

# Array

---

- Sequenze ordinate di
    - Tipi primitivi (int, float, etc.)
    - Riferimenti ad oggetti (vedere classi!)
  - Elementi dello stesso tipo
    - Indirizzati da indici
    - Raggiungibili con l'operatore di indicizzazione: le **parentesi quadre** []
    - Raggruppati sotto lo stesso nome
- 

## In Java gli array sono Oggetti

- Sono allocati nell'area di memoria riservata agli oggetti creati dinamicamente (heap)

### Dimensione

- Può essere stabilita a run-time (quando l'oggetto viene creato)
  - È fissa (non può essere modificata)
  - E' nota e ricavabile per ogni array
- 

## Array Mono-dimensionali (vettori)

### Dichiarazione di un riferimento a un array

- `int[] voti;`
- `int voti[];`

La dichiarazione di un array non assegna alcuno spazio

```
voti == null
```

---

## Creazione di un Array

### L'operatore new crea un array:

- Con costante numerica

```
int[] voti;  
...  
voti = new int[10];
```

- Con costante simbolica
-

```
final int ARRAY_SIZE = 10;
int[] voti;
...
voti = new int[ARRAY_SIZE];
```

- 
- Con valore definito a run-time

```
int[] voti;
... definizione di x (run-time) ...
voti = new int[x];
```

---

*\*Utilizzando un inizializzatore- (che permette anche di riempire l'array)*

- L'operatore new inizializza le variabili
  - 0 - per variabili di tipo numerico (inclusi i char)
  - false - per le variabili di tipo boolean

```
int[] primi = {2,3,5,7,11,13};
...
int [] pari = {0, 2, 4, 6, 8, 10,};
// La virgola finale e' facoltativa
// (elenchi lunghi)
```

- 
- Dichiarazione e creazione possono avvenire contestualmente
  - L'attributo length indica la lunghezza dell'array, cioè il numero di elementi
  - Gli elementi vanno da 0 a length-1

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

---

## In Java viene fatto il bounds checking

- Maggior sicurezza
- Maggior lentezza di accesso

## Il riferimento ad array

- Non è un puntatore al primo elemento
- È un puntatore all'oggetto array
- Incrementandolo non si ottiene il secondo elemento

## Array di oggetti

Per gli array di oggetti (e.g., Integer) `Integer [] voti = new Integer [5];` ogni elemento e' un riferimento

**L'inizializzazione va completata con quella dei singoli elementi**

```
voti[0] = new Integer (1);  
voti[1] = new Integer (2);  
...  
voti[4] = new Integer (5);
```

---

## Array Multi-dimensionali (Matrici)

**Array contenenti riferimenti ad altri array**

Sintatticamente sono estensioni degli array a una dimensione

**Sono possibili righe di lunghezza diverse** (matrice = array di array)

```
int[][] triangle = new int[3][]
```

---

**Le righe non sono memorizzate in posizioni adiacenti**

- Possono essere spostate facilmente

```
// Scambio di due righe  
double[][] saldo = new double[5][6];  
...  
double[] temp = saldo[i];  
saldo[i] = saldo[j];  
saldo[j] = temp;
```

- L'array è una struttura dati efficiente ogni volta che il numero di elementi è noto
- Il ridimensionamento di un array in Java risulta poco efficiente
- Utilizzare altre strutture dati se il numero di elementi contenuto non è noto

---

**Il pacchetto java.util contiene metodi statici di utilità per gli array**

- Copia di un valore in tutti gli (o alcuni) elementi di un array
  - `Arrays.fill (<array>, <value>);`
  - `Arrays.fill (<array>, <from>, <to>, <value>);`
- Copia di array

- `System.arraycopy (<arraySrc>, <offsetSrc>,<arrayDst>, <offsetDst>,<#elements>);`
  - Confronta due array
    - `Arrays.equals (<array1>, <array2>);`
  - Ordina un array (di oggetti che implementino l'interfaccia Comparable)
    - `Arrays.sort (<array>);`
  - Ricerca binaria (o dicotomica)
    - `Arrays.binarySearch (<array>);`
- 

## Esempi di Array

### Array Monodimensionali

```
int[] list = new int[10];

list.length;

int[] list = {1, 2, 3, 4};
```

---

### Array Multidimensionali

```
int[][] list = new int[10][10];
list.length;
list[0].length;
int[][] list = {{1, 2}, {3, 4}};
```

---

### Array irregolari

```
int[][] m = {
    {1, 2, 3, 4},
    {1, 2, 3},
    {1, 2},
    {1}
};
```

---

[esempi ed esercizi su array](#)