Array di array

Walter Didimo

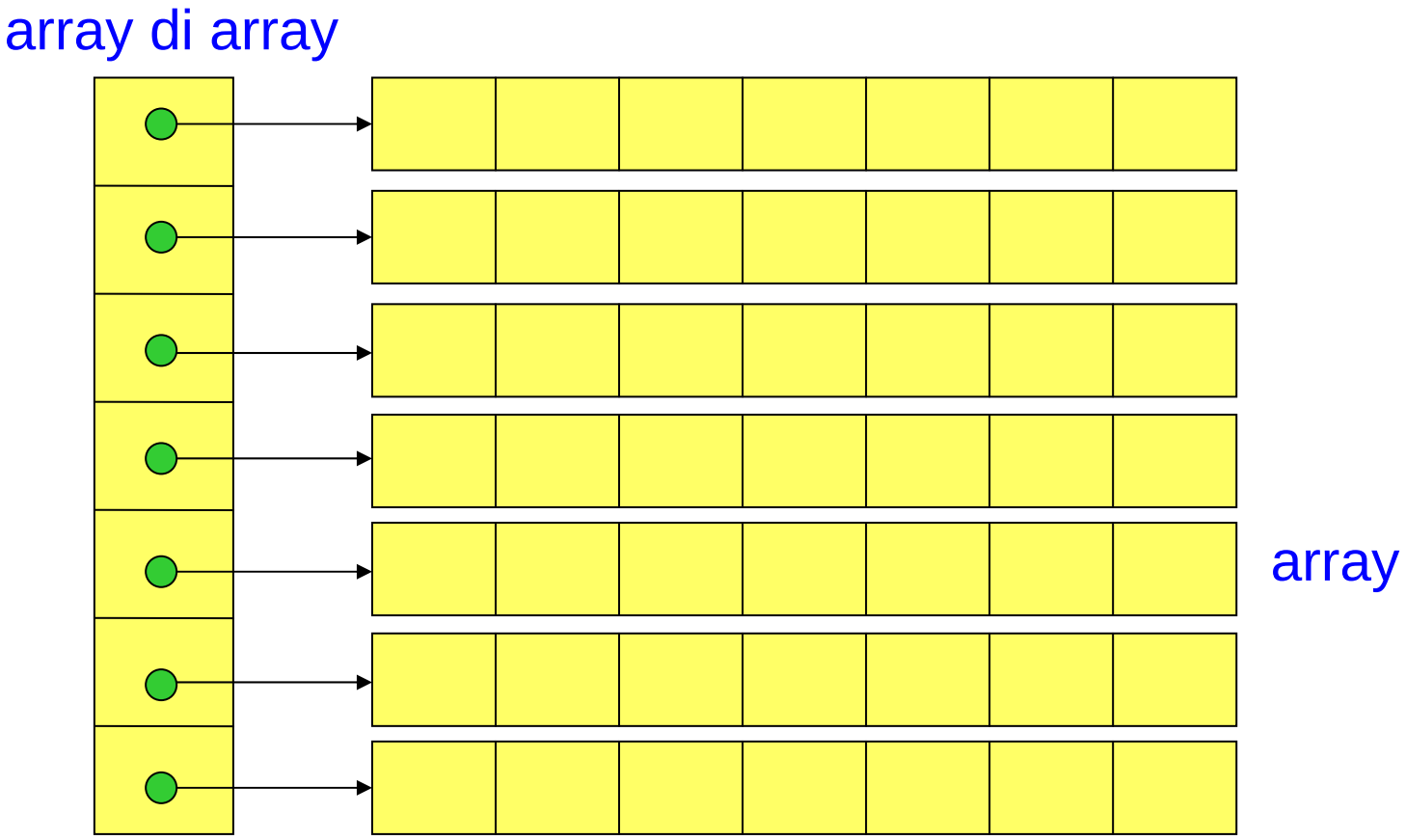
Array di array

Abbiamo visto che il tipo di dato degli elementi di un array può essere qualsiasi tipo valido

Il tipo di dato degli elementi di un array può dunque anche essere un tipo array

- si parla in questi casi di array di array
- un array di array è un array i cui elementi memorizzano a loro volta riferimenti ad array

Schema di un array di array



Multidimensionalità degli array

Un array i cui elementi sono di tipo array si dice anche array bidimensionale

In linea di principio è possibile anche creare array di array di array, o array di array di array di array, e così via

- noi ci limiteremo a studiare array bidimensionali

Array bidimensionali

Un array bidimensionale può essere pensato come un array con due indici

Tipicamente gli array bidimensionali sono usati per rappresentare matrici di valori di uno stesso tipo, anche se il loro potere espressivo è superiore a quello delle matrici (lo vedremo più avanti)

I due indici di un array bidimensionale si chiamano anche indice di riga e indice di colonna

Creazione di array bidimensionali

Un array bidimensionale si crea usando l'operatore *new*, in base alla seguente sintassi

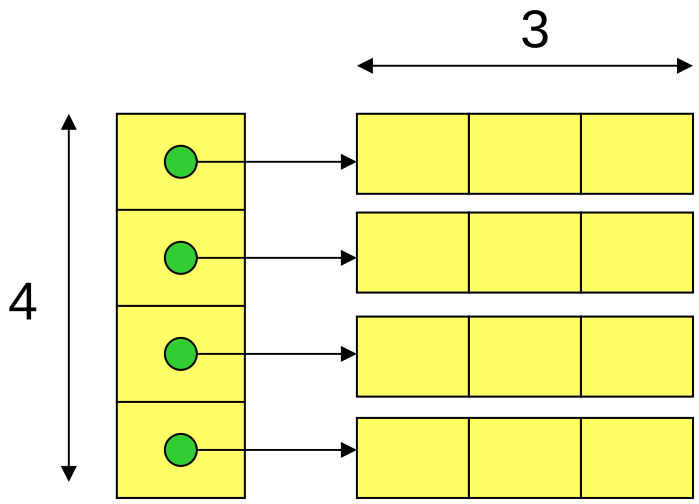
new <tipo di dato>[dimR][dimC]

Dove *<tipo di dato>* è un qualunque tipo di dato valido (ma non un tipo array), mentre *dimR* e *dimC* sono rispettivamente la dimensione dell'indice di riga e dell'indice di colonna

Creazione di array bidimensionali

Ad esempio, per creare un array bidimensionale i cui elementi sono di tipo *int*, e le cui dimensioni sono 4 (per le righe) e 3 (per le colonne), debbo scrivere

new int[4][3]



schema dell'array creato

Variabili di tipo array bidimensionale

Se voglio dichiarare una variabile che può contenere il riferimento ad un array bidimensionale di tipo *T*, debbo scrivere

T[ ][ ] <nome variabile>

Diremo che <nome variabile> è di tipo "array bidimensionale di tipo T"

Posso per esempio scrivere

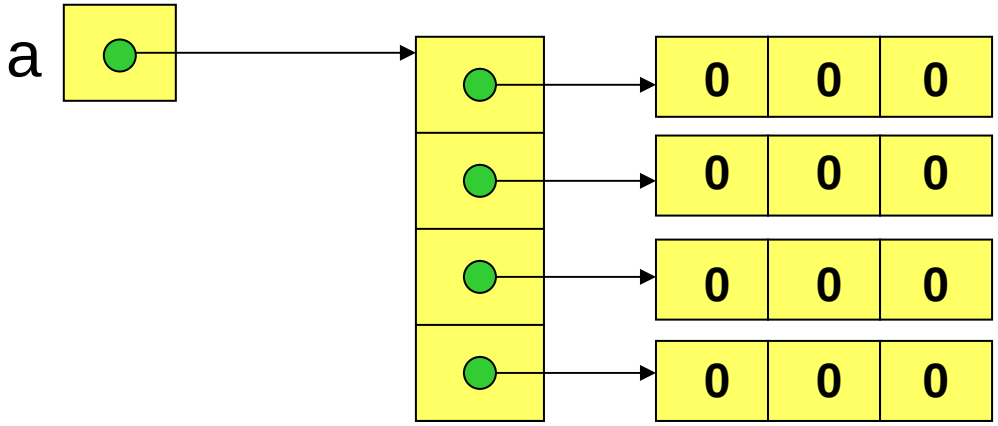
int[ ][ ] a = new int[4][3]

Variabili di tipo array bidimensionali

Se scrivo

int[ ][ ] a = new int[4][3]

la variabile *a* conterrà il riferimento ad un array bidimensionale di tipo *int*; di default, gli elementi *int* dell'array sono inizializzati a 0

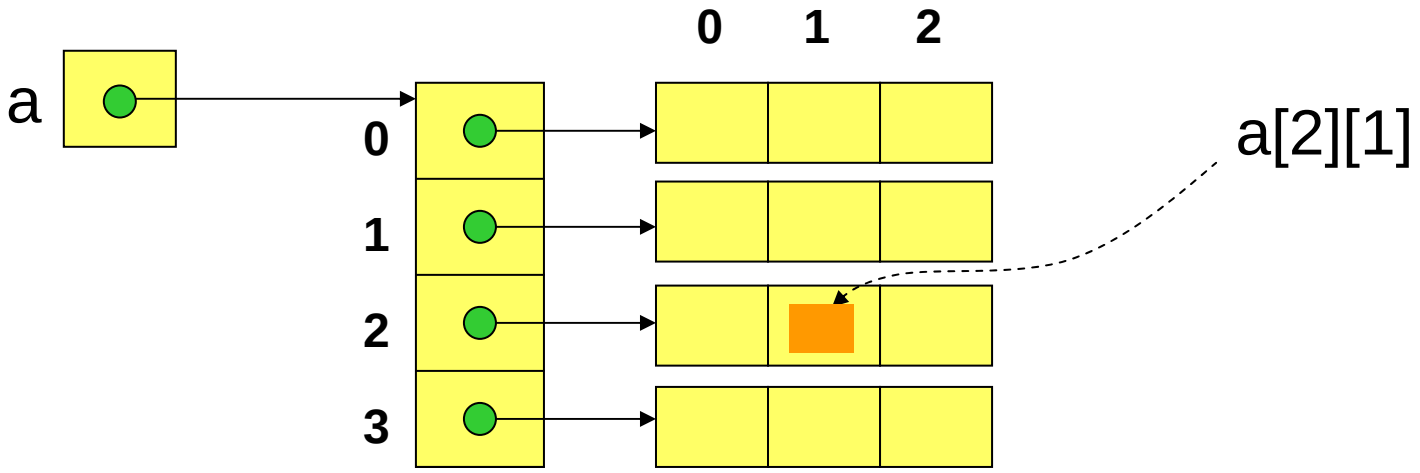


schema dell'array creato

Elementi di un array bidimensionale

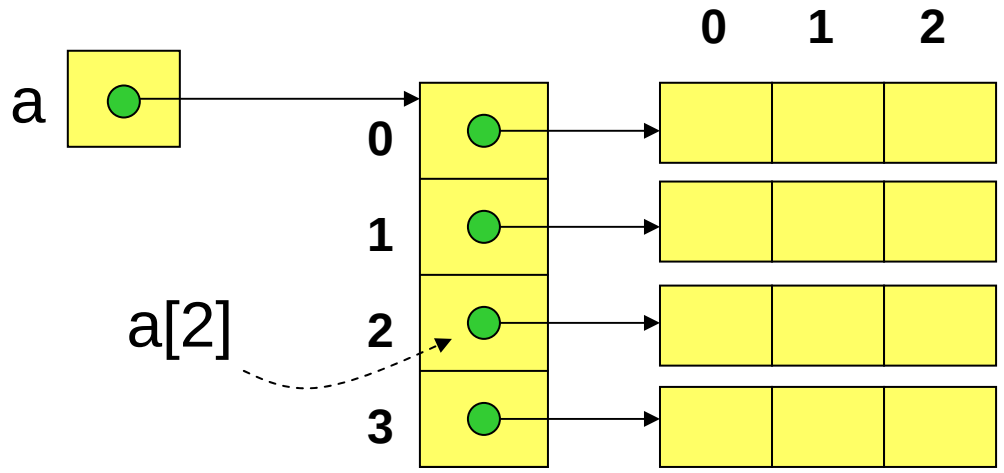
Supponiamo che *a* sia una variabile di tipo “array bidimensionale di tipo *T*”

Per denotare l’elemento di tipo *T* di *a* di indici (i,j) debbo scrivere *a[i][j]*



Elementi di un array bidimensionale

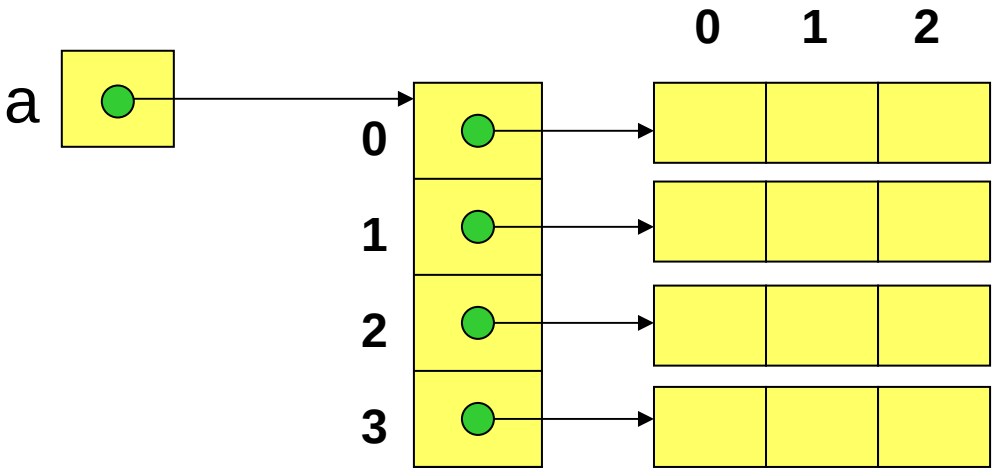
Scrivendo invece soltanto *a[i]*, sto denotando il riferimento all'array di tipo *T* contenuto alla riga i-esima



Lunghezze di array bidimensionali

Se *a* è un array bidimensionale:

- *a.length* fornisce il numero degli indici di riga
- *a[i].length* fornisce il numero degli elementi dell'array referenziato da *a[i]*



- a.length vale 4
- a[i].length vale 3 per qualunque i

Esercizio sull'uso di array bidimensionali

Scriviamo un semplice programma che fa inserire all'utente una matrice *m*×*n* di numeri interi e che visualizza all'utente la somma degli elementi di ogni riga

Ad esempio, se l'utente inserisce la matrice di seguito riportata, l'output del programma deve essere quello mostrato a fianco

12	**1**	**-5**
3	**4**	**0**
0	**10**	**7**
-2	**-4**	**4**

Somma riga 0 = 8

Somma riga 1 = 7

Somma riga 2 = 17

Somma riga 3 = -2

Schema di soluzione

```
class SommaRigheMatrice{

   public static void main (String[] args){
       InputWindow in = new InputWindow ();
       OutputWindow out = new OutputWindow ();

       /* crea una matrice di dimensioni specificate */
       ….

       /* fa inserire all'utente i valori della matrice */
              ….

       /* visualizza le somme degli elementi di ogni riga */
       ….
   }
}
```

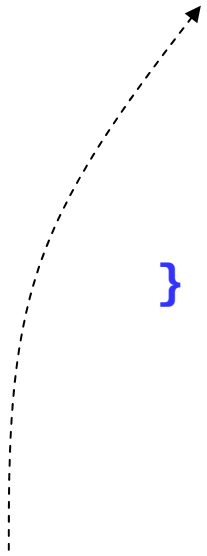
Creazione e inserimento della matrice

```
...
/* crea una matrice di dimensioni specificate */
int m = in.readInt (“Numero righe?”);
int n = in.readInt (“Numero colonne?");
int[][] matrice = new int[m][n];


/* fa inserire all'utente i valori della matrice */
for (int i=0; i<matrice.length; i++)
   for (int j=0; j<matrice[i].length; j++)
       matrice[i][j] = in.readInt (“Elemento (” + i
                          + “,” + j + “)?”);
...
```

Visualizzazione delle somme per riga

```
...
/* visualizza le somme degli elementi di ogni riga */
 int somma;
 for (int i=0; i<matrice.length; i++){
   // calcola la somma sulla riga i
   somma = 0;
   for (int j=0; j<matrice[i].length; j++)
      somma += matrice[i][j];
   // visualizza la somma sulla riga i
   out.writeln ("Somma riga " + i + " = " + somma);
 }
```



La somma deve essere riazzerata all'inizio di ogni nuovo ciclo, cioè quando si cambia riga

Esercizio sulle matrici

Vogliamo ora definire il codice di una classe (la classe *Matrice*) i cui oggetti rappresentano matrici rettangolari di numeri reali

Un oggetto della classe *Matrice* ci permetterà di effettuare alcune operazioni tipiche tra matrici

La classe Matrice - scheletro

```
class Matrice{

        // variabile che memorizza la matrice rappresentata

        private double[][] mat;

        /* costruttore: crea un oggetto che rappresenta una
           matrice copia di quella passata come parametro */
        public Matrice (double[][] matrice){…}

        /* ritorna la Matrice somma della Matrice
           ricevente (this) con il parametro (altra)
           PRE: this ed altra hanno le stesse dimensioni */
        public Matrice somma (Matrice altra){…}

        /* ritorna la Matrice prodotto della Matrice
           ricevente (this) con il parametro (altra)
           PRE: num. col. di this = num. righe di altra */
        public Matrice prodotto (Matrice altra){…}

        /* ritorna una descrizione della matrice */
        public String toString (){…}
}
```

Costruttore

```
public Matrice (double[][] matrice){
   int m = matrice.length;
   int n = matrice[0].length;
   this.mat = new double[m][n];
   for (int i=0; i<m; i++)
       for (int j=0; j<n; j++)
             this.mat[i][j]=matrice[i][j];
}
```

Il costruttore crea una nuova matrice equivalente a quella passata come parametro. In questo modo si evitano side-effect

Il metodo somma

```
public Matrice somma (Matrice altra){
   int m = this.mat.length;
   int n = this.mat[0].length;
   double[][] matrice = new double[m][n];
   for (int i=0; i<m; i++)
     for (int j=0; j<n; j++)
         matrice[i][j] = this.mat[i][j]+altra.mat[i][j];
   Matrice somma = new Matrice (matrice);
   return somma;
}
```

Viene creato un nuovo oggetto *Matrice* che incarta una matrice somma tra *this.mat* e *altra.mat*

Il metodo prodotto

```
public Matrice prodotto (Matrice altra){
  int m = this.mat.length;     // righe di this
  int n = this.mat[0].length;  // colonne di this
  int p = altra.mat[0].length; // colonne di altra
  double[][] matrice = new double[m][p];
  for (int i=0; i<m; i++){
       for (int j=0; j<p; j++){
          matrice[i][j] = Σ_{k={0,..,n-1}} this.mat[i][k] * altra.mat[k][j]
       }
  }
  Matrice prodotto = new Matrice(matrice);
  return prodotto;
}
```

Il metodo prodotto

```
public Matrice prodotto (Matrice altra){
  int m = this.mat.length;     // righe di this
  int n = this.mat[0].length;  // colonne di this
  int p = altra.mat[0].length; // colonne di altra
  double[][] matrice = new double[m][p];
  for (int i=0; i<m; i++){
       for (int j=0; j<p; j++){
          double somma = 0;
          for (int k=0; k<n; k++)
              somma += this.mat[i][k]*altra.mat[k][j];
          matrice[i][j]=somma;
       }
  }
  Matrice prodotto = new Matrice(matrice);
  return prodotto;
}
```

Il metodo toString

```
public String toString (){
   String s = "";
   for (int i=0; i<this.mat.length; i++){
       for (int j=0; j<this.mat[i].length; j++)
              s += this.mat[i][j] + "\t";
       s += "\n";
       }
   return s;
}
```

I valori sulla stessa linea sono separati da un carattere di tabulazione (*'\t'*)

Esercizio

Scrivere una classe di prova che verifica il corretto funzionamento della classe *Matrice*

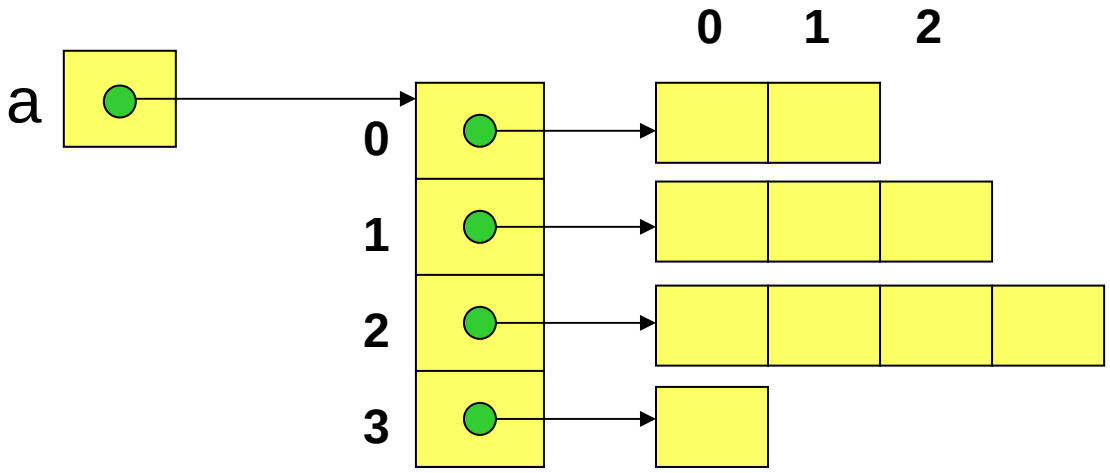
Il metodo *main* della classe di prova deve:

1. far inserire all'utente tre matrici A, B, C, tali che le dimensioni di A e B coincidono, mentre C ha il numero di righe pari al numero di colonne di A;
2. visualizzare all'utente una descrizione delle tre matrici inserite
3. visualizzare all'utente la matrice A+B e la matrice AxB

Array bidimensionali non matriciali

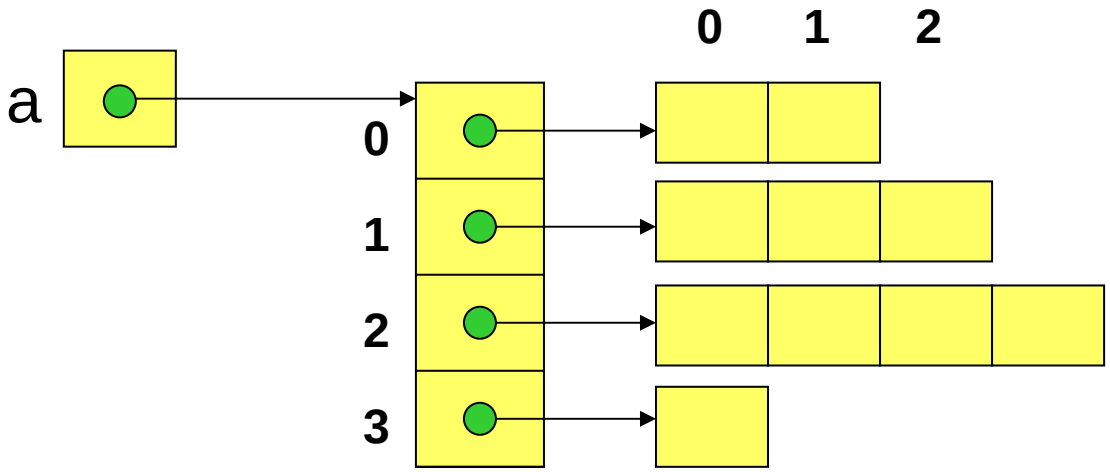
Come accennato, il potere di rappresentazione di un array bidimensionale è superiore a quello di una forma matriciale

- le righe di un array di array possono avere dimensioni diverse

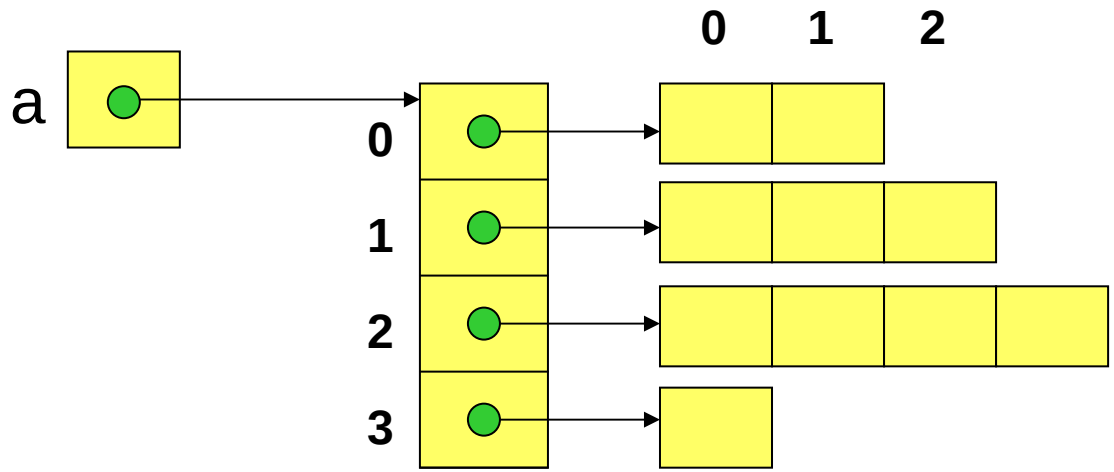


Definizione di un array non matriciale

Supponiamo ad esempio di voler definire il seguente array bidimensionale di tipo *int*



Definizione di un array non matriciale




```
int[ ][ ] a = new int[3][ ];
a[0] = new int[2];
a[1] = new int[3];
a[2] = new int[4];
a[3] = new int[1];
```

Letterali array

Gli array (e gli array di array) si possono anche creare ed inizializzare attraverso l'uso di valori costanti, detti letterali array

- Ad esempio, la seguente istruzione crea un array di dimensione 5, dove a[0] vale 10, a[1] vale 20, ecc.

```
int[ ] a = {10,20,10,5,0};
```

- Ad esempio, la seguente istruzione crea ed inizializza un array di array di *String* di dimensioni 3 x 2

```
String[ ] [ ] a = { {"aaa", "abc", "abb"},
                    {"bcd", "bdb", "ccc"} } ;
```

Glossario dei termini principali

Termine	Significato
Array di array (array bidimensionali)	Array i cui elementi sono essi stessi di tipo array
Indici di riga/colonna di un array di array	Gli indici interi associati alle variabili dell'array di array
Letterali array	Valori array costanti. Un array si può creare ed inizializzare contestualmente usando letterali array