Creazione array in Java

Questa breve dispensa riassume alcuni costrutti che si possono utilizzare per creare vari tipi di array. Faremo gli esempi solo su array di interi e array di stringhe. Tutto quello che diremo sugli array di interi si estende anche agli array degli altri tipi primitivi (short, double, etc…) mentre quello che diremo sugli array di stringhe si estende agli array di classi.

Array monodimensionali

Array nulli

Un array nullo in realtà non è un array. Il valore `null`, come abbiamo detto a lezione, è un valore speciale per tutte le variabili di tipo riferimento (array, stringhe, classi, etc…) che indica la mancanza di un valore. Queste istruzioni creano una variabile array e la inizializzano con il valore nullo.

```
int[] a = null;
int[] a;
String[] a = null;
String[] a;
```

Le seguenti operazioni su un array nullo falliscono:

- `a.length` fallisce con un errore `NullPointerException`
- `a[i]` fallisce con un errore `NullPointerException`

Array vuoti

Un array vuoto è un array vero e proprio, ma che non contiene alcun elemento, perché la sua lunghezza è zero. Queste istruzioni creano un variabile array e la inizializzano con un array vuoto.

```
int[] a = new int[0];
int[] a = new int[0] { };
int[] a = { };
String[] a = new String[0];
String[] a = new String[] { };
String[] a = { };
```

Un array vuoto è un array in piena regola ma, visto che è vuoto, non è possibile usare le parentesi quadre per accedere ai suoi elementi.

- `a.length` restituisce 0
- `a[i]` restituisce un errore `ArrayIndexOutOfBoundsException` qualunque sia `i`.

Array pieni di zero / null

È possibile creare facilmente array lunghi a piacere e pieni di zero (per gli interi) o `null` (per le stringhe). Le seguenti istruzioni creano array di lunghezza 5.

```
int[] a = new int[5];
String[] a = new String[5];
```

Nel caso degli interi, l'array è riempito di zeri `{ 0, 0, 0, 0, 0 }`, mentre nel caso di stringhe è riempito di valori nulli `{ null, null, null, null, null }`.

- `a.length` restituisce 5
- `a[i]` restituisce 0/`null` se $0 \leq i < 5$, altrimenti fallisce con `ArrayIndexOutOfBoundsException`

Array pre-riempiti con valori a scelta

Se al momento in cui scrivo un programma so già che valori dovrà contenere un array, posso usare una sintassi speciale per creare un array già inizializzato con questi valori. Ad esempio:

```
int[] a = { 1, 2, 3 };
int[] a = new int[] { 1, 2, 3 };
String[] a = { "Ciao", "sono", "io" };
String[] a = new String[] { "Ciao", "sono", "io" };
```

In queso caso:

- `a.length` restituisce 3
- `a[i]` restituisce il valore dell'i-esimo elemento di `a`, purché sia $0 \leq i < 3$, altrimenti genera un errore `ArrayIndexOutOfBoundsException`

C'è una importante differenza tra la notazione `{ 1, 2, 3 }` e la notazione `new int[] { 1, 2, 3 }`. La prima si può usare solo durante la dichiarazione di una variabile, mentre la seconsa si può usare ovunque sia ammessa una espressione di tipo array.

Ad esempio:

```
System.out.println(Arrays.toString(new int[] { 1, 2, 3 }));
a = new int[] { 1, 2, 3 };
```

funzionano (se a è una variabile il cui tipo è `int[]`), mentre

```
System.out.println(Arrays.toString({ 1, 2, 3 }));
a = { 1, 2, 3 };
```

non funzionano ma generano un errrore in fase di compilazione.

Array bidimensionali

Array nulli

Tutto come per gli array monodimensionali:

```
int[][] a = null;
int[][] a;
String[][] a = null;
String[][] a;
```

Le seguenti operazioni su un array nullo falliscono:

- `a.length` fallisce con un errore `NullPointerException`
- `a[i]` fallisce con un errore `NullPointerException`
- `a[i].length` fallisce con un errore `NullPointerException`
- `a[i][j]` fallisce con un errore `NullPointerException`

Array bidimensionali creati parzialmente

Ricordate che in realtà gli array bidimensionali sono array il cui contenuto è a sua volta un array. Un array creato parzialmente è un array in cui l'array esterno (quello contenente le righe) è stato creato, ma gli array interni no.

Ad esempio, le seguenti istruzioni generano un array creato parzialmente di 3 righe;

```
int[][] a = new int[3][];
int[][] a = new int[][] { null, null, null };
int[][] a = { null, null, null };
String[][] a = new String[3][];
String[][] a = new String[][] { null, null, null };
String[][] a = { null, null, null };
```

Questo è quello che succede quanto proviamo ad accedere ad `a`.

- `a.length` restituisce 3
- `a[i]` restituisce `null` se $0 \leq i < 3$, altrimenti genera `ArrayIndexOutOfBoundsException`
- `a[i].length` fallisce con un errore `NullPointerException`
- `a[i][j]` fallisce con un errore `NullPointerException`

Le righe di un array creato in questo modo posson poi essere create successivamente. Ad esempio,

```
a[0] = new int[] { 2, 3, 5 };
```

crea la prima riga (di indice 0) di `a` e la riempie con i numeri 2, 3 e 5.

Ovviamente, il numero di righe dell'array creato parzialmene può essere zero. In qusto caso si ha un array bidimensionale vuoto.

Array bidimensionali riempiti di zeri/null

Le seguenti istruzioni creano un array "rettangolare", in cui tutte le righe hanno lo stesso numero di elementi. L'array può essere considerato come una matrice, con un certo numero di righe e di colonne. Tutti gli elementi sono inizializzati con zero (per gli interi) o `null` (per le stringhe) Ad esempio, le seguenti istruzioni creano array di 3 righe e 4 colonne.

```
int[][] a = new int[3][4];
String[][] a = new String[3][4];
```

Nel caso degli interi, a è l'array

```
{
    {0, 0, 0, 0},
    {0, 0, 0, 0},
    {0, 0, 0, 0}
}
```

Questo è quello che succede quanto proviamo ad accedere ad `a`.

- `a.length` restituisce 3
- `a[i]` restituisce un array di 4 elementi riempito di 0/`null` se $0 \leq i < 3$, altrimenti fallisce con `ArrayIndexOutOfBoundsException`
- `a[i].length` restituisce 4 se se $0 \leq i < 3$, altrimenti fallisce con `ArrayIndexOutOfBoundsException`
- `a[i][j]` restituisce 0/`null` se $0 \leq i < 3$ e $0 \leq j < 4$, altrimenti fallisce con `ArrayIndexOutOfBoundsException`

Array pre-riempiti con valori a scelta

Se al momento in cui scrivo un programma so già che valori dovrà contenere un array, posso crearli già riempiti come segue:

```
int[] a = { { 1, 2, 3 }, { 4, 5 };
int[] a = new int[][] { { 1, 2, 3 }, { 4, 5 } };
String[] a = { { “Ciao” }, { “sono”, “io” } };
String[] a = new String[][] { { “Ciao” }, { “sono”, “io” } };
```