Impariamo a programmare in Java™

Lezione 4

Gr.it

Variabili
e
tipi di dati numerici

Un programma che elabora numeri

Proviamo a **creare un programma** che dato un **valore in lire**, e uno in **euro**, dopo aver **convertito quello in lire** li sommi e stampi a video il risultato.

Lire: 15000

Euro: 2,35

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
   System.out.print("Valore totale in Euro ");
   System.out.println(totalEuro);
   }
}
```

Un programma che elabora numeri

Questo programma elabora due tipi di numeri

- ***Numeri interi*** per le lire italiane che non prevedono l'uso di decimi e centesimi e quindi non hanno bisogno di una parte frazionaria.
- ***Numeri frazionari*** ("in virgola mobile"***)*** per gli euro che prevedono l'uso dei decimi e dei centesimi e assumono valori con il separatore decimali.

I numeri interi (positivi e negativi) si rappresentano in Java con il tipo di dati int

I numeri in virgola mobile (positivi e negativi, ***a precisione doppia***) si rappresentano in Java con il tipo di dati double

Un programma che elabora numeri

In realtà sarebbe possibile usare numeri in virgola mobile anche per rappresentare i numeri interi, ma ecco due buoni motivi per non farlo:

- **"filosofica"**: indicando esplicitamente che per le lire italiane usiamo un numero intero, rendiamo ***evidente*** il fatto che non esistono i decimali per le lire italiane.
    - è importante rendere comprensibile i programmi.
- **"pratica"**: i numeri interi rappresentati come tipo di dati **int** sono più ***efficienti***, perché ***occupano meno spazio in memoria*** e sono ***elaborati più velocemente.***

I commenti

Nel programma sono presenti anche dei ***commenti***, che vengono ***ignorati*** dal compilatore, ma che rendono il programma molto più comprensibile.

Un commento ***inizia con una doppia barra // e termina alla fine della riga***

Nel commento si può scrivere *qualsiasi cosa.*

```
// Lire italiane
```

Se il commento si deve estendere ***per più righe***, è molto scomodo usare tante volte la sequenza //

Si può iniziare un commento con /* e terminarlo con */

```
// questo è un commento
// lungo, inutile …..
// … e anche scomodo
```

```
/*
questo è un commento
lungo ma inutile …..
*/
```

Alcune note sintattiche

L'operatore che indica la divisione è /, quello che indica la moltiplicazione è *

```
lit / 1936.27     oppure     euro * 1936.27
```

Quando si scrivono numeri in virgola mobile, bisogna usare il punto come separatore decimale, invece della virgola (uso anglosassone)

```
1936.27
```

Quando si scrivono i numeri, non bisogna indicare il punto separatore delle migliaia

```
15000
```

I numeri in virgola mobile si possono anche esprimere in notazione esponenziale

```
1.9E3 // vale 1.93 x 10^3
```

L’uso delle variabili

Il programma fa uso di variabili di tipo numerico

lit di tipo int, euro e totalEuro di tipo double

Le ***variabili*** sono spazi di memoria, identificati da un ***nome***, che possono conservare ***valori*** di un ***determinato tipo***.

Ciascuna variabile deve essere ***definita*** indicandone il tipo ed il nome.

```
int lit
```

Una variabile può contenere soltanto valori del suo ***stesso tipo***

Nella ***definizione di una variabile*** è possibile assegnarle un valore iniziale

```
int lit = 15000
```

L'uso delle variabili

Il programma poteva risolvere lo stesso problema anche senza fare uso 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 stato molto meno comprensibile e modificabile con difficoltà

Attenzione:
Questo visualizza il risultato dell'espressione

I nomi delle variabili

La scelta dei nomi per le variabili è molto importante, ed è bene ***scegliere nomi che descrivano adeguatamente la funzione della variabile***

In Java, un nome (di variabile, di metodo, di classe…) può essere composto da ***lettere***, da ***numeri*** e dal ***carattere di sottolineatura*** (_), ma:

- *deve iniziare con una lettera*
- *non può essere una parola chiave del linguaggio*
- *non può contenere spazi*

Le lettere ***maiuscole*** sono diverse dalle ***minuscole***! Ma è buona norma non usare nello stesso programma nomi di variabili che differiscano soltanto per una lettera

Definizioni di variabili

Sintassi:

```
nomeTipo nomeVariabile
```

```
nomeTipo nomeVariabile = espressione
```

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

Di solito in java si usano le seguenti convenzioni

- i nomi di variabili e di metodi iniziano con una lettera minuscola

  lit main

- i nomi di classi iniziano con la lettera maiuscola Coins1
- i nomi composti, in entrambi i casi si ottengono attaccando le parole successive alla prima con la maiuscola

L’uso delle variabili

Abbiamo visto come i programmi usino le variabili per memorizzare i valori da elaborare ed i risultati dell’elaborazione

Le ***variabili*** sono posizioni di memoria che possono conservare ***valori*** di un ***determinato tipo***

Il valore memorizzato in una variabile può essere ***modificato***, non soltanto ***inizializzato***…

Il cambiamento del valore di una variabile si ottiene con un ***enunciato di assegnazione***

L’uso delle variabili

```
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 * 0.93;
    // stampa il valore totale
    System.out.print("Valore totale in Euro ");
    System.out.println(totalEuro);
   }
}
```

L'uso delle variabili

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

Per prima cosa la variabile viene inizializzata contestualmente alla sua definizione

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

Poi la variabile viene incrementata due volte

```
totalEuro = totalEuro + euro;
totalEuro= totalEuro + dollars * 0.93;
```

mediante *enunciati di assegnazione*

L'assegnazione

MOLTO IMPORTANTE

Analizziamo l'enunciato di assegnazione

```
totalEuro = totalEuro + euro;
```

Cosa significa? ***Non*** certo che la variabile **totalEuro** ***è uguale*** a se stessa più qualcos'altro…

L'enunciato di assegnazione significa

> *Calcola* il valore dell'espressione a *destra* del segno = e scrivi il risultato nella posizione di memoria assegnata alla variabile indicata a *sinistra* del segno =

Stringhe

Il tipo di dati “Stringa”

I tipi di dati più importanti nella maggior parte dei programmi sono i numeri e le stringhe

Una stringa è una sequenza di caratteri che in java (come in molti altri linguaggi) vanno racchiusi tra virgolette

- le virgolette non fanno parte della stringa “Hello”

Possiamo dichiarare e inizializzare variabili di tipo stringa

```
String name = “John”;
```

Possiamo assegnare un valore ad una variabile di tipo stringa

```
name = “Michael”;
```

Il tipo di dati “Stringa”

Il metodo lenght della classe String non è un metodo statico

- infatti per invocarlo usiamo un oggetto della classe String, e non il nome della classe stessa.

```
// NON FUNZIONA!
String s = “John” ;
int n = String.lenght(s) ;
```

```
//  FUNZIONA!
String s = “John” ;
int n = s.lenght() ;
```

Una stringa di lunghezza zero, che non contiene caratteri, si chiama stringa vuota e si indica con due caratteri virgolette consecutivi, senza spazi interposti.

```
String empty = “” ;
System.out.println(empty.lenght());
```

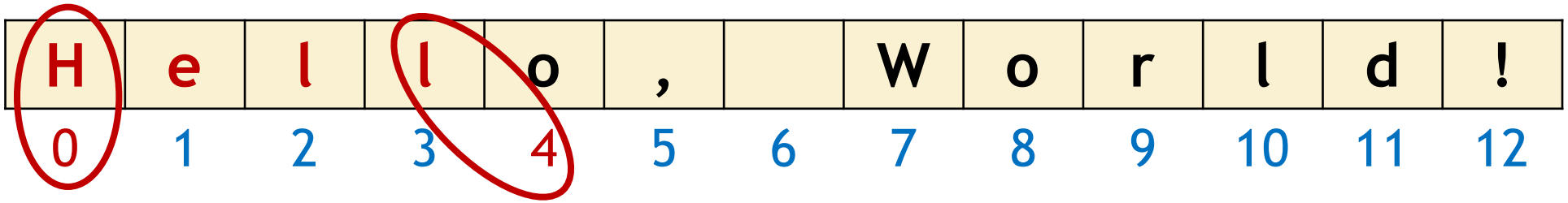
0

Estrazione di sottostringhe

Per estrarre una sottostringa da una stringa si usa il metodo substring

```
String greeting = “Hello, World” ;
String sub = greeting.substring(0 , 4);
//sub contiene “Hell”
```

- il ***primo*** parametro di substring è la ***posizione del primo carattere*** che si vuole estrarre
- il ***secondo*** parametro è la ***posizione successiva all’ultimo carattere*** che si vuole estrarre

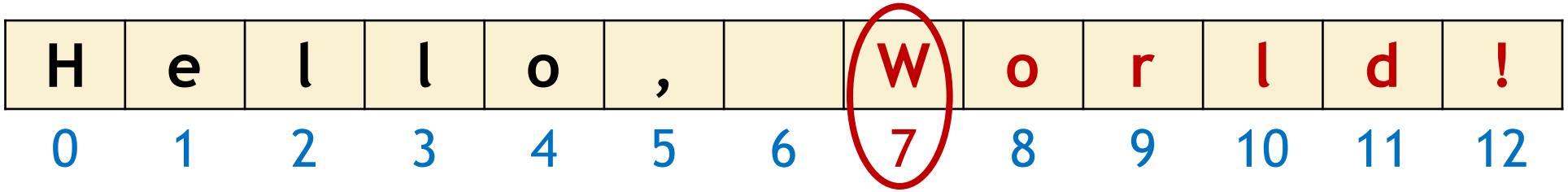


Estrazione di sottostringhe

Per estrarre una sottostringa da una stringa si usa il metodo substring

```
String greeting = “Hello, World” ;
String sub = greeting.substring(7);
//sub contiene “World!”
```

- In caso ci si un solo parametro questo indicherà la ***posizione del primo carattere*** da cui parte l’estrazione.



Estrazione di sottostringhe

Cosa succede se si fornisce un ***parametro errato*** a **substring**?

```
// NON FUNZIONA
String greeting = “Hello, World” ;
String sub = greeting.substring(0 , 14);
```

Il programma viene compilato correttamente, ma viene generato un errore in esecuzione

```
Exception in thread “main”
Java.lang.StringIndexOutOfBoundsException
String index out of range: 14
```

Alcuni metodi utili di String

Un problema che capita spesso di affrontare è quello della conversione di una stringa per ottenerne un'altra tutta in maiuscolo o tutta in minuscolo

La classe **String** mette a disposizione due metodi:

- toUpperCase *converte tutto in maiuscolo*
- toLowerCase *converte tutto in minuscolo*

```
String s = "Hello";
String ss = s.toUpperCase() + s.toLowerCase();
// ss vale "HELLOhello"
```

Si noti che l'applicazione di uno di questi metodi alla stringa *s* ***non altera il contenuto della stringa s, ma restituisce una nuova stringa.***

In particolare, ***nessun metodo della classe String modifica l'oggetto con cui viene invocato!***

Esempio

Scriviamo un programma che genera password per un utente, con la regola seguente:

Si prendono le iniziali dell'utente, le si rendono minuscole e si concatena l'età dell'utente espressa numericamente

Utente: Andrea Greguoldo

Età: 30

Password: ag30

(Attenzione questa regola non è assolutamente da usare, prevedibile e poco sicura)