Gocce di Java

Tipi di dato primitivi

Pierluigi Crescenzi

Università degli Studi di Firenze

- Concetti base
  - Dati
    - Variabile
    - Tipo
  - Istruzioni
    - Istruzioni base
    - Strutture di controllo
    - Sotto-programmi

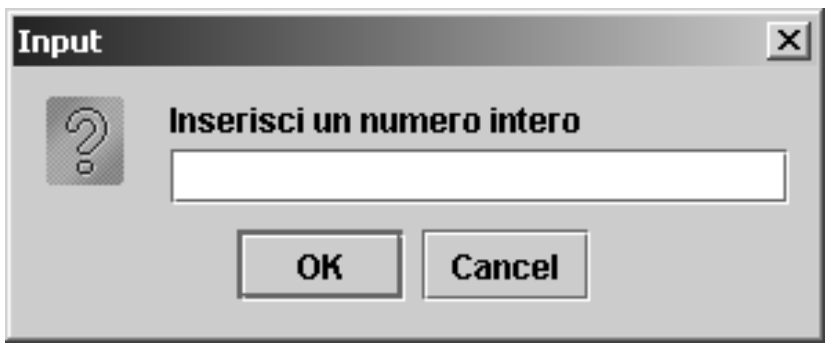
- **Variabili**: usate per immagazzinare dati come numeri e lettere
  - **Nome**: identificatore con cui riferirsi alla variabile
  - **Valore**: dato contenuto in variabile
  - **Tipo**: genere di dati che variabile può contenere
- **Istruzioni di base**
  - **Istruzione di assegnamento**: permettono di assegnare valore a variabile
    - Esempio

```
somma = primoNumero+secondoNumero;
```

  - **Istruzioni di input** e **output**: permettono di inserire dati da file o da tastiera e scrivere su disco, monitor o stampante

- Input e output in JAVA--

```
int x=Input.getInt("Inserisci un numero intero");
System.out.println( "Numero inserito: "+x );
```



  - Stampa nel pannello etichettato `Output`
  - Anche `getLong`, `getDouble` e `getChar`

- **Dichiarazione**: consente di conoscere nome di variabile, memoria da riservare per variabile e come memorizzare
  - Consiste di tipo, seguito da lista di nomi (separati da virgole) terminata con punto e virgola
- Sintassi

  tipo variabile_1,variabile_2, ...;
- Esempio

```
int punteggio;
char lettera;
double larghezza,altezza;
```

- **Identificatore**: nome di variabili (e di altro)
  1. Non possono cominciare con una cifra
  2. Contengono lettere, cifre, simbolo _ e simbolo $
  3. Sono *sensibili alle maiuscole*
- Buone norme
  1. Devono avere un significato e suggerire utilizzo di variabile
  2. Devono iniziare con lettera minuscola e seguire notazione "a cammello"
  3. Non devono contenere simbolo $
- Errori di ortografia rilevati da compilatore

Nome	Tipo	Memoria
`byte`	intero	1 byte
`short`	intero	2 byte
`int`	intero	4 byte
`long`	intero	8 byte
`float`	reale	4 byte
`double`	reale	8 byte
`char`	carattere	2 byte
`boolean`	vero/falso	1 byte

Numeri in virgola mobile

- Rappresentazione **floating point** di x: *mantissa* m ed *esponente* e tali che $x = m \cdot B^e$
  - Notazione scientifica: $B = 10$, anche indicata con E
    - Esempio: `2.34E+2` è 234.0 e `1.234E-3` è 0.001234
- Mantissa ed esponente occupano quantità fissata di memoria e sono limitati inferiormente e superiormente
  - Numero cifre decimali in mantissa indica precisione rappresentazione
    - Numeri `double`: numeri in virgola mobile a precisione doppia

Istruzione di assegnazione

- Sintassi

  variabile `=` *espressione*`;`

- Esempio

```
punteggio = giuste-sbagliate;
area = larghezza*altezza;
punteggio = punteggio+1;
```

- Variabile a sinistra segno uguale posta uguale a valore espressione a destra
  - Segno di uguale detto *operatore di assegnazione*
  - *Espressione*: variabile, costante o espressione costituita da operatori aritmetici

Esempio

```
double base;
double altezza;
double area;
base = 4.0;
altezza = 8.0;
area = (base*altezza)/2;
System.out.println( "Area: "+area );
```

Formula di Gauss

- Somma primi n numeri interi

$$\sum_{i=1}^{n} i = \frac{n(n+1)}{2}$$

```
int n;
n = Input.getInt( "Inserisci il numero n" );
int somma;
somma = n*(n+1)/2;
System.out.println( "Somma: "+somma );
```

- Stesso identificatore può apparire in due lati operatore di assegnazione

```
int contatore;
contatore = 5;
System.out.println( "Contatore: "+contatore );
contatore = contatore-1;
System.out.println( "Contatore: "+contatore );
contatore = contatore-1;
System.out.println( "Contatore: "+contatore );
```

  - Sottrae uno a valore di `contatore` e usa risultato per impostare nuovo valore di `contatore`

Letterali

- A differenza di una variabile, il valore di un letterale non cambia
  - Letterali di tipo numero come 5.0 (costante reale) e 5 (costante di tipo intero)
  - Letterali di tipo carattere come `'A'`, `'B'` e `'*'`
  - Letterali di tipo Booleano, ovvero le costanti `true` e `false`
  - Letterali di tipo stringa: come `"Contatore:  "`

Numeri reali e perdita di precisione

- Numeri `double` più estesi di numeri `int` ma meno precisi

```
double numeroOrig = 2E14;
double numeroRid = numeroOrig-0.05;
double differenza = numeroOrig-numeroRid;
System.out.println( "Differenza: "+differenza );
```

Scambio di variabili

- Operazione molto frequente
- Non può essere ridotta a esecuzione di

```
x = y;
y = x;
```

  - Si perde valore iniziale di `x` ed entrambe le variabili hanno al termine dell'esecuzione il valore di `y` prima dello scambio

- Può essere eseguita da istruzioni:

```
z = x;
x = y;
y = z;
```

- Opzionale ma consigliato
- Sintassi

  tipo var_1 = esp_1,var_2 = esp_2, ...;
- Esempio

```
int punteggio = 0;
char lettera = 'p';
double altezza = 12.34, base = 5.1;
```

- Ottenuti combinando operatore di assegnazione con operatore aritmetico

```
int punteggio = 0;
System.out.println( "Punteggio: "+punteggio );
punteggio += 5;
System.out.println( "Punteggio: "+punteggio );
```

  - Istruzione
    `punteggio += 5;`
    equivalente a
    `punteggio = punteggio+5;`
- Espressione a destra trattata come singola unità
  - `x *= a+b;`
    equivalente a
    `x = x*(a+b);`
    e non a
    `x = x*a+b;`

- Due tipi: quelli su una singola riga e quelli su righe multiple
- Sintassi

```
// commento limitato a una singola linea

/* commento distribuito su piu' linee
senza limiti sul numero di righe */
```

- Esempio

```
boolean primo; // indica se il numero e' primo

/* La variabile primo indica se il numero e'
primo: la primalita' viene determinata dividendo
il numero per tutti i suoi possibili divisori. */
boolean primo;
```

- Conversione: necessaria per assegnare valore di un tipo a variabile di tipo diverso
  - Cambia il tipo del valore non della variabile
- *Implicita* (ovvero automatica): si assegna valore di tipo "più basso" a variabile di tipo "più alto" nella gerarchia
  1. `double`
  2. `float`
  3. `long`
  4. `int`
  5. `short`
  6. `byte`
- Esempio

```
double x;
int n = 5;
x = n;
System.out.println( "x: "+x );
```

Tipo di un'espressione

- Operandi tutti dello stesso tipo: tipo del valore di ritorno è quello degli operandi
- Operandi di tipo diverso: il tipo del valore di ritorno è quello più alto nella gerarchia
- Esempio

```
double a;
int n = 2;
double x = 5.1;
double y = 1.33;
a = (n*x)/y;
System.out.println( "a: "+a );
```

Conversione implicita e perdita di precisione

- Conversioni da tipi interi a tipi in virgola mobile: perdita di precisione
  - Non tutti i valori di tipo `int` sono rappresentati nel tipo `float`

```
int x = 2109876543;
float y = x;
int z = (int)y;
System.out.println( "x: "+x );
System.out.println( "y: "+y );
System.out.println( "z: "+z );
```

- Necessaria per assegnare valore di tipo “più alto“ a variabile di tipo “più basso”
  - `intero = reale;`
    genera errore
    ```
    possible loss of precision
    found   : double
    required: int
    ```
- Sintassi *var_1* `=` `(`*tipo*`)`*var_2*`;`
- Esempio

```
numeroIntero = (int)numeroReale;
carattere = (char)numeroIntero;
```

Troncamento

- Quando si converte un valore di tipo in virgola mobile in un tipo intero, la parte decimale viene ignorata

```
int numeroEuro;
double conto = 26.99;
numeroEuro = (int)conto;
System.out.println( "Euro: "+numeroEuro );
```

- Non tutti i numeri reali sono rappresentati in modo esatto: troncamento può causare perdita di precisione

```
double f = 4.35;
int n = (int)(100*f);
System.out.println( "n: "+n );
```

- Se almeno uno dei due operandi è di tipo `float` o `double`, risultato quello aspettato
- Se entrambi operandi sono di tipo intero, parte frazionaria ignorata

```
System.out.println( "5/4.0: "+5/4.0 );
System.out.println("5/4: "+5/4);
```

- In ASCII e Unicode codice cifre diverso da loro valore numerico

```
char cifra = '6';
int cifraIntera = cifra;
System.out.println( "Cifra: "+cifraIntera );
```

- Però codici cifre sono numeri interi consecutivi a partire da 48

```
char cifra = '6';
int cifraIntera = cifra-48;
System.out.println( "Cifra: "+cifraIntera );
```

- %: restituisce resto divisione primo operando per secondo
  - Il valore di ritorno di 20%6 è 2
- Diverse applicazioni
  - Consente di contare modulo un certo valore *n*
    - Ad esempio, 0, 1, 2, 0, 1, 2, ...
  - Consente di decidere se un numero multiplo di un altro
    - Primo algoritmo per massimo comun divisore
  - Parte integrante di algoritmo
    - Algoritmo di Euclide

- Espressioni seguono normali regole di precedenza
  1. Operatori **unari** (ovvero con un solo argomento) +, -, ++, --.
  2. Operatori **binari** (ovvero con due argomenti) *, /, %.
  3. Operatori binari +, -.
- Parentesi forzano precedenza

```
int x = 10,y = 2;
double f = 0.2;
double risultato = (x+y)*f;
System.out.println( "(x+y)*f: "+risultato );
risultato = x+(y*f);
System.out.println( "x+(y*f): "+risultato );
```

- Aumentano o diminuiscono di uno il valore di variabile intera
- `punteggio++;`
  equivale a
  `punteggio = punteggio+1;`
  e `punteggio--;`
  equivale a
  `punteggio = punteggio-1;`
- Operatore precede (segue) operando: valore variabile modificato prima (dopo) di essere usato

```
int contatore = 5;
int n = 2*(++contatore);
int m = 2*(contatore++);
System.out.println( "n: "+n );
System.out.println( "m: "+m );
System.out.println( "contatore: "+contatore );
```

- Tipo di dato usato per memorizzare collezione di variabili dello stesso tipo
- Sintassi

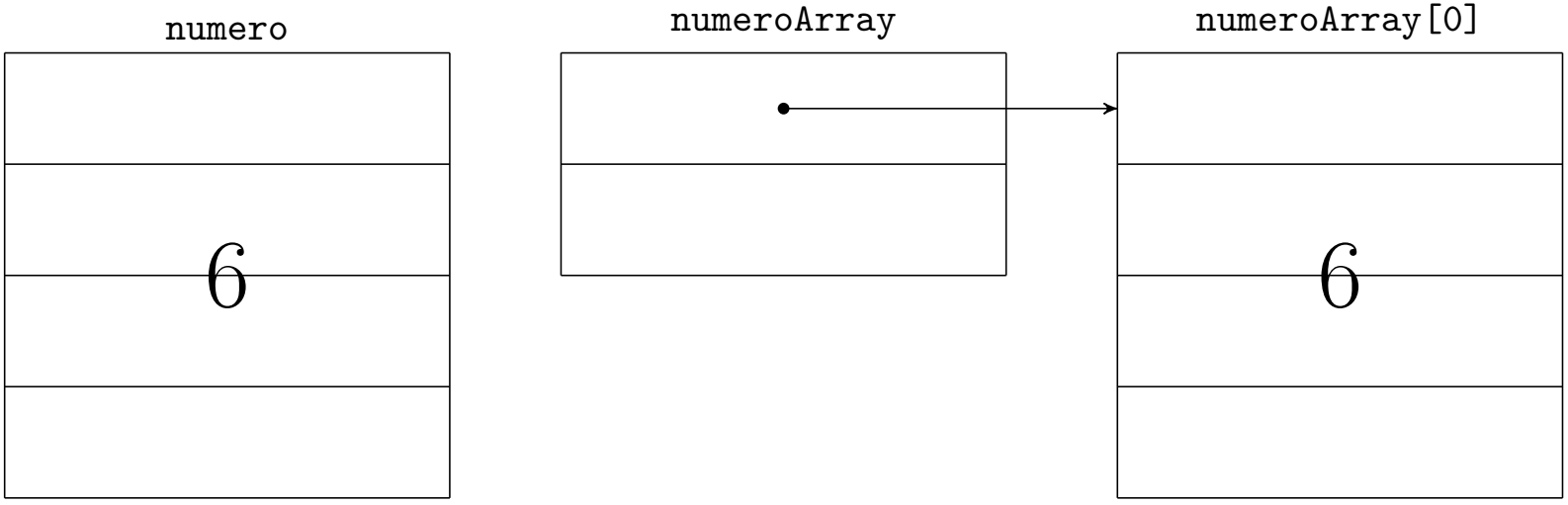
```
tipo_Base[] nome = {lista_Valori};
tipo_Base[] nome = new tipo_Base[numero_Elementi];
```

- Esempio

```
int[] comb = {2, 1, 4, 4};
double[] voto = new double[30];
```

Variabili di tipo array e locazioni di memoria

```
int numero = 6;
int[] numeroArray = {6};
```



Creazione di array e accesso a elementi

- Inizializzando i valori degli elementi

```
int[] comb = {2, 1, 4, 4};
int i = Input.getInt( "Indice tra 0 e 3" );
System.out.println( "comb["+i+"] = "+comb[i] );
```

- Specificando numero di elementi

```
char[] carattere = new char[80];
```

  - Elementi vanno inizializzati

Array e parentesi quadre

- Creare nome di tipo di dati

```
int[] arrayIntero;
```

- Parte di speciale sintassi Java per creare nuovi array

```
int[] arrayIntero = new int[100];
```

- Accedere a elemento di array

```
arrayIntero[3] = 1024;
```

- Numero di elementi contenuti in array
  - Accessibile con nome array seguito da punto e da `length`

```
int[] primo = {2,3,5,7,11,13};
System.out.println( "Lung: "+primo.length );
```

  - Non modificabile

```
primo.length = 10;
```

genera errore

```
cannot assign a value to final variable length
```

- Indici partono da 0
  - Ultimo indice uguale a lunghezza array meno 1

Indice	0	1	2	3
Valore	97	86	92	71

- Adattare proprio schema numerazione a numerazione Java

```
int[] punti = {97,86,92,71};
int i = 1;
System.out.println( "Punti "+i+": "+punti[i-1] );
i = i+1;
System.out.println( "Punti "+i+": "+punti[i-1] );
i = i+1;
System.out.println( "Punti "+i+": "+punti[i-1] );
i = i+1;
System.out.println( "Punti "+i+": "+punti[i-1] );
```

- Accesso a variabile indicizzata con indice fuori dominio genera errore `ArrayIndexOutOfBoundsException`

Gioco dell'oca

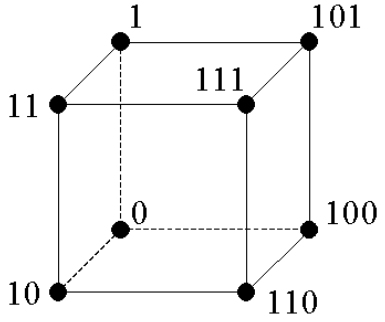
- Tabellone rappresentato da array
  - Valore elementi array: 0 oppure casella in cui spostarsi (premio o penalità)
    - Numerazione caselle a partire da 1

```
int[] tabellone =
{0,0,0,0,0,0,0,0,0,0,20,0,0,0,0,0,0,0,0,0,
0,33,0,0,0,0,0,0,35,0,0,0,0,19,0,0,0,48,0,0,
15,0,0,0,0,0,0,0,0,0,40,0,0,0,0,0,0,8,0,0};
```

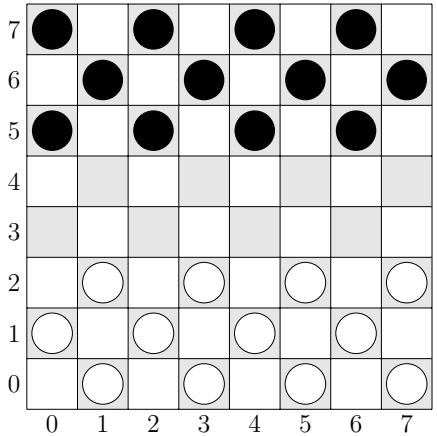
  - Quattro premi (20, 33, 35 e 48)
  - Quattro penalità (19, 15, 40 e 8)

- Consentono di strutturare elementi di collezione in forma di tabelle a più dimensioni
  - Numero di dimensioni determina numero di indici

```
byte[][][] punto = new byte[2][2][2];
punto[0][0][0] = 0;
punto[0][0][1] = 1;
punto[0][1][0] = 10;
punto[0][1][1] = 11;
punto[1][0][0] = 100;
punto[1][0][1] = 101;
punto[1][1][0] = 110;
punto[1][1][1] = 111;
```



Gioco della dama




```
char[][] damiera = {
   {'n','*','n','*','n','*','n','*'},
   {'*','n','*','n','*','n','*','n'},
   {'n','*','n','*','n','*','n','*'},
   {'*',' ','*',' ','*',' ','*',' '},
   {' ','*',' ','*',' ','*',' ','*'},
   {'*','b','*','b','*','b','*','b'},
   {'b','*','b','*','b','*','b','*'},
   {'*','b','*','b','*','b','*','b'}};
```