Azzolini Riccardo 2018-10-30

Array

1 Array

Un **array** è un insieme ordinato di variabili dello *stesso tipo* (detto **tipo base**), ognuna accessibile specificando la posizione in cui si trova.

Il tipo base può essere primitivo o riferimento (*array di oggetti*):

- se è primitivo, ogni posizione dell'array contiene direttamente un valore
- se è riferimento, ogni posizione contiene appunto un riferimento a un oggetto

2 Variabili e costruzione

Gli array sono oggetti, quindi i tipi array sono tipi riferimento.

Una variabile di tipo array si dichiara con la sintassi:

```
tipo_base[] identificatore;
```

La costruzione di un oggetto array si effettua con l'operatore `new`:

```
new tipo_base[espressione_int]
```

- È un'espressione con
  - tipo: `tipo_base[]`
  - valore: riferimento all'oggetto array creato
- `espressione_int` specifica la dimensione dell'array.
- Le posizioni dell'array sono inizializzate a
  - `null`, se il tipo base è un tipo riferimento
  - un valore di default per i tipi primitivi (`0`, `false`, ecc.)

3 Lunghezza

Una volta creato un array, la sua lunghezza è fissa: non si possono aggiungere o rimuovere posizioni.

Oltre agli elementi, ogni oggetto array contiene la sua lunghezza all'interno del campo `length`, di tipo `int`.

Esempio:

```
frazioni.length // 4
```

4 Accesso agli elementi

`nome_array[selettore]`

- È una *variabile* con
  - tipo: il tipo base dell'array
  - valore: il contenuto della posizione corrispondente dell'array
- Il `selettore` dev'essere un'espressione di tipo `int` (oppure `byte`, `short` o `char`, che vengono automaticamente promossi a `int`).
- Le posizioni sono contate a partire da 0.
- Il tentativo di accesso a una posizione non esistente dell'array provoca un errore in fase di esecuzione.

4.1 Esempio

```
Frazione[] frazioni = new Frazione[4];

frazioni[0] = new Frazione(1, 4);
frazioni[1] = new Frazione(2, 4);
frazioni[2] = new Frazione(3, 4);

int i = 2;
frazioni[2 * i - 1] = frazioni[2 * i - 2].piu(frazioni[1]);
// frazioni[3] <- 4/4

frazioni[4] = new Frazione(5, 4);
// ArrayIndexOutOfBoundsException
```

5 Array e cicli `for`

```
for (int i = 0; i < nome_array.length; i++) {
    ... nome_array[i] ...
}
```

5.1 Ciclo *for-each*

Esiste una forma di ciclo `for` apposita per l'iterazione di array, chiamata *for-each*:

```
for (tipo_base identificatore : array)
    istruzione
```

- Consente di ottenere uno dopo l'altro i valori contenuti nell'array, senza dover gestire manualmente la variabile indice e l'accesso.
- Non permette di accedere alle posizioni dell'array per modificarne i contenuti.

5.2 Esempio

```
Frazione[] frazioni = new Frazione[4];

for (int i = 0; i < frazioni.length; i++)
    frazioni[i] = new Frazione(i + 1, 4);

for (Frazione f : frazioni)
    out.println(f.toString());
```

6 Inizializzazione esplicita

È possibile inizializzare un array direttamente in fase di dichiarazione, specificando gli elementi tra parentesi graffe:

```
tipo_base[] nome_array = { elemento_0, elemento_1, ..., elemento_n };
```

La dimensione viene dedotta automaticamente dal compilatore.

6.1 Esempio

```
Frazione[] frazioni = {
    new Frazione(1, 4),
    new Frazione(2, 4),
    new Frazione(3, 4),
    new Frazione(4, 4)
};
```