Programmazione I

A.A. 2002-03

STRINGHE

(Lezione XXIII)

Dichiarazione e metodi

Prof. Giovanni Gallo
Dr. Gianluca Cincotti
Dipartimento di Matematica e Informatica
Università di Catania
e-mail : {gallo, cincotti}@dmi.unict.it

Stringhe di caratteri

- La libreria standard di Java mette a disposizione due classi per gestire stringhe di caratteri
  - La classe **`String`**
  - La classe **`StringBuffer`**

La classe String

- La classe **`String`** definisce *oggetti* (cioè non si tratta di un tipo di dato primitivo) che rappresentano stringhe di caratteri.
  - Tutte le stringhe letterali nei programmi Java sono implementate come oggetti di questa classe.
    - "abc", "defghi"
- Le stringhe sono *costanti.*
  - Il loro valore *non* può essere modificato dopo che è stato generato.

La dichiarazione di una stringa

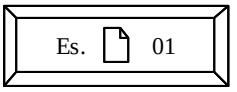
Sintassi per la dichiarazione di un'istanza della classe stringa:

String <nomevariabile> = "<stringa>" ;

```
String  str = "Ecco la stringa !";
```

L'operatore di concatenazione

```
String catena = "catena";
System.out.println
        ("con" + catena + "zione");
```



L'operatore di assegnazione

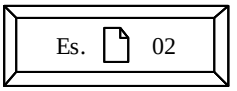
- Concatenazione lecita !
  - **`parola = "buona";`**
  - **`parola += "sera";`**
- Ma gli oggetti della classe **`String`** non sono costanti ?
  - Infatti, ad ogni concatenazione una nuova stringa viene creata con un'implicita chiamata del costruttore **`new String`**

Istanziare un oggetto `String`

La classe **`String`** fornisce inoltre differenti costruttori per inizializzare le stringhe.

```
str1 = “abc”
str2 = new String (“abc”)

char data[] = {'a', 'b', 'c'};
String str3 = new String(data);
```



Come confrontare due stringhe

Attenzione: non usare assolutamente l’operatore di confronto

==

Questo infatti è l’operatore di eguaglianza tra NUMERI e non tra *oggetti* String

Usare il metodo “equals” della classe String !

Sintassi: s.equals(t) \\confronta le stringhe s e t.

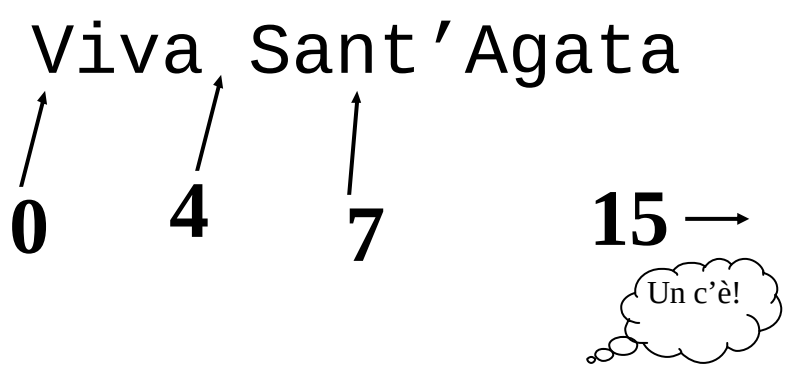
Metodi della classe `String`

- Dichiarare una variabile stringa
- Concatenare stringhe
- Estrarre sottostringhe
- Modificare stringhe
- Controllare se una stringa termina o inizia con un dato carattere
- Prendere in input una stringa e restituirne una nuova avendo eliminato tutti gli spazi iniziali e finali
- ….e molto altro

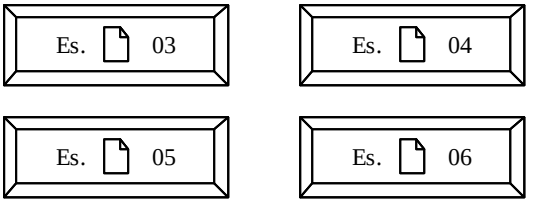
Metodi della classe `String` *(cont.)*

- Sono disponibili numerosi metodi per manipolare le stringhe.
  - **`charAt`** `(int index)`
  - **`compareTo`** `(String str)`
  - **`compareToIgnoreCase`** `(String str)`
  - **`concat`** `(String str)`
  - **`equals`** `(String str)`
  - **`equalsIgnoreCase`** `(String str)`
  - **`length`** `()`
  - **`substring`** `(int,int)`
  - **`toString`** `()`
  - **`toUpperCase`** `()` e **`toLowerCase`** `()`
  - **`toCharArray`** `()`

Attenzione alla numerazione dei caratteri: inizia da ZERO



Esempi



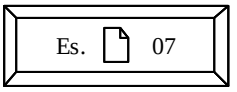
Estrazione di una sottostringa

String <Destinazione> = <Sorgente>.substring (

<posizione del carattere da cui partire per tagliare la stringa>,

<posizione del primo carattere che si intende escludere>);

Esiste anche la versione overloaded del metodo con un solo parametro !



La classe `StringBuffer`

- Al contrario della classe **`String`**, la classe **`StringBuffer`** fornisce stringhe che possono essere modificate nel tempo.
  - Si usa questo tipo di dato quando si sa che una stringa viene manipolata e cambiata.
- Questa classe *non* verrà presa in considerazione in questo corso !

Esercizio

- Solo per i più bravi !
  - Scrivere una funzione che presa in input una stringa, fornisca tutti i possibili anagrammi della stringa (ovvero, tutte le possibili permutazioni dei caratteri contenuti nella stringa).
    - È conveniente usare la ricorsione !

Fine