

Linguaggi di Programmazione I – Lezione 8

Prof. Marcello Sette
mailto://marcello.sette@gmail.com
http://sette.dnsalias.org

29 aprile 2008

Identificatori e parole chiavi	3
Commenti	4
Blocchi	5
Identificatori	6
Parole chiavi	7
Esempi	8
Tipi primitivi	9
Elenco	10
boolean	11
char	12
E le stringhe?	13
Tipi interi	14
Litterali interi	15
Tipi a virgola mobile	16
Litterali a	17
Esempio	18
Tipi reference	19
Cosa sono	20
Costruzione di	21
- Allocazione di	22
- Inizializzazione	23
- Esecuzione del	24
- Assegnazione	25
E non è tutto!	26
Esempio	27
Parametri	28
Passaggio per valore	29
Esempio	30
Il riferimento this	31
Il riferimento this	31
Esempio (1)	32
Esempio (2)	33

Convenzioni sul codice	34
Convenzioni varie	34
Esercizi	35
Esercizi	35

Panoramica della lezione

Identificatori e parole chiavi

Tipi primitivi

Tipi reference

Parametri

Il riferimento `this`

Convenzioni sul codice

Esercizi

LP1 – Lezione 8

2 / 35

Identificatori e parole chiavi

3 / 35

Commenti

Tre stili di scrittura di commento al codice:

```
// commento su una singola riga
```

```
/* commento su una  
o piu' righe */
```

```
/** commento usato per la documentazione automatica  
del codice */
```

Il formato di quest'ultimo commento e l'uso dello strumento javadoc è discusso nella cartella `guide/javadoc` della documentazione delle API per Java 2 SDK.

LP1 – Lezione 8

4 / 35

Blocchi

- Un *enunciato* è costituito da una o più righe di codice terminate da un `';`:

```
totale = a + b + c  
        + d + e + f;
```

- Un *blocco* è la collezione di enunciati racchiusi tra parentesi graffe:

```
{  
  x = y + 1;  
  y = x + 1;  
}
```

- I blocchi possono essere annidati uno nell'altro.
- Il numero di spazi o righe bianche sono influenti.

LP1 – Lezione 8

5 / 35

Identificatori

- Sono nomi assegnati a variabili, classi, metodi.
- Possono cominciare con un carattere Unicode, underscore(_), oppure il dollaro (\$).
- Sono *case sensitive* e non hanno lunghezza massima.
- Esempi:

```
identificatore
userName
user_name
_sys_var1
$change
```

- Il nome di una classe è costituito solo da caratteri ASCII poiché molti sistemi non supportano Unicode.

LP1 – Lezione 8

6 / 35

Parole chiavi

abstract	continue	float	long	short	transient
boolean	default	for	native	static	true
break	do	goto	new	strictfp	try
byte	double	if	null	super	void
case	else	implements	package	switch	volatile
catch	extends	import	private	synchronized	while
char	false	instanceof	protected	this	
class	final	int	public	throw	
const	finally	interface	return	throws	

- Non possono essere usate come identificatori.
- true, false, null sono in minuscolo, non in maiuscolo come in C++. Strettamente parlando sono literal, non parole chiavi.
- Non c'è l'operatore sizeof: la dimensione e la rappresentazione dei tipi è fissa e non dipende dalla realizzazione della JVM.
- goto e const sono parole chiavi che non sono usate in Java.

LP1 – Lezione 8

7 / 35

Esempi

```
foobar // legale
BIGinterface // legale; contiene parola chiave
$guadagniMenoSpese // legale
3_node5 // illegale: comincia con cifra
!ilCubo // illegale: deve cominciare con
// lettera, $, o _
```

LP1 – Lezione 8

8 / 35

Elenco

Otto tipi primitivi:

- Logici: boolean
- Testuali: char
- Interi: byte, short, int, long
- Floating point: float, double

LP1 – Lezione 8

10 / 35

boolean

- Il tipo boolean ha due literal: true, false.
- Esempio:

```
boolean ok = true;
```

- Non c'è cast tra tipi interi e boolean. Interpretare valori numerici come valori logici non è permesso in Java.

LP1 – Lezione 8

11 / 35

char

- Rappresenta un carattere Unicode (16 bit).
- I literal di questo tipo sono inclusi tra apici singoli (' ').
- Esempi:

```
'a'           // la lettera a
'\t'          // una tabulazione
'\u03A6'      // la lettera greca phi
```

- Le stringhe non sono tipi primitivi.
- Fare riferimento alle specifiche del linguaggio Java per ulteriori codici '?\??'.

LP1 – Lezione 8

12 / 35

E le stringhe?

- String NON È UN TIPO PRIMITIVO, è una classe (comincia per lettera maiuscola).
- "Ha i suoi literal racchiusi tra apici doppi".
- Può essere usata come segue:

```
String saluto = "Buon giorno!! \n";
String messaggioErrore = "File not found!";
```

- Non è finita qui ...

LP1 – Lezione 8

13 / 35

Tipi interi

Rappresentano i valori:

Lunghezza	Tipo	Range
8 bit	byte	$-2^7 \dots 2^7 - 1$
16 bit	short	$-2^{15} \dots 2^{15} - 1$
32 bit	int	$-2^{31} \dots 2^{31} - 1$
64 bit	long	$-2^{63} \dots 2^{63} - 1$

LP1 – Lezione 8

14 / 35

Litterali interi

■ I litterali hanno tre forme: decimale, ottale ed esadecimale.

- ◆ 2 il valore decimale è due.
- ◆ 077 lo zero iniziale denota un valore ottale.
- ◆ 0xBAAC la parte 0x iniziale denota un valore esadecimale.

■ Assumono come tipo di default int.

■ Suffisso L oppure l se si vuole che siano di tipo long.

- ◆ 2L è due, rappresentato come long.
- ◆ 077L è un valore ottale, rappresentato come long.
- ◆ 0xBAACL è un valore esadecimale, rappresentato come long.

■ Quando si assegna il *valore di un litterale* ad una variabile, il compilatore determina la dimensione del litterale a seconda della variabile:

```
short s = 9; // questo e' ok
```

■ Quando si assegna il *valore di una espressione* ad una variabile, la dimensione del valore non è modificabile:

```
short s1 = 9 + s; // questo causa errore di compilazione
// perche' 9 + s e' int non short
```

LP1 – Lezione 8

15 / 35

Tipi a virgola mobile

Rappresentano i valori:

Lunghezza	Tipo
32 bit	float
64 bit	double

secondo lo standard (IEEE) 754.

LP1 – Lezione 8

16 / 35

Litterali a virgola mobile

- Hanno come tipo di default il tipo double.
- Suffisso F o f per litterali del tipo float.
- Suffisso D o d per litterali del tipo double.
- Esempi
 - ◆ 3.14 un semplice valore a virgola mobile (double).
 - ◆ 6.02E23 un altro valore.
 - ◆ 2.718F un semplice float.
 - ◆ 123.4E-5D un double con una D superflua.

LP1 – Lezione 8

17 / 35

Esempio

```
public class Assign {
    public static void main (String args []) {
        int x, y;
        float z = 3.1415f;
        double w = 3.14145;
        boolean truth = true;
        char c;
        String str;
        String str1 = "ciao";
        c = 'A';
        str = "ciao a tutti";
        x = 6;
        y = 1000;
    }
}
```

Queste sono assegnazioni illegali:

```
y = 3.14159; // 3.14159 non e' un int; richiede casting
w = 175,000; // la virgola invece del punto decimale
truth = 1; // errore comune tra programmatori C / C++
z = 3.14159; // 3.14159 non e' un float; richiede casting
```

LP1 – Lezione 8

18 / 35

Cosa sono

- Tutti i tipi non primitivi sono tipi *reference*.
- Una variabile *reference* contiene la "maniglia" di un oggetto.
- Esempio:

```
public class MiaData {
    private int giorno = 1;
    private int mese = 1;
    private int anno = 2006;
}
```

La classe MiaData può essere usata in questo modo:

```
public class TestMiaData {
    public static void main (String[] args) {
        MiaData oggi = new MiaData();
    }
}
```

Costruzione di oggetti

- `new Xxx()` serve ad allocare spazio per il nuovo oggetto. Scatena i seguenti processi:
 1. Viene allocato lo spazio per il nuovo oggetto e le variabili dell'istanza sono inizializzate al loro valore di default (e.g. 0, false, null, e così via).
 2. Viene eseguita ogni inizializzazione esplicita degli attributi.
 3. Viene eseguito un costruttore.
 4. Viene assegnato il riferimento finale all'oggetto.
- Esaminiamo separatamente ciascuna di queste fasi, mostrando ciò che succede quando viene eseguito il codice:

```
MiaData nascita = new MiaData (23, 4, 1964);
```


- Allocazione di memoria

Nell'enunciato:

```
MiaData nascita = new MiaData (23, 4, 1964);
```

- la dichiarazione `MiaData nascita` causa l'allocazione dello spazio memoria del solo riferimento (non ancora inizializzato):

nascita	????
---------	------

- l'uso della parola chiave `new`, alloca spazio per `MiaData` (inizializzata ai valori di default – vediamo poi):

nascita	????
---------	------

giorno	0
mese	0
anno	0

LP1 – Lezione 8

22 / 35

- Inizializzazione degli attributi

- Successivamente, per gli attributi sono usate le inizializzazioni esplicite all'interno della classe, quelle che eventualmente sono scritte nel momento della definizione dell'attributo (inizializzazione dell'oggetto da parte del progettista della classe):

nascita	????
---------	------

giorno	1
mese	1
anno	2006

LP1 – Lezione 8

23 / 35

- Esecuzione del costruttore

- Ora è invocato il costruttore. Con esso si possono sostituire inizializzazioni personali dell'utente dell'oggetto a quelle di default previste dal progettista della classe. Si possono anche passare argomenti, così che il codice che richiede la costruzione del nuovo oggetto possa controllare l'oggetto che verrà creato:

nascita	????
giorno	23
mese	4
anno	1964

LP1 – Lezione 8

24 / 35

- Assegnazione del riferimento

- L'assegnazione, infine, inserisce l'indirizzo del nuovo oggetto nella locazione del riferimento:

nascita	0x0123abcd
	↓
giorno	23
mese	4
anno	1964

LP1 – Lezione 8

25 / 35

E non è tutto!

Purtroppo il processo di costruzione di oggetti e della loro inizializzazione è notevolmente più complesso di come è stato descritto qui.

Ritorniamo successivamente su questo argomento.

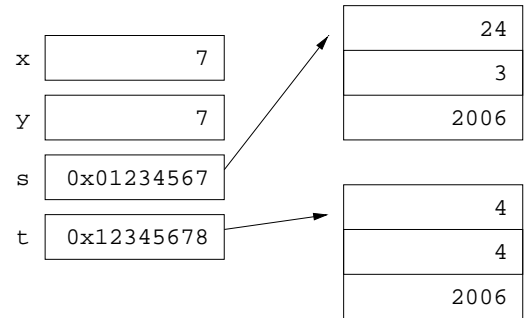
LP1 – Lezione 8

26 / 35

Esempio

```
int x = 7;
int y = x;

MiaData s = new MiaData(24, 3, 2006);
MiaData t = s;
t = new MiaData(4, 4, 2006);
```



LP1 – Lezione 8

27 / 35

Parametri

28 / 35

Passaggio per valore

- Java permette il passaggio dei parametri per valore (nella nostra tassonomia, parametri IN realizzati per copia).
- Il passaggio per riferimento (che permette la modifica del valore del parametro nel contesto della procedura chiamante) è PROIBITO IN JAVA.
- Quando si passa una istanza di oggetto come argomento di un metodo, quello che si sta passando non è l'oggetto, ma solo un riferimento a quell'oggetto. Questo riferimento è copiato nel parametro formale.
- Attenzione che, in quest'ultimo caso, sarà possibile la modifica nel contesto del chiamante (mai del valore della variabile) dell'oggetto a cui fa riferimento quella variabile.

LP1 – Lezione 8

29 / 35

Esempio

```
public class PassTest {
    public static void
        cambiaValore(int v) {
        v = 55;
    }
    public static void
        cambiaOggetto(MiaData d) {
        d = new MiaData(1, 1, 2005);
    }
    public static void
        cambiaAttributo(MiaData d) {
        d.setGiorno(16);
    }

    public static void
        main(String args[]) {
        MiaData data =
            new MiaData (24, 4, 2006);
        int val = 11;
```

```
        cambiaValore(val);
        System.out.println(
            "val vale: " + val);

        cambiaOggetto(data);
        data.print();

        cambiaAttributo(data);
        data.print();
    }
}
```

L'output è:

```
$ java PassTest
val vale: 11
MiaData: 24-4-2006
MiaData: 16-4-2006
```

LP1 – Lezione 8

30 / 35

Il riferimento this

La parola chiave this può essere usata:

- Per fare riferimento, all'interno di un metodo o di un costruttore locale, ad attributi o metodi locali.
Questa tecnica è usata per risolvere ambiguità in alcuni casi in cui una variabile locale di un metodo maschera un attributo locale dell'oggetto.
- Per permettere ad un oggetto di passare il riferimento a se stesso come parametro ad un altro metodo o costruttore.

L'esempio seguente mostra le tecniche precedenti.

Esempio (1) – La classe MiaData

```
public class MiaData {
    private int giorno = 1;
    private int mese = 1;
    private int anno = 2006;

    public MiaData (int giorno,
                    int mese,
                    int anno) {
        this.giorno = giorno;
        this.mese = mese;
        this.anno = anno;
    }

    public MiaData(MiaData data) {
        giorno = data.giorno;
        mese = data.mese;
        anno = data.anno;
    }
}
```

```
public MiaData addGiorni
                (int g) {
    MiaData data =
        new MiaData(this);

    data.giorno = data.giorno + g;
    // Non ben realizzato ...

    return data;
}

public void print() {
    System.out.println(
        "MiaData: " + giorno +
        "-" + mese + "-" + anno);
}
}
```

Esempio (2) – La classe TestMiaData

Per provare il funzionamento, scriviamo:

```
public class TestMiaData {
    public static void main(String[] args) {
        MiaData nascita = new MiaData(23, 4, 1964);
        MiaData la_settimana_dopo = nascita.addGiorni(7);

        la_settimana_dopo.print();
    }
}
```

Quindi:

```
$ java TestMiaData
MiaData: 30-4-1964
```

Convenzioni varie

- Pacchetti:
package oggetti.geometria;
- Classi:
class Circonferenza;
- Interfacce:
interface FiguraPiana;
- Metodi:
getOffset();
- Variabili:
dimensioneX
- Costanti:
PI_GRECO
- Parentesi:

```
if ( condition ) {
    do something
} else {
```

```
do something else
}
```

- Spaziatura: un solo enunciato per riga; due spazi di indenting per i blocchi annidati.
- Commenti:

```
// Un commento su una riga

/* Commenti su piu' righe
   ehgir 'uip us itnemmoC */

/** Commento per documentazione
 * automatica.
 * @see Altra classe per
 * maggiori informazioni
 */
```

Esercizi

Esercizi

1. Sulla creazione e uso degli oggetti.
2. Estensione dell'esercizio della lezione precedente (pacchetto banca).