

STRINGHE IN JAVA

- In Java, le stringhe non sono “pezzi di memoria” con dentro dei caratteri, come in C: sono *oggetti appartenenti alla classe String*
- Una stringa Java rappresenta *uno specifico valore* e come tale è *immodificabile*: una `String` **non è un contenitore!**
 - se occorre un contenitore esiste `StringBuffer`
- L'operatore `+` denota la concatenazione:

`"ciao" + " mondo\n"`

NB: la concatenazione è fatta a *tempo di compilazione*, quindi *non introduce inefficienze*

COSTANTI String

- Le costanti `String` possono essere denotate nel modo usuale:

`"ciao"`

`"mondo\n"`

- Quando si scrive una costante `String` tra virgolette, viene creato *implicitamente un nuovo oggetto di classe String*, inizializzato a tale valore.
- Una costante `String` **non può eccedere la riga**: quindi, dovendo scrivere stringhe più lunghe, conviene *spezzarle e concatenarle con `+`*.

STRINGHE IN JAVA

- Le stringhe Java *non sono array di caratteri*
- Sono oggetti *concettualmente diversi*
 - *non si applicano gli operatori degli array!*
 - in particolare, non si applica il classico `[]`
- Per *selezionare il carattere i-esimo* si usa il metodo `charAt()`:

```
String s = "Nel mezzo del cammin";
char ch = s.charAt(4);
```
- La classe `String` definisce *decine di metodi*: si veda la documentazione (Java 2 API)

STRINGHE IN JAVA

- Tutte le classi Java definiscono un metodo `toString()` che produce una `String` a partire da un oggetto della classe: *ciò consente di “stampare” facilmente qualunque oggetto di qualunque classe*
- È responsabilità del progettista definire un metodo `toString()` che produca una stringa “significativa”
- Quello di default stampa un identificativo alfanumerico dell'oggetto.

ESEMPIO

```
public class Esempio5 {
    public static void main(String args[]) {
        String s = "Nel mezzo del cammin";
        char ch = s.charAt(4);
        System.out.println(ch);
        System.out.println("Carattere: " + ch);

        Counter c = new Counter(10);
        System.out.println(c);
    }
}
```

Converte ch in stringa e lo concatena alla frase.

Usa il metodo `toString()` predefinito di `Counter` → Stampa un identificativo dell'oggetto c.

VARIANTE

- La stampa di poco fa non vi piace?
- Potete ridefinire esplicitamente il metodo `toString()` della classe `Counter`, *facendogli stampare ciò che preferite*

Ad esempio:

```
public class Counter {
    ...
    public String toString() {
        return "Counter di valore " + val;
    }
}
```

VARIANTE

Lo stesso identico esempio:

```
public class Esempio5 {
    public static void main(String args[]) {
        String s = "Nel mezzo del cammin";
        char ch = s.charAt(4);
        System.out.println(ch);
        System.out.println("Carattere: " + ch);
        Counter c = new Counter(10);
        System.out.println(c);
    }
}
```

ora stamperà:



ARRAY IN JAVA

- Gli array Java sono *oggetti*, istanze di una *classe speciale* denotata da []
- Quindi, prima si definisce un riferimento...

```
int[] v;           int v[];
Counter[] w;       Counter w[];
```

- ...e poi si crea dinamicamente l'oggetto:

```
v = new int[3];
w = new Counter[8];
```

ARRAY IN JAVA

- Gli array Java sono *oggetti*, istanze di una

È un riferimento, quindi *non deve specificare alcuna dimensione!*

```
int[] v;           int v[];
Counter[] w;       Counter w[];
```

- La posizione delle [] è a scelta: o dopo il nome, come in C, oppure di seguito al tipo (*novità*)

```
w = new Counter[8];
```

La dimensione *si specifica all'atto della creazione*

ARRAY IN JAVA

Attenzione!! Ogni elemento dell'array:

- è una variabile, se gli elementi dell'array sono di un *tipo primitivo* (int, float, char, ...)

```
v = new int[3];
```



- è un riferimento a un (futuro) oggetto, se gli elementi dell'array sono (*riferimenti a*) *oggetti*

```
w = new Counter[3];
```



ARRAY IN JAVA

Quindi:

- nel primo caso, ogni elemento dell'array è una normale variabile, "già usabile" così com'è:

```
v = new int[3];
v[0] = 1; v[1] = 34;
```

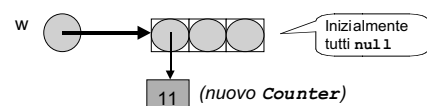


ARRAY IN JAVA

Quindi:

- nel secondo caso, invece, ogni elemento dell'array è *solo un riferimento*: se si vuole un nuovo oggetto, bisogna crearselo!

```
w = new Counter[3];
w[0] = new Counter(11);
```



ARRAY IN JAVA

- In quanto *istanze di una classe "array"*, gli array Java hanno alcune *proprietà*
- Tra queste, il campo-dati pubblico `length` dà la *lunghezza* (dimensione) dell'array:

`v.length` *vale 3*

Una volta creato, un array ha comunque *dimensione fissa*

- non può essere "allargato" a piacere
- per tali necessità esiste la classe `Vector`

ESERCIZIO

Stampare a video gli argomenti passati dalla linea di comando.

- Il main riceve un *array di String*

```
public static void main(String[] args){
    ...
}
```
- Non c'è `argc`, perché la dimensione dell'array è indicata dalla proprietà `length`
- Ogni elemento di `args` è un (riferimento a) `String`

ESERCIZIO

Quindi:

```
public class EsempioMain{
    public static void main(String[] args){
        if (args.length==0)
            System.out.println("Nessun argomento");
        else
            for (int i=0; i<args.length; i++)
                System.out.println("argomento " + i
                                   + ": " + args[i]);
    }
}
```

L'operatore `+` concatena `String` e anche valori di tipi primitivi (che vengono automaticamente convertiti in `String`)

ESERCIZIO

Quindi:

```
public class EsempioMain{
    else
        for (int i=0; i<args.length; i++)
            System.out.println("argomento " + i
                               + ": " + args[i]);
    }
}
```

In Java si possono definire variabili *in ogni punto del programma* (non solo all'inizio come in C): in particolare è possibile definire l'indice *i* dentro al ciclo `for`

A differenza del C, `args[0]` è il 1° argomento (non il nome del programma!)

ESERCIZIO

Esempio d'uso:

`C:> java EsempioMain 34 e 56 "aaa eee"`

Output:

