

**Università di degli Studi della Campania**  
**Luigi Vanvitelli**  
**Dipartimento di Ingegneria**

**Programmazione ad Oggetti**

*a.a. 2020-2021*

**Sintassi Java**

Docente: Prof. Massimo Ficco

E-mail: *massimo.ficco@unicampania.it*

1

1

## Commenti java



I commenti possono essere inseriti nel codice Java in vari modi:

- ***/\** commento su una o più righe *\*/***
- ***//* commento sulla singola riga**
- ***/\*\** commenti di documentazione *\*/***

Esiste la possibilità di generare documentazione in formato HTML da un codice sorgente Java con la funzionalità ***javadoc*** messa a disposizione dal Java Developer's Kit (JDK)



2

# Identificatori java



Gli identificatori possono iniziare con una:

- lettera
- underscore (\_)
- dollaro(\$)
- una sequenza di caratteri che può contenere dei numeri

Java fa distinzione tra maiuscole e minuscole (case sensitivity)

Le parole chiave sono identificatori che hanno un significato speciale e possono essere usate solo nel modo opportuno



# Parole chiave java



Parole chiave comuni tra i linguaggi Java e C++

break	case	catch	char
class	const	continue	default
do	double	else	float
for	goto	if	int
long	new	private	protected
public	return	short	static
switch	this	throw	try
void	while	false	true
volatile			

Parole chiave del linguaggio Java

abstract	boolean	byte
byvalue	extends	final
finally	implements	import
instanceof	interface	native
null	package	super
synchronized	throws	



# Parole chiave



Abstract	else	interface
boolean	extends	long
break	false <sub>1</sub>	native
byte	final	new
case	finally	null <sub>1</sub>
catch	float	package
char	for	private
class	goto <sub>2</sub>	protected
const <sub>2</sub>	if	public
continue	implements	return
default	import	short
do	instanceof	static
double	int	strictfp

1. parole riservate; 2. parole al momento non utilizzate



# Tipi e variabili



Un tipo di dato identifica un insieme di valori e di operazioni applicabili a tali valori

Le variabili sono contenitori di valori di un qualche tipo. Ogni variabile è un nome che indica una locazione di memoria usata per contenere un valore che può cambiare nel tempo

Le costanti sono simili alle variabili ma possono contenere un unico valore per tutta la durata della loro esistenza

Java è un linguaggio **fortemente tipizzato**:

**“è impossibile assegnare ad una variabile un valore che sia incompatibile col suo tipo”**



# Java: tipi e variabili



Il linguaggio Java utilizza quattro tipi di dati semplici:

- **integer**
- **floating point**
- **character**
- **Boolean**

Esempio: **int** x;  
x = 32 \* 54;

**final int** Posti=27; // Costante

Per utilizzare una variabile di un tipo predefinito occorre

- dichiararla
- inizializzarla



Programmazione ad Oggetti - Prof. Massimo Ficco

7

## Java: tipi di dati



<i>Interi</i>	
bit	tipo
8 (±)	<b>byte</b>
16 (±)	<b>short</b>
32 (±)	<b>int</b>
64 (±)	<b>long</b>

(\*) tipo **char** unicode a 16 bit  
compatibile con i numeri interi

<i>Virgola mobile (reali)</i>	
bit	tipo
32	<b>float</b>
64	<b>double</b>

<i>Tipo</i>	
<i>Carattere</i>	<b>char (*)</b> (16 bit)
<i>Logico</i>	<b>boolean</b>

Il tipo boolean ha due valori:

- true
- false



Programmazione ad Oggetti - Prof. Massimo Ficco

8

# Java: dichiarazioni



	<i>Esempi</i>
V a r i a b i l i	<b>int</b> n; <b>float</b> x; <b>char</b> c; <b>boolean</b> enunciato; <b>int</b> v[] = <b>new int</b> [3]; String parola;
Costanti	<b>final float</b> IVA = 0,20;
Librerie	<b>import</b> awt.*;



# Java: tipi e variabili



Le variabili in Java sono valide soltanto dal punto in cui vengono dichiarate fino alla fine del blocco di istruzioni che le racchiude  
Non è possibile dichiarare una variabile con lo stesso nome di un' altra contenuta in un blocco più esterno

*Esempio:*

```
class Scope {  
    public static void main (String args[]) {  
        int count = 100;  
        for ( int i=0; i<3; i++) {  
            System.out.println("Valore : " + i );  
            int count = 2; //Compile time error...  
        } // end for  
    } // end main  
} // end class
```



# Conversioni e casting



A volte è necessario o utile convertire un valore di un tipo nel corrispondente valore di un altro tipo

In Java le conversioni di tipo possono avvenire in due modi

- conversioni durante un assegnamento (**casting implicito**) che sono ammissibili solo per le cosiddette conversioni larghe
- **casting esplicito**: operatore java specificato da un tipo racchiuso tra parentesi che viene messo davanti al valore da convertire

Non è possibile effettuare il casting tra i tipi interi e booleani



# Conversioni e casting



<i>Implicita (automatica)</i>	<i>esempio</i>
<i>A un tipo più 'capiente' viene assegnato un tipo meno 'capiente'</i> <b>double ← float ← long ← int ← char ← short ← byte</b>	<b>int i = 24;</b> <b>long n = i;</b>

<i>Esplicita (casting)</i>	<i>esempio</i>
<i>Si indica di fronte alla variabile il nuovo tipo tra parentesi: (nuovo_tipo) variabile;</i>	<b>long n = 24;</b> <b>int i = (int) n;</b>



# Istruzioni di controllo flusso



- ♦ L'ordine in cui le istruzioni sono eseguite in un programma è detto **flusso di controllo** o di esecuzione
- ♦ Le istruzioni condizionali e i cicli permettono di controllare il flusso di esecuzione all'interno dei metodi
- ♦ In java appartengono a tale categoria le seguenti istruzioni:
  - ♦ **if, if-else**
  - ♦ **switch**
  - ♦ **for**
  - ♦ **while, do-while**



## Costrutto If



- ♦ L'istruzione **if** permette al programma di decidere se eseguire o meno una determinata istruzione
- ♦ L'istruzione **if-else** permette al programma di eseguire una istruzione se si verifica una certa condizione e un'altra in caso contrario

<b>if</b>	<b>if</b> (condizione) istruzione;
<b>if else</b>	<b>if</b> (condizione) istruzione; <b>else</b> istruzione;
<b>if else (blocco)</b>	<b>if</b> (condizione) { istruzione; istruzione; ...; } <b>else</b> {istruzione; istruzione; ...; }



## Esempio : if /else



Esempio:

```
int count = 5;
if ( count < 0 ) {
    System.out.println("Error: count value is negative");
}
else {
    System.out.println("Il valore di count è " + count );
}
```

n.b.: NON C'è pericolo di confondere == e =



## Switch



- ◆ Lo **switch** valuta una espressione per determinarne il valore e poi lo confronta con quello delle clausole **case**
- ◆ L'istruzione **break** usata alla fine di ogni case, serve per saltare alla fine dello switch stesso

<b>switch</b>	<pre>switch (i) // i variabile byte o short o int o char {     case 1: istruzione;         break;     case 2: { istruz; istruz; ...; }         break;     case 3: { istruz; istruz; ...; }         break;     default:         { istruz; istruz; ...; } }</pre>
---------------	---





# Cicli: for



- ♦ L'istruzione **for** è usata di solito quando un ciclo deve essere eseguito un numero prefissato di volte

<b>for</b>	<b>for</b> (int i=start; i<=stop; i++) { istruzione; istruzione; ...; }
	<b>for</b> (int i=start; i<=stop; i--) { istruzione; istruzione; ...; }

Nota: si possono usare le istruzioni **break** o **continue** per uscire dal ciclo o riprenderlo



# Cicli condizionati: do while



- ♦ Una istruzione **while** permette al programma di eseguire più volte un blocco di istruzioni valutando ogni volta una condizione booleana
- ♦ Con il **while** il ciclo **do** esegue il suo corpo finchè la condizione non diventa falsa

<b>do while</b>	<b>do</b> { istruzione; istruzione; ...; } <b>while</b> (condizione);
	<b>while</b> (condizione) { istruzione; istruzione; ...; }



# Operatori



## Operatori aritmetici



<i>Operatore</i>	<i>Simbolo</i>
Addizione	+
Sottrazione	-
Moltiplicazione	*
Divisione	/
Modulo	%
Incremento	++n; n++
Decremento	--n; n--

- L'operatore di incremento somma 1 a qualsiasi valore intero o in virgola mobile mentre quello di decremento sottrae



## Operatori: forme abbreviate



- Java offre vari operatori che permettono di abbreviare le espressioni di assegnazione

<i>Operazione</i>	<i>Forma abbreviata</i>	<i>equivale a</i>
Addizione	<b>a += b</b>	<b>a = a + b</b>
Sottrazione	<b>a -= b</b>	<b>a = a - b</b>
Moltiplicazione	<b>a *= b</b>	<b>a = a * b</b>
Divisione	<b>a /= b</b>	<b>a = a / b</b>



## Operatori di assegnazione



<i>Operatore</i>	<i>Simbolo</i>
Di assegnazione	<b>=</b>
Uguale	<b>==</b>
Diverso	<b>!=</b>
Maggiore	<b>&gt;</b>
Minore	<b>&lt;</b>
Non maggiore	<b>&lt;=</b>
Non minore	<b>&gt;=</b>



# Operatori logici



- Gli operatori logici restituiscono un valore booleano e sono spesso usati per costruire condizioni complesse

<i>Operatore</i>	<i>Simbolo</i>
Negazione	!
Congiunzione	&&
Disgiunzione Inclusiva	
Disgiunzione esclusiva	^



# Caratteri speciali



<i>Carattere</i>	<i>Descrizione</i>
\n	A capo (new line)
\t	Tabulazione
\b	Cancella a sinistra (backspace)
\r	Inizio riga (carriage return)
\f	Avanzamento pagina (form feed)
\\	Barra inversa
\"	Virgolette
\'	Apice



# Java: Oggetti



Le variabili non semplici (cioé strutturate) o di un tipo non predefinito sono considerate oggetti

Gli oggetti vanno:

- **dichiarati**
- **istanziati**
- **Inizializzati**

Per l'istanziazione si usa la parola chiave **new**

L'istanziazione produce l'effettiva allocazione in memoria dell'oggetto

Esempio:

***String s=new String("Hello word");***



## Esempio Stringa 1

**V:**

```
class TestStringa{
public static void main(String arg[])
{
    String a=new String("Nel mezzo del cammin");
    String b="di nostra vita";

    a=a+b;
    System.out.println(a);
    System.out.println(a.substring(3));
    System.out.println(a.length());
}
}
```



# Output

V:

Nel mezzo del cammin di nostra vita  
mezzo del cammin di nostra vita  
33



# Le Stringhe 1/3

V:

**In Java le stringhe sono oggetti appartenenti alla classe `String`** (è un tipo particolare in java)

Costant stringa vengono racchiusa tra "

```
String s = "ciao a tutti"; // viene creato implicitamente un  
                           oggetto di classe String
```

La variabile `s` è un riferimento ad un oggetto di classe `String` e  
corrisponde a:

```
String s = new String("ciao a tutti");
```

In sostanza è il compilatore che nel primo caso istanzierà l'oggetto stringa  
"ciao a tutti" in memoria



## Le Stringhe 2/3

V:

La concatenazione di due stringhe in realtà genera automaticamente una nuova istanza di String il cui contenuto è costituito dai caratteri della prima e della seconda messi insieme

```
String s = "ciao"+" a tutti"
```

- 1) Le due stringhe "ciao" e " a tutti" rappresentano due oggetti stringa in memoria.
- 2) Grazie all'operatore di concatenazione viene creato un terzo oggetto stringa che verrà puntato da s.
- 3) Le due stringhe "ciao" e " a tutti" Verranno cancellate dalla memoria.

**In Java le stringhe sono immutabili!**

Quando viene creata una stringa il suo valore non può più essere cambiato

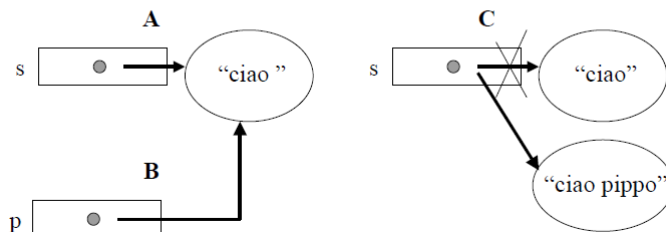


## Esempio

V:

```
String s = "ciao ";  
p=s;  
s= s + "pippo";
```

s==p????

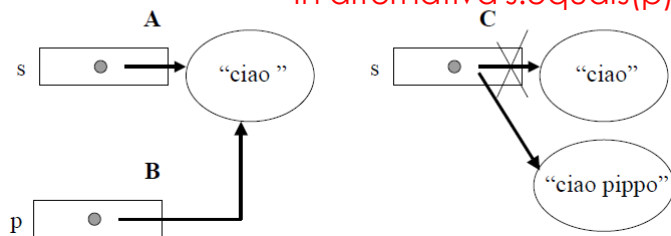


## Esempio

V:

```
String s = "ciao ";  
p=s;  
s = s + "pippo";
```

s==p????



## Le Stringhe 3/3

V:

```
String a=new String("Nel mezzo ");  
String b =new String("del cammin");  
String c =a+b;  
String a =a+b;
```

Le istanze puntate da a e b non vengono cancellate





## Esempio Stringa 2

V:

```
class TestStringa{  
    public static void main(String arg[])  
    {  
        String a=new String("Nel mezzo del cammin");  
        String e=new String("Nel mezzo del cammin");
```

Due stringhe puntate  
da due riferimenti  
differenti

**// b, c, d puntano fisicamente alla stessa stringa**

```
        String b="di nostra vita";  
        String c="di nostra vita";  
        String d="di nostra vita";  
  
        a=a+b;  
        b=a+b;  
        System.out.println(a);  
        System.out.println(a.substring(3));  
        System.out.println(a.length());
```

Il compilatore si accorge  
che esiste già la stringa "di  
nostra vita", per cui non  
istanzierà una nuova stringa  
ma semplicemente farà  
puntare c e d alla stessa  
stringa puntata da b

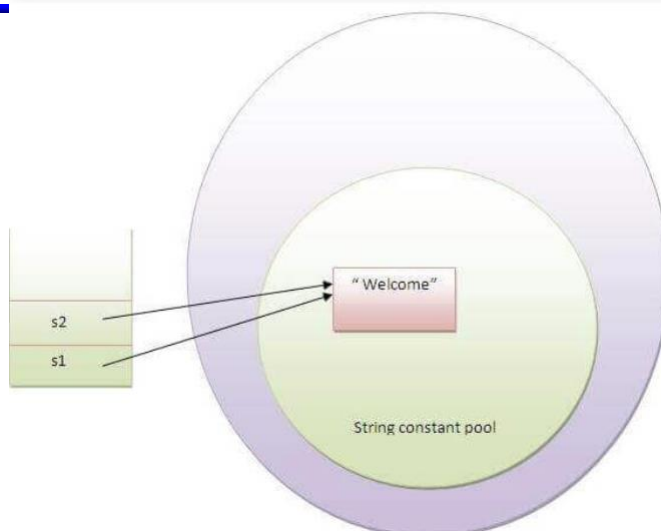


Programmazione ad Oggetti - Prof. Massimo Ficco

33

```
String s1="Welcome";  
String s2="Welcome";//It doesn't create a new instance
```

V:



Programmazione ad Oggetti - Prof. Massimo Ficco

34

34

## \*Prova interi

V:

```
class TestStringa{
public static void main(String arg[])
{
    int a=6;
    int b=5;
    a=b;
    b=10;
    System.out.println(a);
    System.out.println(b);
}
}
```



## \*Riferimenti

V:

La dichiarazione di un oggetto equivale a dichiarare una reference

La creazione dell'oggetto rende valida la reference e alloca memoria per l'oggetto

Tutti gli oggetti devono essere creati



# Array Java



In Java un array è un oggetto

Si possono dichiarare array di qualsiasi tipo

*Esempio:*

- *char s[ ]; // dichiarazione di array*
- *int [ ] array;*
- *int [ ] x, y[ ];*
- *int x[ ], y[ ];*

Gli array sono creati mediante la parola chiave **new**, occorre indicare un tipo e un numero intero, non negativo, di elementi da allocare

Si possono creare array di array

Java controlla in modo rigoroso che il programmatore non tenti accidentalmente di memorizzare o far riferimento ad elementi dell'array tramite un indice non valido



## Esempio Array

V:

```
public class Prova{
    public static void main(String args[])
    {
        System.out.println("ciao");

        int v[] = new int[10];
        v[1]=2;
        ....
        v[10]=9;    //Java effettua un controllo sulle
                  //dimensioni e da un eccezione
    }
}
```

