

Tipi di dati primitivi

Programmazione
Corso di laurea in Informatica

La nozione di tipo di dato

- Il tipo del dato consente di esprimere la natura del dato
- Indica il modo con cui verrà interpretata la sequenza di bit che rappresenta il dato
 - La stessa sequenza può rappresentare un intero o un carattere ad esempio
- Determina il campo dei valori che un dato può assumere
- Specifica le operazioni possibili sui dati

AA 2003/04 © Alberti 2 Programmazione 5. Tipi di dato

Tipi di dati e Java

- Java è un linguaggio **fortemente tipizzato**
- Il tipo di ogni variabile o espressione può essere identificato leggendo il programma ed è già noto al momento della compilazione
 - È obbligatorio dichiarare il tipo di una variabile prima di utilizzarla
- Dopo la dichiarazione non è possibile assegnare alla variabile valori di tipo diverso
 - Salvo i casi di **shadowing** della **visibilità**

AA 2003/04 © Alberti 3 Programmazione 5. Tipi di dato

Tipi di dato

- Ogni tipo di dato è individuato da un **nome** (parola riservata)
 - `int, double, char`
- Che definisce un **insieme di valori** letterali possibili
 - `3, 3.1, 'c'`
- E di **di operazioni lecite**
 - `+, *`

In Java i dati sono di due categorie di tipi:

- tipi semplici
- tipi di oggetti o riferimenti a oggetti

AA 2003/04 © Alberti 4 Programmazione 5. Tipi di dato

Tipi di dati primitivi semplici

8 tipi di dati **primitivi** per rappresentare:

- numeri interi
 - `byte, short, int, long`
- numeri decimali in virgola mobile
 - `float, double`
- i caratteri
 - `char`
- i valori booleani
 - `boolean`

AA 2003/04 © Alberti 5 Programmazione 5. Tipi di dato

Tipi di dati primitivi numerici

- La differenza tra i diversi tipi per dati numerici consiste nello spazio di memoria che viene utilizzata per la rappresentazione
 - E quindi nell'intervallo di valori che possono rappresentare

Tipo	Memoria		Valore max
	in byte	Valore min	
<code>byte</code>	1	-128	127
<code>short</code>	2	-32,768	32,767
<code>int</code>	4	-2,147,483,648	2,147,483,647
<code>long</code>	8	< -9 x 10 ¹⁸	> 9 x 10 ¹⁸
<code>float</code>	4	+/- 3.4 x 10 ³⁸ con 7 cifre significative	
<code>double</code>	8	+/- 1.7 x 10 ³⁰⁸ con 15 cifre significative	

AA 2003/04 © Alberti 6 Programmazione 5. Tipi di dato

Ordini di grandezza degli interi

tipo	occ	numero di combinazioni	circa	ordine
byte	1	2 ⁸	256	
		2 ¹⁰		10 ³ mille
short	2	2 ¹⁶	65536	
		2 ²⁰		10 ⁶ milione
		2 ³⁰		10 ⁹ miliardo
int	4	2 ³²	4.294.967.296	
		2 ⁴⁰		10 ¹² mille miliardi
		2 ⁵⁰		10 ¹⁵ milione di miliardi
		2 ⁶⁰		10 ¹⁸ miliardo di miliardi
		2 ⁸⁰		
long	8	2 ⁶⁴	1.844.672.545.073.135.616	

Notazione scientifica e precisione

- I tipi che rappresentano numeri decimali possono essere visualizzati in diversi modi:
 - 3.14159712 9.0 0.5e+001 -16.3e+002
- Dove e indica la notazione scientifica in potenze di 10 (**notazione-e**) e separa il numero dall'esponente cui elevare la base 10
- La velocità della luce è di 299.792,5 Km/sec si può scrivere come 2.997925e8 in Java
- I tipi **float** e **double** si archiviano come valori approssimati
 - Circa le prime 7 o 15 cifre decimali possono essere archiviate

Operatori sui tipi aritmetici

tipo	simbolo	operazione	esempio
float, double	+	somma	4.50e01 + 5.30e00 = 5.03e01
	-	sottrazione	6.57e02 - 5.7e01 = 6.00e02
	*	moltiplicazione	7e03 * 3.0e00 = 2.1e04
	/	divisione	9.6e01 / 2e01 = 4.8e00
		%	resto
byte, short, int, long	+	somma	45 + 5 = 50
	-	sottrazione	657 - 57 = 600
	*	moltiplicazione	7000 * 3 = 21000
	/	divisione	10 / 3 = 3
		%	resto

Intervalli numerici e precisione

- I tipi interi (**byte, short, int, long**) rappresentano numeri interi in un dato intervallo
 - Per il tipo **int**, le costanti **Integer.MIN_VALUE** e **Integer.MAX_VALUE** danno gli estremi dell'intervallo
- I tipi in virgola mobile (**float, double**) rappresentano i numeri con una precisione finita e introducono approssimazioni
- Problemi di **overflow** e **perdita di precisione**
 - Es: [Overflow.java](#) e [Precisione.java](#)

I caratteri

- Una variabile **char** contiene un singolo carattere dell'insieme dei caratteri **Unicode**
 - Un insieme di caratteri (ASCII, Unicode, ...) è una *lista ordinata*, e a ogni carattere corrisponde un numero
- L'insieme **Unicode** usa 16 bits per rappresentare un carattere, ammettendone quindi 65,536 diversi
 - E' uno standard internazionale e contiene caratteri e simboli per molti diversi linguaggi
- I caratteri letterali sono delimitati dal carattere **'**

'a' 'x' '7' '\$' ', ' '\n'

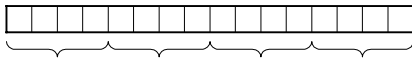
I caratteri ASCII

- L'insieme dei caratteri **ASCII** è più vecchio e più piccolo dell'Unicode, ma ancora in uso
- L'insieme dei caratteri **ASCII** è un sottoinsieme dell'insieme Unicode che comprende:

maiuscole	A, B, C, ...
minuscole	a, b, c, ...
punteggiatura	punto, punto e virgola, ...
cifre	0, 1, 2, ...
simboli speciali	&, , \, ...
caratteri di controllo	ritorno, tabulazioni, ...

I caratteri UNICODE

- Sono codificati tra i valori `\u0000` e `\uffff`
- La sequenza di escape `\u` indica che il numero che segue è un carattere **Unicode** e i numeri sono espressi in esadecimale
 - **Numeri esadecimali** sono rappresentati in base 16 quindi mediante le cifre 0-9 e a-f. 16 cifre in tutto appunto



ciascuna delle 16 cifre esadecimali rappresenta una delle 16 possibili configurazioni di 4 bit

La rappresentazione dei caratteri

Formato ASCII esteso

- Utilizza 8 bit, quindi rappresenta 256 valori
- La prima metà è il formato ASCII, ottenuta utilizzando solo 7 bit
- La seconda metà rappresenta vari caratteri speciali
 - Le lettere accentate o con segni particolari: umlaut o cediglie ...
- ISO 8859 del 1980 stabilisce lo standard per il formato ASCII esteso a 8 bit
 - 0-127 ISO-Latin 1 (il vecchio ISO 646)
 - 128-159 non sono usati
 - 160-255 i caratteri speciali
- Ma **256 caratteri non** sono comunque **sufficienti** per rappresentare i caratteri di tutte le lingue

Formato ISO10646

- ISO10646 è una collezione di 2^{32} caratteri organizzati in un ipercubo a 4 dimensioni
 - 256 gruppi di 256 piani di 256 righe di 256 caratteri di 8 bit (g,p,r,c)
 - Il formato **Unicode**, chiamato anche *Basic Multilingual Plane*, è (0,0,r,c) e rappresenta tutti i set di caratteri inclusi il cinese, il giapponese e il coreano
- **Encoding** è la funzione che mappa un codice in una sequenza di byte per la trasmissione o l'archiviazione
 - **Quoted Printable**: i caratteri tra 128 e 255 sono rappresentati con 3 byte di cui il 1° è il segno =, gli altri contengono il valore del codice in esadecimale
 - è diventa =E9
 - **UCS-2** trasmette solo i due piani utilizzati da UNICODE
 - **UTF8** inoltre li trasmette in opportune sequenze di byte a 8 bit

Il tipo boolean

- Un valore **boolean** rappresenta una condizione di verità o falsità
- Una variabile di tipo boolean può rappresentare un valore a due stati
 - come un interruttore che è **acceso** o **spento**
- Le parole riservate **true** e **false** sono gli unici valori ammessi per il tipo boolean

```
boolean eseguito = false;
```

Espressioni booleane

- Sono espressioni che riportano un valore di tipo booleano
- Vengono usate principalmente per esprimere le condizioni in istruzioni di controllo del flusso


```
dato > 10
nome_1 < nome_2
(dato < 5) && !finito
```

Espressioni aritmetiche

- Una **espressione** è una combinazione di operatori e operandi
- Una **espressione aritmetica** calcola valori numerici e usa operatori aritmetici:

somma	+
sottrazione	-
moltiplicazione	*
divisione	/
resto	%
- Se uno o entrambi gli operandi di un operatore sono di tipo virgola mobile, il risultato è di tipo virgola mobile

AA 2003/04 © Alberti 19 Programmazione 5. Tipi di dato

Divisione e resto

- Se entrambi gli operandi dell'operatore / sono interi, il risultato è intero e la parte decimale è persa

14 / 3	uguale a	4
8 / 12	uguale a	0
- L'operatore resto % riporta il resto della divisione

14 % 3	uguale a	2
8 % 12	uguale a	8
- Esempio [Divisione.java](#)

AA 2003/04 © Alberti 20 Programmazione 5. Tipi di dato

Precedenza tra operatori

- Gli operatori possono venire combinati in espressioni complesse


```
risultato = totale + cont / max - scarto;
```
- Gli operatori hanno una **precedenza** ben definita implicita che determina l'ordine con cui vengono valutati
 - Moltiplicazione, divisione e resto sono valutati prima di somma, sottrazione e concatenazione tra stringhe
- Gli operatori che hanno la stessa precedenza sono valutati da sinistra a destra
- Mediante le parentesi si può alterare l'ordine di precedenza

AA 2003/04 © Alberti 21 Programmazione 5. Tipi di dato

Precedenza tra operatori

- Ordine di valutazione dell'espressione:

a + b + c + d + e	a + b * c - d / e
1 2 3 4	3 1 4 2

a / (b + c) - d % e
2 1 4 3

a / (b * (c + (d - e)))
4 3 2 1

AA 2003/04 © Alberti 22 Programmazione 5. Tipi di dato

Operatore di assegnamento

- Ha la precedenza più bassa di qualunque altro operatore

Prima si valuta l'espressione alla destra dell'operatore =

```
risposta = somma / 4 + MAX * altro;
```

4	/	1	+	3	*	2
---	---	---	---	---	---	---

Poi il risultato è assegnato alla variabile alla sinistra

AA 2003/04 © Alberti 23 Programmazione 5. Tipi di dato

Assegnamento, ancora

- I membri di destra e sinistra dell'operatore = (**assegnamento**) possono contenere la stessa variabile

dapprima, si aggiunge 1 al valore originario della variabile

```
numero = numero + 1;
```

Poi il risultato è immagazzinato come nuovo valore della variabile (il vecchio valore è sovrascritto)

Si può anche usare l'assegnamento +=; numero += 1;

AA 2003/04 © Alberti 24 Programmazione 5. Tipi di dato

Assegnamento in variabili di riferimento

- Si consideri l'assegnamento
 $x = y;$
- Se x e y sono *variabili di tipo semplice*, il risultato dell'istruzione sarà che il contenuto di y viene copiato come contenuto di x
- Se x e y sono *variabili di tipo oggetto* e contengono il riferimento all'oggetto allora conseguenza dell'istruzione è che il riferimento di y viene copiato come contenuto di x
 - x e y si riferiscono ora allo stesso oggetto
- In entrambi i casi il contenuto della variabile y non cambia

AA 2003/04 © Alberti 25 Programmazione 5. Tipi di dato

Conversione tra tipi primitivi

- Qualche volta è utile o necessario convertire i dati da un tipo ad un altro
 - Un intero considerarlo come un numero in virgola mobile
- Le conversione vanno fatte con cautela per non perdere informazioni
- Le *conversioni larghe* sono sicure perchè vanno da un tipo di dato che usa un certo quantitativo di memoria ad uno più grande (da **short** a **int**)
- Le *conversioni strette* possono perdere informazioni perchè restringono l'occupazione di memoria a disposizione del dato (da **int** a **short**)

AA 2003/04 © Alberti 26 Programmazione 5. Tipi di dato

Conversioni - 2

- Le conversioni tra tipi di dato possono avvenire in 3 modi:
 - Conversioni* durante una operazione di assegnamento
 - Promozione* in una espressione aritmetica
 - Casting*
- a.** quando un valore di un tipo viene assegnato ad una variabile di un altro tipo
 - È consentita solo la conversione larga
- b.** avviene automaticamente quando operatori aritmetici devono convertire gli operandi

AA 2003/04 © Alberti 27 Programmazione 5. Tipi di dato

Conversioni - 3

- c.** conversione detta *casting* è la tecnica di conversione più pericolosa e potente
- Tramite un casting esplicito si possono realizzare sia la *conversione larga* sia quella *stretta*
- Per effettuare un casting di tipo si dichiara il nuovo tipo tra () di fronte al valore di cui si vuole fare la conversione
- **int totale, numero;**
float risultato; vogliamo dividere senza perdita d'informazione **totale** per **numero** effettuiamo un cast su **totale**:

risultato = (float) totale / numero;

AA 2003/04 © Alberti 28 Programmazione 5. Tipi di dato

Tabella conversioni tipi

da \ a	boolean	byte	short	char	int	long	float	double
boolean		n	n	n	n	n	n	n
byte 8bit	n		s	c	s	s	s	s
short 16bit	n	c		c	s	s	s	s
char 16 bit	n	c	c		s	s	s	s
int 32bit	n	c	c	c		s	s*	s
long 64bit	n	c	c	c	c		s*	s
float 32bit	n	c	c	c	c	c		s
double 64bit	n	c	c	c	c	c	c	

n non si applica; s viene fatto automaticamente; c mediante casting esplicito

AA 2003/04 © Alberti 29 Programmazione 5. Tipi di dato

Librerie di classi

- Una *libreria* è una collezione di classi che possono essere usate nei programmi
- La *libreria standard* fa parte di ogni sistema di sviluppo Java
- Le classi della libreria NON fanno parte del linguaggio, ma vengono usate continuamente
- La classe **System** e la classe **String** sono parte della libreria di classi standard di Java
- Altre librerie possono essere prodotte da terze parti o sviluppate da voi stessi

AA 2003/04 © Alberti 30 Programmazione 5. Tipi di dato

Packages

- Le classi della libreria standard sono organizzate in pacchetti

Package

```
java.lang
```

```
java.applet  
java.awt  
javax.swing  
java.net  
java.util  
java.text  
java.math
```

Scopo

supporto generale allo sviluppo importato automaticamente
creare applets per il web grafica e interfacce grafiche per GUI
ulteriori componenti grafiche per GUI
comunicazione di rete
utilità varie
visualizzare testo formattato
eseguire calcoli

AA 2003/04
© Alberti

31

Programmazione
5. Tipi di dato

Dichiarazione di import

- Per usare una classe di un pacchetto, si indica il suo nome per esteso

```
java.util.Random;
```

- O si importa la classe e quindi si indica solo il nome della classe

```
import java.util.Random;
```

- Per importare tutte le classi di un pacchetto si usa il carattere 'jolly' *

```
import java.util.*;
```

AA 2003/04
© Alberti

32

Programmazione
5. Tipi di dato

Dichiarazione di import - 2

- Tutte le classi del pacchetto `java.lang` sono importate automaticamente
 - Quindi non dobbiamo esplicitamente importare le classi `System` o `String` ad esempio
- La classe `Random` class è parte del pacchetto `java.util`
 - Fornisce metodi per generare numeri pseudocasuali
 - Per determinati scopi occorrono numeri in certi intervalli
- Esempio [RandomNumbers.java](#)

AA 2003/04
© Alberti

33

Programmazione
5. Tipi di dato

Metodi statici di classe

- Alcuni metodi possono essere invocati tramite la classe, invece che tramite un oggetto della classe
- Sono i **metodi di classe** o **metodi statici**
- La classe `Math` del pacchetto `java.lang` contiene molti metodi statici, che eseguono varie funzioni matematiche
 - Il valore assoluto, funzioni trigonometriche, radici quadrate, etc.

```
temp = Math.cos(90) + Math.sqrt(delta);
```

AA 2003/04
© Alberti

34

Programmazione
5. Tipi di dato

Invocazione dei metodi statici

- Si possono invocare in 2 modi:
 - `<NomeClasse>.<NomeMetodo>`
 - `<referimentoOggetto>.<NomeMetodo>`
 - Usare sempre il `|` modo
- I metodi statici sono risolti durante la compilazione
- I metodi d'istanza in esecuzione
- Le eventuali sottoclassi non hanno una propria copia dei metodi statici, ma condividono la stessa

AA 2003/04
© Alberti

35

Programmazione
5. Tipi di dato

Numeri pseudo-casuali

- Oltre che tramite la classe `Random` del pacchetto `java.util`, i numeri pseudocasuali sono generabili con il metodo statico `random()` della classe `Math` del pacchetto `java.lang`

```
public static double random()
```

- Riporta un valore `x` nell'intervallo `0.0 <= x < 1.0`
- Per calcolare un valore intero effettuare un cast esplicito. Es: `0 <= x <= 4`

```
int x = (int) (Math.random () * 5)
```

AA 2003/04
© Alberti

36

Programmazione
5. Tipi di dato

La classe Keyboard

- La classe **Keyboard** non fa parte della libreria di classi standard di Java
- I dettagli della classe **Keyboard** sono contenuti nel Capitolo 8 del Lewis
- Istruzioni per scaricarla e installarla alla pagina: homes.dico.unimi.it/~alberti/Prog03/
- [Echo.java](#) e [Quadratic.java](#)
- [TestDado.java](#) e [Dado.java](#)

AA 2003/04
© Alberti

37

Programmazione
5. Tipi di dato

Formattare l'output

- La classe **NumberFormat** offre metodi statici per formattare oggetti. Ad esempio:
`getCurrencyInstance()`
`getPercentInstance()`
- E' definita nel pacchetto **java.text**
- Non è necessario instanziare un oggetto mediante il costruttore **new**
- A ogni oggetto della classe **NumberFormat** può essere inviato il messaggio **format** che accetta un argomento numerico e riporta una stringa contenente il valore formattato nel modo appropriato
- Esempio [Price.java](#)

AA 2003/04
© Alberti

38

Programmazione
5. Tipi di dato

Fomattare l'output – 2

- La classe **DecimalFormat** può essere usata per formattare numeri in virgola mobile in vari modi
 - Ad esempio vogliamo solo 3 cifre decimali
- Un oggetto della classe **DecimalFormat** va instanziato con il costruttore **new**
- Il costruttore della classe **DecimalFormat** richiede che si specifichi il formato in una stringa che le viene passata come parametro
- Esempio [CircleStats.java](#)

AA 2003/04
© Alberti

39

Programmazione
5. Tipi di dato

La classe DecimalFormat

Esempi di pattern diversi per rappresentare il numero 123.456 mediante DecimalFormat.

pattern	risultato
"###.##"	123.46
"#####.#####"	123.456
"000000.000000"	000123.456000
"#####.000000"	123.456000
"000000.#####"	000123.456
" ###.###"	123.456
"\$#####.#####"	\$123.456
"\$\$#####.#####"	\$\$123.456

AA 2003/04
© Alberti

40

Programmazione
5. Tipi di dato