultato dell'esecuzione:

```
Capitale maturato: 115762.5
```

In questo esempio:

- Alcune variabili sono inizializzate nella dichiarazione
- Le variabili hanno tipo `double` (virgola mobile a precisione doppia) e le operazioni su tali variabili sono come tra letterali frazionari (divisione reale)
- Le variabili **cambiano valore** durante l'esecuzione
- Il nuovo valore di una variabile può essere calcolato a partire dal precedente (`capitale=capitale+interesse`)
- Il metodo `print()` visualizza un messaggio senza andare a capo

Costanti (1)

Nell'esempio Interessi la variabile tasso non cambia mai

Nella dichiarazione delle **variabili che NON DEVONO mai cambiare** valore si può utilizzare il modificatore `final`

```
final double tasso=0.05;
```

Il modificatore `final` trasforma la variabile in una **costante**

Il **compilatore** si occuperà di controllare che il valore delle costanti non venga mai modificato (ri-assegnato) dopo essere stato inizializzato.

Aggiungere il modificatore `final` **non cambia funzionamento programma...** ma serve a **prevenire errori** di programmazione

- Si chiede al compilatore di controllare che una variabile non venga ri-assegnata per sbaglio.
- Sapendo che una variabile non cambierà mai valore, il compilatore può anche eseguire delle **ottimizzazioni** sull'uso di tale variabile...

Costanti (2)

```
public class Interessi2 {  
    public static void main(String[] args) {  
        double capitale=100000;  
        final double tasso=0.05;  
        double interesse;  
  
        //calcola l'interesse maturato dopo un anno  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        //calcola l'interesse maturato dopo due anni  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        //calcola l'interesse maturato dopo tre anni  
        interesse=capitale*tasso;  
        capitale=capitale+interesse;  
  
        // stampa il risultato  
        System.out.print("Capitale maturato: ");  
        System.out.println(capitale);  
    }  
}
```

Input dall'utente (1)

Per ricevere valori in input dall'utente si usa la classe Scanner, che fa parte della **Libreria Standard** di Java (detta **Java API**)

```
import java.util.Scanner; // richiama la classe Scanner

public class Somma {

    public static void main(String[] args) {

        // predispose il programma per la lettura dell'input
        Scanner input = new Scanner(System.in);

        System.out.println("Inserisci due numeri interi:");

        int n1, n2;

        // attende l'inserimento di due numeri e li assegna
        n1 = input.nextInt();
        n2 = input.nextInt();

        // calcola e stampa la somma
        System.out.print("Somma: ");
        System.out.println(n1+n2);

    }
}
```

Input dall'utente (2)

La classe `Scanner` deve essere richiamata usando la direttiva `import` prima dell'inizio della classe

```
import java.util.Scanner;
```

L'oggetto `input` (che può essere chiamato anche diversamente) deve essere predisposto (creato) prima di iniziare a chiedere i valori all'utente

```
Scanner input = new Scanner(System.in);
```

La lettura vera e propria dei valori in `input` si fa tramite metodi diversi a seconda del tipo dei valori attesi

```
x = input.nextInt(); // legge un numero intero  
y = input.nextDouble(); // legge un numero frazionario
```