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

Monica Nesi

Università degli Studi dell'Aquila

16 Novembre 2022

Metodi statici che restituiscono array

Finora abbiamo considerato la definizione di metodi statici che possono avere array monodimensionali o bidimensionali tra i loro parametri formali.

Ora vediamo la definizione di metodi statici che *restituiscono* (un riferimento ad) un array monodimensionale o bidimensionale.

Esempio 1 (in cui si restituisce un array monodimensionale): scrivere un metodo che, dato un array monodimensionale di stringhe `a`, restituisce un array monodimensionale di interi `b` tale che l'elemento `b[i]` è la lunghezza della stringa `a[i]`.

Ad esempio, se `a = {"abc","","fddgf","bb"}`, allora il metodo deve restituire l'array `{3,0,5,2}`.

Metodi statici che restituiscono array (cont.)

Esempio 2 (in cui si restituisce un array bidimensionale):
scrivere un metodo che, dato un array monodimensionale di stringhe a, restituisce un array bidimensionale di caratteri c tale che la riga `c[i]` contiene i caratteri della stringa `a[i]` (letta da sx a dx).

Ad esempio, dato ancora l'array `a = {"abc","","fddgf","bb"}`, il metodo deve restituire l'array
`{{'a','b','c'},{},{'f','d','d','g','f'},{'b','b'}}`.

N.B. L'array da restituire *non* è un parametro formale del metodo.

Tale array deve essere
- dichiarato e creato all'inizio del corpo del metodo,
- ai suoi elementi devono essere assegnati dei valori secondo la specifica del problema,
- infine va restituito il riferimento a tale array.

Esempio 1

Scrivere un metodo che, dato un array monodimensionale di stringhe `a`, restituisce un array monodimensionale di interi `b` tale che l'elemento `b[i]` è la lunghezza della stringa `a[i]`.

Input: `String[] a`
Output: `int[] b` t.c. `b[i] = a[i].length()`

Scriviamo l'intestazione del metodo:

```
public static int[] lunghezze (String[] a)
```

Nel corpo del metodo si dichiara e crea l'array `b`. Occorre capire dalla specifica del metodo quanto deve essere lungo tale array:

```
int[] b = new int[a.length];
```

Esempio 1 (cont.)

Una volta creato l'array `b` (i cui elementi sono inizializzati al valore di default 0), occorre *riempirlo* con i valori che soddisfano la specifica del metodo.

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

Infine, viene restituito il riferimento all'array `b`:

```
return b;
```

Il tipo di `b` è `int[]`, ovvero è il tipo del risultato che compare nell'intestazione del metodo.

Esempio 1: metodo

Ricapitolando, una definizione per il metodo richiesto è la seguente:

```
public static int[] lunghezze (String[] a) {
  int[] b = new int[a.length];
  for (int i=0; i<b.length; i++)
    b[i] = a[i].length();
  return b;
}
```

Assumiamo che tale metodo sia nella classe `ArrayReturn`.

Un semplice metodo `main` per testare il metodo sull'array
`a = {"abc","","fddgf","bb"}`, può essere definito come segue:

Una classe di test

```
public class ArrayReturnTest {
  public static void main (String[] args) {

    String[] a = {"abc","","fddgf","bb"};
    int[] b = ArrayReturn.lunghezze(a);

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

Dopo la compilazione con `javac ArrayReturnTest.java`, si ha:

```
java ArrayReturnTest
3 0 5 2
```

Esempio 2

Scrivere un metodo che, dato un array monodimensionale di stringhe a, restituisce un array bidimensionale di caratteri c tale che la riga `c[i]` contiene i caratteri della stringa `a[i]` (da sx a dx).

Input: `String[] a`
Output: `char[][] c` t.c. `c[i]` contiene i caratteri di `a[i]`

Scriviamo l'intestazione del metodo:

```
public static char[][] caratteri (String[] a)
```

Nel corpo del metodo si dichiara e crea l'array bidimensionale c, di cui *va specificata la prima dimensione* (i.e., il numero di righe di c), che coincide con il numero di stringhe in a:

```
char[][] c = new char[a.length][];
```

La seconda dimensione *non è specificata*, in quanto dipende dalla lunghezza della stringa `a[i]`, che *ancora non è nota*.

Esempio 2 (cont.)

Una volta creato l'array c (i cui elementi `c[i]` sono inizializzati al valore di default `null`, in quanto sono array monodimensionali), occorre *creare* le righe di c lunghe tanto quanto il numero di caratteri nella stringa `a[i]` e poi *riempire* tali righe con i valori che soddisfano la specifica del metodo.

Possiamo implementare ciò attraverso un ciclo che esamina le righe ed un ciclo annidato che, fissata la riga, la scorre per assegnare i caratteri richiesti:

```
for (int i=0; i<c.length; i++) {
  c[i] = new char[a[i].length()];
  for (int j=0; j<c[i].length; j++)
    c[i][j] = a[i].charAt(j);
}
```

Infine, viene restituito il riferimento all'array c:

```
return c;
```

Esempio 2: metodo

Ricapitolando, una definizione per il metodo richiesto è la seguente:

```
public static char[][] caratteri (String[] a) {
  char[][] c = new char[a.length][];
  for (int i=0; i<c.length; i++) {
    c[i] = new char[a[i].length()];
    for (int j=0; j<c[i].length; j++)
      c[i][j] = a[i].charAt(j);
  }
  return c;
}
```

Assumiamo che anche tale metodo sia nella classe `ArrayReturn`.

Per testare il metodo sull'array `a = {"abc","","fddgf","bb"}`, basta scrivere un metodo `main` con la chiamata

```
char[][] c = ArrayReturn.caratteri(a);
```

(con c locale al `main`) e poi stampare i valori di c.

Altri esempi: *rovesciare* un array

Scrivere un metodo che, dato un array monodimensionale di interi a, restituisce un *nuovo* array monodimensionale di interi con gli elementi di a rovesciati. Ad esempio, dato `a = {5,-7,3,12,1}`, il metodo restituisce l'array `{1,12,3,-7,5}`.

Input: `int[] a`
Output: `int[] b` con gli elementi di a rovesciati

```
public static int[] reverse (int[] a) {
  int[] b = new int[a.length];
  for (int i=0; i<a.length; i++)
    b[i] = a[a.length-1-i];
  return b;
}
```

N.B. Il reverse non viene fatto *in loco*, ovvero sull'array a in input. Qui non si vuole modificare il parametro formale a, ma si genera un *nuovo* array che soddisfa la specifica data.

Attaccare due array

Scrivere un metodo che, dati due array monodimensionali di interi `a` e `b`, restituisce un nuovo array monodimensionale, in cui `b` è stato attaccato alla fine di `a`.
Ad esempio, dati `a = {7,-3,2}` e `b = {-3,4,11,-21}`, il metodo restituisce l'array `{7,-3,2,-3,4,11,-21}`.

Input: `int[] a`, `int[] b`
Output: `int[] c` dato dagli elementi di `a` seguiti da quelli di `b`

```
public static int[] append(int[] a, int[] b) {
  int[] c = new int[a.length + b.length];
  for (int i=0; i<a.length; i++)
   c[i] = a[i];
  for (int i=0; i<b.length; i++)
   c[a.length+i] = b[i];
  return c;
}
```

Generare i prefissi di una stringa

Scrivere un metodo che, data una stringa `s`, restituisce un array monodimensionale di stringhe contenente tutti i *prefissi* di `s` in *ordine crescente* (inclusa `s`).
Ad esempio, data la stringa `"abcdef"`, il metodo restituisce l'array `{"a","ab","abc","abcd","abcde","abcdef"}`.

Input: `String s`
Output: `String[] a` contenente i prefissi di `s`

```
public static String[] prefissi(String s) {
  int n = s.length();
  String[] a = new String[n];
  for (int i=0; i<n; i++) {
    a[i] = s.substring(0,i+1);
  }
  return a;
}
```

Le lunghezze di un array bidim. di stringhe

Scrivere un metodo che, dato un array bidimensionale di stringhe `a`, restituisce un array bidimensionale di interi `b` contenente le lunghezze degli elementi di `a` nelle posizioni corrispondenti.
Ad esempio, se `a` è l'array

```
{{"abcd","ab","kzz"},
 {"cde","","hkkhh","a"},
 {"pprs","lp"}}
```

il metodo restituisce l'array `b = {{4,2,3},{3,0,5,1},{4,2}}`.

Input: `String[][] a`
Output: `int[][] b` t.c. `b[i][j] = a[i][j].length()`

Si tratta dell'estensione al caso bidimensionale dell'Esempio 1.

Le lunghezze di un array bidim. di stringhe: metodo

Quindi, è possibile dare una definizione che sfrutta il metodo `lunghezze` da invocare sulle righe `a[i]`, oppure una definizione con due cicli annidati:

```
public static int[][] lunghezzeBi(String[][] a) {
  int[][] b = new int[a.length][];
  for (int i=0; i<a.length; i++) {
    b[i] = new int[a[i].length];
    for (int j=0; j<a[i].length; j++) {
      b[i][j] = a[i][j].length();
    }
  }
  return b;
}
```

Triangoli di una matrice quadrata

Scrivere un metodo che, dato un array bidimensionale *quadrato* di caratteri a, restituisce un array bidimensionale di caratteri che rappresenta il triangolo in alto a sinistra inclusa la diagonale. *Quadrato* significa che ogni riga ha lo stesso numero di elementi n (i.e., è una matrice) e questo coincide con il numero di righe (i.e., è una matrice quadrata $n \times n$).

Ad esempio, dato l'array (matrice quadrata 3×3)

```
{{'a','b','c'},
 {'d','e','f'},
 {'g','h','i'}}
```

il metodo restituisce l'array

```
{{'a','b','c'},
 {'d','e'},
 {'g'}}
```

Triangoli di una matrice quadrata: metodo

Input: `char[][] a`
Output: `char[][] b` uguale al triangolo in alto a sinistra inclusa la diagonale

```
public static char[][] triangoloA(char[][] a) {
  char[][] b = new char[a.length][];
  for (int i=0; i<a.length; i++) {
    b[i] = new char[a[i].length-i];
    for (int j=0; j<a[i].length-i; j++) {
      b[i][j] = a[i][j];
    }
  }
  return b;
}
```

In modo simile possono essere definiti i metodi che restituiscono i triangoli in alto/basso a sx/dx inclusa la diagonale.

Esercizio

Esercizio 1 (Secondo Parziale - 30 Novembre 2018)

Scrivere un metodo che, dati un array bidimensionale di stringhe a ed un array monodimensionale di caratteri c, restituisce un array monodimensionale di stringhe b tale che $b[i]$ è la *prima* stringa nella riga $a[i]$ (letta da sx a dx) il cui primo carattere è diverso da $c[i]$. Se in $a[i]$ non esiste alcuna stringa che soddisfa tale condizione, allora $b[i]$ è `"::"`.

Esempio: se $a =$ `{{"dh","stk"},{"jm","qsq","yw"},{"grt","gw","g","gpw"}}` e $c =$ `{'d','j','g'}`,

il metodo restituisce l'array $b =$ `{"stk","qsq","::"}`.

Si assuma che gli array a e c abbiano la stessa lunghezza e che tutte le stringhe in a siano non vuote.