Lab. Programmazione (CdL Informatica) & Informatica (CdL Matematica) a.a. 2022-23

Monica Nesi

Università degli Studi dell'Aquila

2 Novembre 2022

Array bidimensionali in Java

Abbiamo visto gli array monodimensionali che consentono di rappresentare *sequenze lineari di elementi dello stesso tipo*.

Un array *bidimensionale* consente di rappresentare *sequenze di sequenze di elementi dello stesso tipo*, ovvero dati più complessi organizzati su più *righe* (e.g. tabelle, matrici).

Ad esempio, una sequenza di sequenze di interi

```
1 4 3 11
5 -3 3
3 -1 5 3 7
```

oppure una sequenza di sequenze di booleani

```
true true false
false true false true
```

Array bidimensionale: definizione

In generale, in Java si possono definire *array multidimensionali*, ovvero di dimensione n ($n \geq 1$).

N.B. L'array è una *struttura dati omogenea* (elementi tutti dello stesso tipo) indipendentemente dalle sue dimensioni.

Come si passa dalla dimensione 1 (array monodimensionale) alla dimensione 2 (array bidimensionale)? (Passare dalla dimensione n alla dimensione $n+1$ per $n>1$ è analogo).

Basta *applicare il costruttore di tipo array* [] *ad un tipo array monodimensionale*: dato un *qualsiasi* tipo array T[], applicare il costruttore di tipo [] risulta nel nuovo tipo T[][], ovvero il tipo *array bidimensionale* con elementi di tipo T.

Esempio: `int[][]` è il tipo *array bidimensionale di interi*, `boolean[][]` è il tipo *array bidimensionale di booleani*, etc.

Array bidimensionale: dichiarazione e creazione

Come per il caso monodimensionale, prima di creare un array bidimensionale occorre *dichiararlo*:

```
int [] []  a;
```

Questa dichiarazione dice che l'identificatore a è di tipo `int [] []`, ovvero lo si vuole usare per *riferire* un array bidimensionale di interi *che ancora non è stato creato*.

N.B. Il numero di coppie di `[]` determina la dimensione dell'array.

Per creare l'array si usa `new` seguito dal tipo degli elementi e da *due coppie di* `[]`, dove la prima coppia *deve* contenere la lunghezza dell'array a, ovvero il *numero delle righe* di a (i.e. gli array monodimensionali che sono gli elementi di a).
La seconda coppia di `[]` può contenere un'espressione numerica oppure essere vuota.

Array bidimensionale: creazione

Vediamo alcuni esempi.

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

Il numero 3 nella prima coppia di `[]` indica che l'array a ha 3 elementi, ovvero 3 array monodimensionali di interi (righe).

Quindi la lunghezza di a è 3. Il comando

```
System.out.println(a.length);
```

stampa 3.

I 3 array non sono però stati ancora creati!
L'operatore `new` ha allocato lo spazio necessario per i 3 riferimenti inizializzandoli a `null`.

Array bidimensionale: creazione (cont.)

Se stampiamo gli elementi dell'array a con il ciclo

```
for (int i=0; i<a.length; i++)
  System.out.println(a[i]);
```

si ottiene

```
null
null
null
```

Possiamo creare gli array monodimensionali `a[i]`, usando ancora `new`. Ad esempio:

```
a[0] = new int[2];
```

La prima riga di a viene creata con lunghezza 2 ed i suoi elementi sono inizializzati a 0.

Array bidimensionale: creazione (cont.)

Creiamo le altre righe dell'array a, ad esempio con lunghezza 4 e 3 rispettivamente:

```
a[1] = new int[4];
a[2] = new int[3];
```

N.B. Le righe di un array bidim. possono avere lunghezze diverse.

Stampando le loro lunghezze

```
for (int i=0; i<a.length; i++)
  System.out.println(a[i].length);
```

si ottiene

```
2
4
3
```

Array bidimensionale: implementazione

Quindi, in Java un array bidimensionale di tipo $T[][]$ è implementato come un *array di array*, ovvero un array monodimensionale i cui elementi sono a loro volta array monodimensionali di elementi di tipo T.

Il nome (*identificatore*) di un array è una referenza ad un'area di memoria in cui sono memorizzati gli elementi dell'array.

Nel caso di un array bidimensionale `a`, i suoi elementi `a[i]` sono array monodimensionali e quindi referenze a zone di memoria in cui sono memorizzati gli elementi `a[i][j]` delle righe `a[i]`.

Array bidimensionale: elementi

Gli elementi di un array bidimensionale sono individuati tramite il *nome/identificatore* dell'array e *due indici* (più in generale, due espressioni numeriche) che indicano, rispettivamente, la posizione della riga e la posizione dell'elemento all'interno della riga.

`a[i][j]` denota l'elemento dell'array `a` che si trova nella posizione `j` della riga `a[i]`.

Il primo indice `i` può variare tra `0` e `a.length-1`, mentre il secondo indice `j` può variare tra `0` e `a[i].length-1`.

Gli elementi di un array bidimensionale possono essere esaminati (e.g., dalla prima all'ultima riga e da sinistra a destra) tramite due *cicli annidati*, con il ciclo esterno il cui indice varia lungo le righe ed il ciclo interno il cui indice varia lungo la riga fissata.

Array bidimensionale: elementi (cont.)

Ad esempio, i seguenti cicli annidati

```
for (int i=0; i<a.length; i++) {
  for (int j=0; j<a[i].length; j++)
    System.out.print(a[i][j]+" ");
  System.out.println();
}
```

danno luogo alla stampa degli elementi di a, ancora tutti inizializzati al valore di default 0:

```
0 0
0 0 0 0
0 0 0
```

Ancora su creazione di array bidimensionali

In fase di creazione di un array bidim., è possibile specificare un valore numerico anche dentro la seconda coppia di `[]`:

```
int[][] a1 = new int[3][2];
```

In tal caso viene dichiarato un identificatore `a1` per riferire un array bidim. di interi avente 3 righe, *tutte* di lunghezza pari a 2 (ovvero, una *matrice* 3x2), con gli elementi tutti inizializzati al valore di default 0.

N.B. La lunghezza dell'array (i.e. la prima dimensione) deve essere *sempre* fissata in fase di creazione dell'array (altrimenti si ha errore in fase di compilazione).

A meno che non sia specificato esplicitamente, non si deve assumere che un array bidimensionale abbia righe della stessa lunghezza.

Creazione per enumerazione

Come nel caso monodimensionale, quando si conoscono gli elementi di un array bidim., è possibile crearlo *enumerando* gli elementi nell'ordine, racchiusi tra parentesi graffe e separati da virgole, *senza* usare `new` e *senza* indicare la lunghezza dell'array e delle singole righe.

Esempio:

```
int[][] b = {{1,2,3},{5,6,7,9},{-2,-5}};
```

Viene dichiarato e creato un array bidim. `b` di interi con 3 righe di lunghezza 3, 4 e 2, rispettivamente, e con un'inizializzazione esplicita dei suoi elementi.

Creazione per enumerazione (cont.)

Dato

```
int[][] b = {{1,2,3},{5,6,7,9},{-2,-5}};
```

stampare i suoi elementi con

```
for (int i=0; i<b.length; i++) {
  for (int j=0; j<b[i].length; j++)
    System.out.print(b[i][j]+"␣");
  System.out.println();
}
```

risulta nel seguente output:

```
1 2 3
5 6 7 9
-2 -5
```

Esempio

Determinare i valori stampati dal seguente programma:

```
class TestArrayBi {
  public static void main (String[] args) {
   int[][] b = new int[4][];
   for (int i=0; i<b.length; i++) {
     b[i] = new int [i+1];
     for (int j=0; j<b[i].length; j++)
       b[i][j] = i+j;
   }
   for (int i=0; i<b.length; i++) {
     for (int j=0; j<b[i].length; j++)
       System.out.print(b[i][j]+"␣");
     System.out.println();
   }
  }
}
```

Esempio: output

L'output del programma precedente è il seguente:

```
0
1  2
2  3  4
3  4  5  6
```

N.B. Nel caso di array bidim. creati con `new` e senza un valore numerico dentro la seconda coppia di `[]`, l'*errore tipico* consiste nel *dimenticare di creare le righe* `b[i]` prima di accedere agli elementi `b[i][j]`.

Se le righe `b[i]` *non* vengono create, allora gli elementi `b[i][j]` *non* esistono!