Impariamo a programmare in Java™

Lezione 7

Gr.it

Ancora Stringhe

Sequenze di “escape”

Proviamo a stampare una stringa che contiene delle virgolette

```
Hello, “World”!
```

```
// NON FUNZIONA!
System.out.println("Hello, "World"!");
```

Il compilatore identifica le seconde virgolette come la fine della prima stringa "Hello, ", ma poi non capisce il significato della parola World.

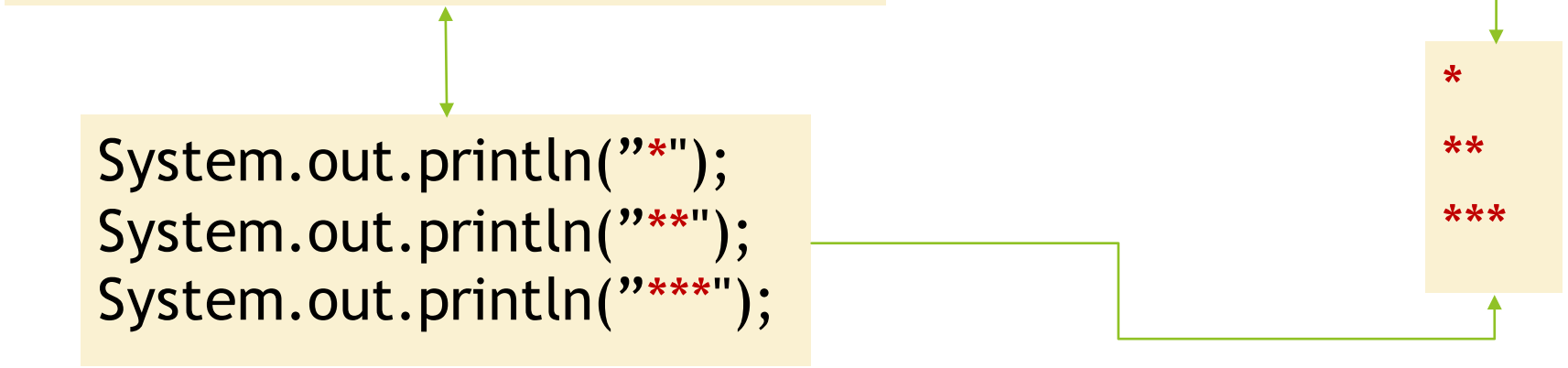
Basta inserire una barra rovesciata \ (backslash) prima delle virgolette all’interno della stringa

```
FUNZIONA!
System.out.println("Hello, \"World\"!");
```

Altra sequenze di “escape”

Un’altra sequenza di escape che si usa \n, che rappresenta il carattere di “nuova riga” o “andare a capo”.

```
System.out.println("*\n**\n***");
```



Le sequenze di escape si usano anche per inserire caratteri di lingue straniere o simboli che non si trovano sulla tastiera.

Tipo di dati *char*

Caratteri in una stringa

Sappiamo già come estrarre sottostringhe da una stringa, con il metodo **substring**.

A volte necessario estrarre ed elaborare ***sottostringhe*** di dimensioni minime, ciò ***di lunghezza unitaria***

- *Una stringa di lunghezza unitaria contiene **un solo carattere, che può essere memorizzato in una variabile***
- *di tipo char anziché in una stringa il tipo char in Java un tipo di dato fondamentale come i tipi di dati numerici ed il tipo **boolean**, cioè non una classe*

Caratteri in una stringa

La presenza del tipo di dati char non strettamente necessaria in Java (ed anche per questo motivo che non l'avevamo ancora studiato)

- *infatti, ogni elaborazione che può essere fatta su variabili di tipo char potrebbe essere fatta su stringhe di lunghezza unitaria*

L'uso del tipo char per memorizzare stringhe di lunghezza unitaria è però importante, perché:

- *una variabile di tipo char occupa meno spazio in memoria di una stringa di lunghezza unitaria*
- *le elaborazioni su variabili di tipo char sono più veloci*

Caratteri in una stringa

Il metodo charAt della classe **String** restituisce il singolo carattere che si trova nella posizione indicata dal parametro ricevuto

*la convenzione sulla numerazione delle posizioni in una stringa la stessa usata dal metodo **substring***

```
String s = "John";
char ch = s.charAt(2); // ch contiene 'h'
```

Una struttura di controllo che si usa spesso l'elaborazione di tutti i caratteri di una stringa

```
String s = "John";
for (int i = 0; i < s.length(); i++)
{ char ch = s.charAt(i);
// elabora ch
}
```

Esercizio

Data una stringa inserita da un utente. Stampare a video la stringa invertita.

Problema

Scrivere un programma che legge dallo standard input una sequenza di dieci numeri, uno per riga. Che chieda all'utente un numero intero index e visualizza il numero che nella sequenza occupava la posizione indicata da index.

Occorre memorizzare tutti i valori della sequenza Potremmo usare dieci variabili diverse per memorizzare i valori, selezionati poi con una lunga sequenza di alternative, ma se i valori dovessero essere mille?

Memorizzare una serie di valori

Lo strumento messo a disposizione dal linguaggio Java per memorizzare una sequenza di dati si chiama array *(che significa "sequenza ordinata")*

- la struttura array esiste in quasi tutti i linguaggi di programmazione
- Un array in Java è un oggetto che realizza una raccolta di dati che siano tutti dello stesso tipo.
- Potremo avere quindi array di numeri interi, array di numeri in virgola mobile, array di stringhe, array di conti bancari...

Costruire un array

Come ogni oggetto, un array deve essere costruito con l'operatore new, dichiarando il tipo di dati che potrà contenere.

```
new double [10];
```

Il tipo di dati di un array può essere qualsiasi tipo di dati valido in Java *(uno dei tipi di dati fondamentali o una classe)*. Nella costruzione deve essere seguito da una coppia di parentesi quadre che contiene la dimensione dell'array, cioè il numero di elementi che potrà contenere.

Riferimento ad un array

Come succede con la costruzione di ogni oggetto, l'operatore new restituisce un riferimento all'array appena creato, che può essere memorizzato in una variabile oggetto dello stesso tipo

```
double[ ] values = new double [10];
```

Attenzione: nella definizione della variabile oggetto devono essere presenti le parentesi quadre, ma non deve essere indicata la dimensione dell'array; la variabile potrà riferirsi solo ad array di quel tipo, ma di qualunque dimensione

```
//si può fare in due passi
double[ ] values;
values = new double [10];
```

Utilizzare un array

```
double[ ] values = new double [10];
double oneValues = values [3];
values[5] = 3.4;
```

Il numero utilizzato per accedere ad un particolare elemento dell'array si chiama *indice*

L'*indice* può assumere un valore compreso tra ***0 (incluso) e la dimensione dell'array (esclusa)***, cioè segue le stesse convenzioni viste per le posizioni dei caratteri in una stringa

il primo elemento ha ***indice 0***

l'ultimo elemento ha ***indice (dimensione -1)***

La dimensione di un array

Per sapere quanto grande è un array si può utilizzare la sua variabile pubblica di esemplare length (attenzione, non un metodo!)

```
double[ ] values = new double [10];
int a = values.length;  //a vale 10
```