Capitolo 5

Array e collezioni

Sommario: Array e collezioni

Array

Array

Insieme ordinato di variabili dello stesso tipo (tipo base), ognuna delle quali è accessibile specificando la posizione in cui si trova.

Array

Array

Insieme ordinato di variabili dello stesso tipo (tipo base), ognuna delle quali è accessibile specificando la posizione in cui si trova.

- Tipo base
  Può essere sia un tipo primitivo sia un tipo riferimento

Array

Array

Insieme ordinato di variabili dello stesso tipo (tipo base), ognuna delle quali è accessibile specificando la posizione in cui si trova.

- Tipo base
  Può essere sia un tipo primitivo sia un tipo riferimento
- Array di oggetti
  Array il cui tipo base è un tipo riferimento

Costruzione di array

In Java gli array sono oggetti

Costruzione di array

In Java gli array sono oggetti

Costruzione di un array

`new` *tipo_base*`[`*espressione_int*`]`

Costruzione di array

In Java gli array sono oggetti

Costruzione di un array

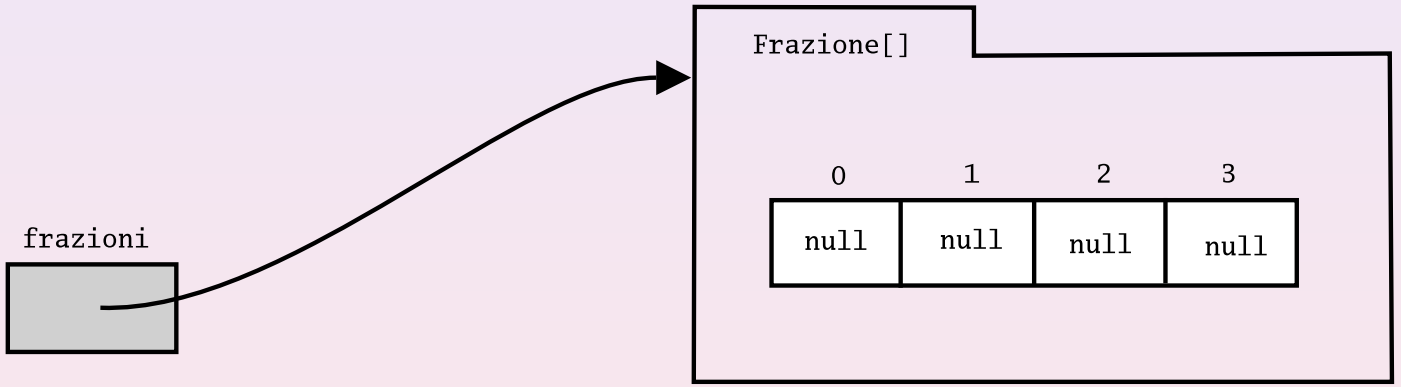
`new` *tipo_base*`[`*espressione_int*`]`

Dichiarazione di variabile

Tipo_base`[]` *identificatore*;

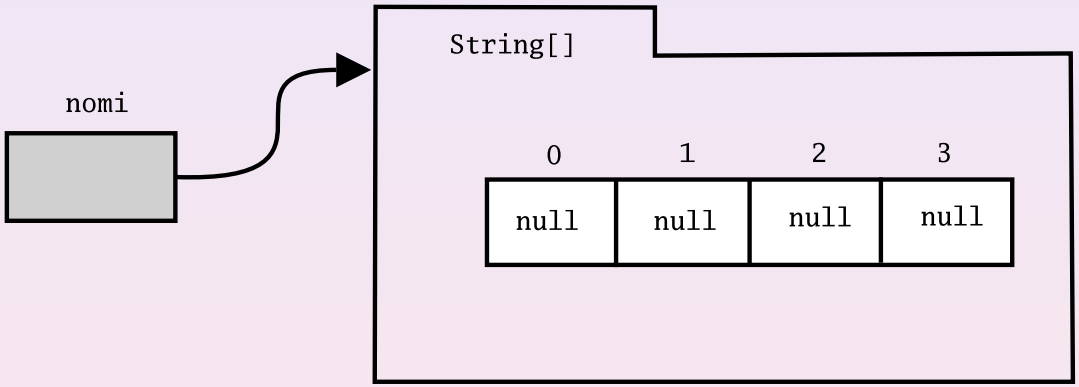
Esempio

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



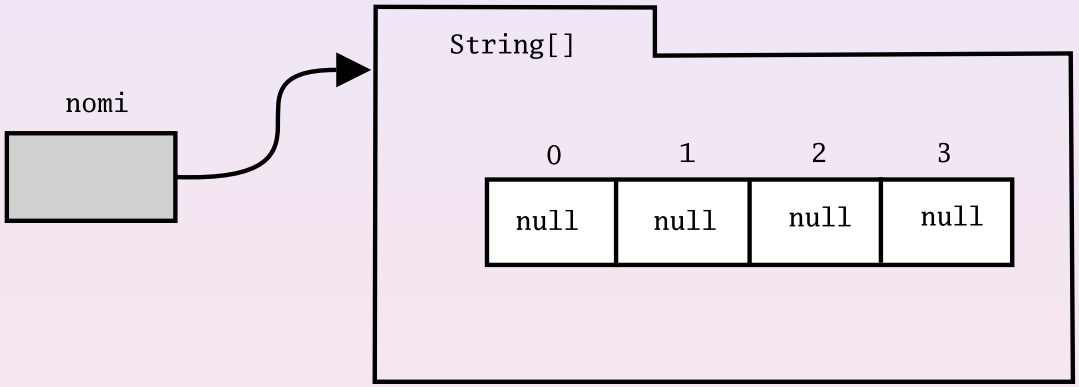
Esempio

```
String[] nomi = new String[4]
```



Esempio

```
String[] nomi = new String[4]
```



In un array di oggetti le posizioni sono automaticamente inizializzate a `null` all'atto della creazione

Lunghezza di un array

- Ogni oggetto di tipo array ha memorizzata nel suo stato l'informazione relativa alla sua lunghezza

Lunghezza di un array

- Ogni oggetto di tipo array ha memorizzata nel suo stato l'informazione relativa alla sua lunghezza
- Tale informazione si trova in un campo di nome `length` e di tipo `int`

Lunghezza di un array

- Ogni oggetto di tipo array ha memorizzata nel suo stato l'informazione relativa alla sua lunghezza
- Tale informazione si trova in un campo di nome `length` e di tipo `int`

Esempio

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

Lunghezza di un array

- Ogni oggetto di tipo array ha memorizzata nel suo stato l'informazione relativa alla sua lunghezza
- Tale informazione si trova in un campo di nome `length` e di tipo `int`

Esempio

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

```
frazioni.length
```

- È un'espressione di tipo `int`
- Il suo valore è `4`

Accesso agli elementi di un array

Accesso agli elementi

nome_array[*selettore*]

Accesso agli elementi di un array

Accesso agli elementi

nome_array[*selettore*]

- *selettore*
  Dev'essere un'espressione di tipo `int`

  Espressioni di tipo `short`, `byte`, o `char` vengono promosse automaticamente a `int`

Accesso agli elementi di un array

Accesso agli elementi

nome_array[*selettore*]

- *selettore*
  Dev'essere un'espressione di tipo `int`

  Espressioni di tipo `short`, `byte`, o `char` vengono promosse automaticamente a `int`

- Le posizioni di un array sono contate a partire da zero

Accesso agli elementi di un array

Accesso agli elementi

nome_array[*selettore*]

- *selettore*
  Dev'essere un'espressione di tipo `int`

  > Espressioni di tipo `short`, `byte`, o `char` vengono promosse automaticamente a `int`

- Le posizioni di un array sono contate a partire da zero
- *nome_array*[*selettore*] è una variabile con:

  tipo tipo base dell'array

Accesso agli elementi di un array

Accesso agli elementi

nome_array[*selettore*]

- *selettore*
  Dev'essere un'espressione di tipo `int`

  > Espressioni di tipo `short`, `byte`, o `char` vengono promosse automaticamente a `int`

- Le posizioni di un array sono contate a partire da zero
- *nome_array*[*selettore*] è una variabile con:

  tipo tipo base dell'array

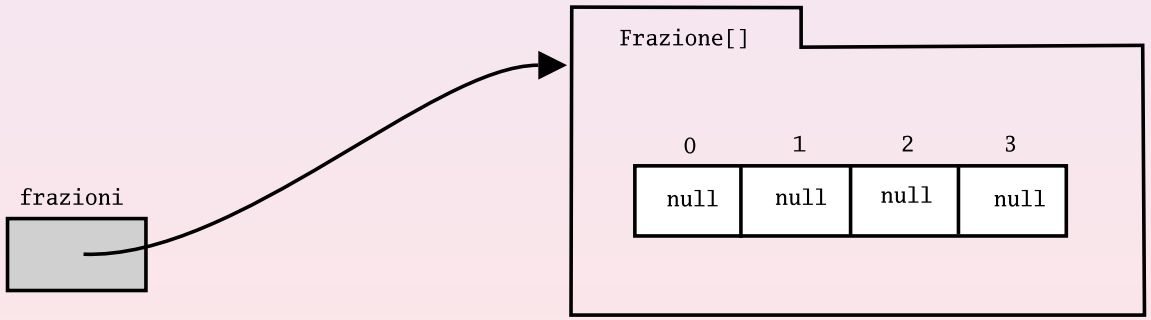
  valore il contenuto della posizione corrispondente dell'array

Esempio

```
Frazione[] frazioni;
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]);
```

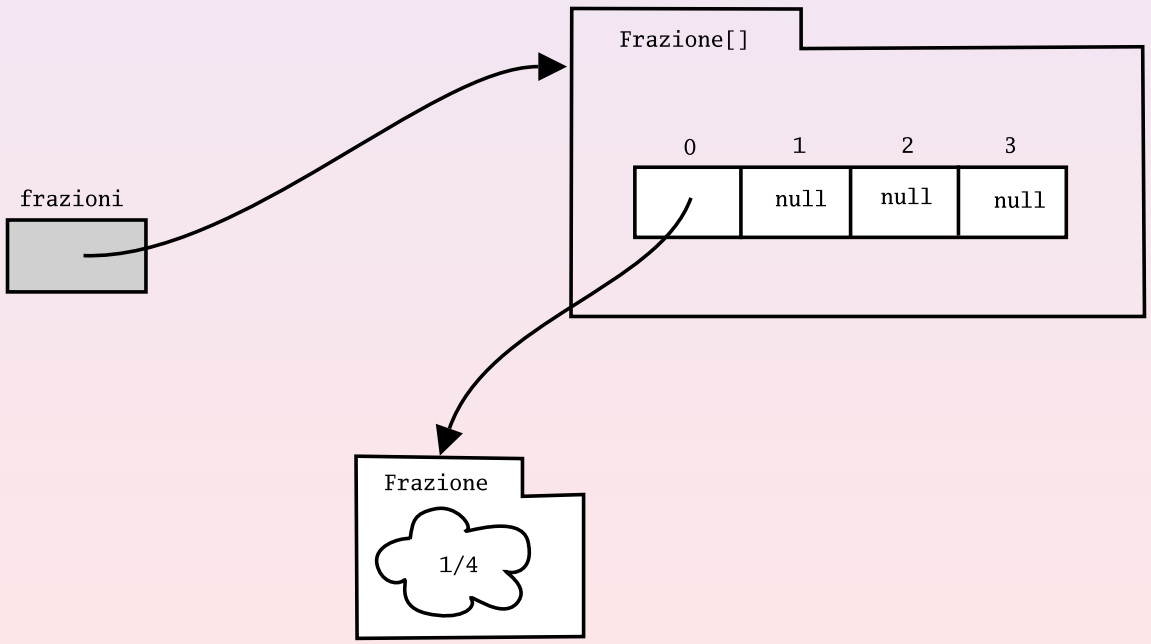
Accesso alle componenti

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



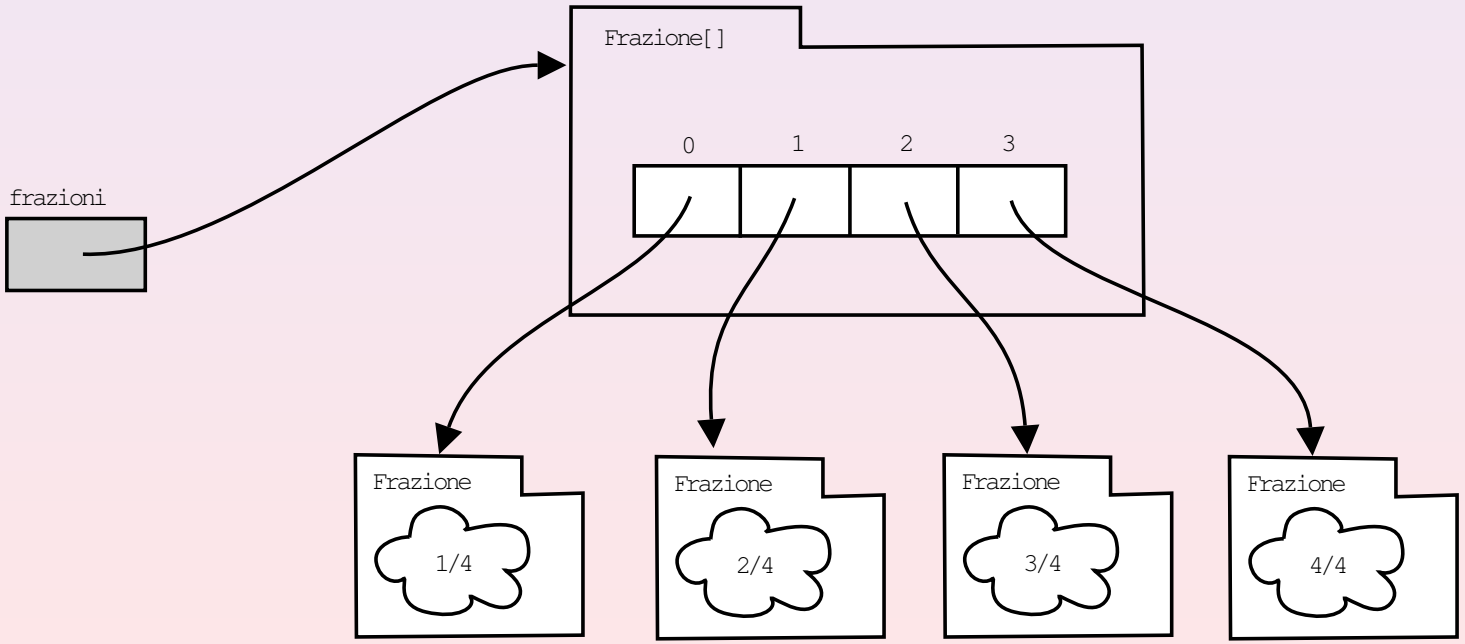
Accesso alle componenti

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

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



Accesso alle componenti

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

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



Accesso agli elementi di un array

Il tentativo di accedere a una componente non definita dell'array causa un errore in fase di esecuzione

Accesso agli elementi di un array

Il tentativo di accedere a una componente non definita dell'array causa un errore in fase di esecuzione

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

Accesso agli elementi di un array

Il tentativo di accedere a una componente non definita dell'array causa un errore in fase di esecuzione

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



Exception in thread "main"
  java.lang.ArrayIndexOutOfBoundsException: 4
...
```

Array e cicli `for`

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

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

for (int i = 0; i < frazioni.length; i++)
   out.println(frazioni[i].toString());
```

for–each

Sintassi

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

for–each

Sintassi

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

Esempio

```
Frazione[] frazioni = new Frazione[4];
...
for (Frazione f: frazioni)
   out.println(f.toString());
```

for–each

Sintassi

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

Esempio

```
Frazione[] frazioni = new Frazione[4];
...
for (Frazione f: frazioni)
   out.println(f.toString());
```

è equivalente a:

```
for (int i = 0; i < frazioni.length; i++)
   out.println(frazioni[i].toString());
```

Osservazione: for–each

- Consente di ottenere uno dopo l'altro i valori contenuti nell'array

Osservazione: for–each

- Consente di ottenere uno dopo l'altro i valori contenuti nell'array
- Non consente di accedere alle posizioni dell'array e quindi non consente di modificare l'array

Osservazione: for–each

- Consente di ottenere uno dopo l'altro i valori contenuti nell'array
- Non consente di accedere alle posizioni dell'array e quindi non consente di modificare l'array

Esempio

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

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

Osservazione: for–each

- Consente di ottenere uno dopo l'altro i valori contenuti nell'array
- Non consente di accedere alle posizioni dell'array e quindi non consente di modificare l'array

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());
```

Inizializzazione di array

Si può specificare fra parentesi graffe la sequenza di valori che costituiscono l'array

Inizializzazione di array

Si può specificare fra parentesi graffe la sequenza di valori che costituiscono l'array

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

Inizializzazione di array

Si può specificare fra parentesi graffe la sequenza di valori che costituiscono l'array

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

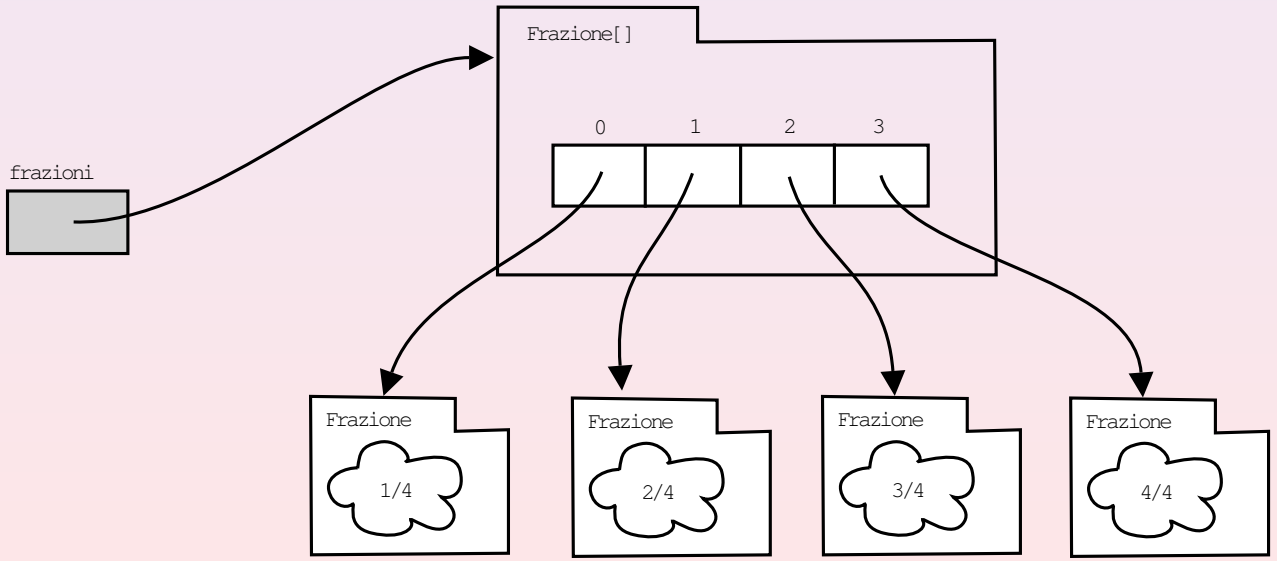
La dimensione dell'array viene dedotta dal compilatore

Inizializzazione di array

Si può specificare fra parentesi graffe la sequenza di valori che costituiscono l'array

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

La dimensione dell'array viene dedotta dal compilatore



Errore: `for`

```
String[] nomi;

//fase di scrittura
for (int pos = 0; pos < nomi.length; pos++)
  out.println(nomi[pos].toString());
```

Errore: for

```
String[] nomi;

//fase di scrittura
for (int pos = 0; pos < nomi.length; pos++)
  out.println(nomi[pos].toString());
```

Compilazione

```
 > javac UsoErratoArray.java
...: variable nomi might not have been initialized
    for (int pos = 0; pos < nomi.length; pos++)
                                ^
1 error
```

Errore: for–each

```
String[] nomi;

//fase di scrittura
for (String s: nomi)
  out.println(nomi.toString());
```

Errore: for–each

```
String[] nomi;

//fase di scrittura
for (String s: nomi)
  out.println(nomi.toString());
```

Compilazione

```
 > javac UsoErratoArray.java
...: variable nomi might not have been initialized
    for (String s: nomi)
                   ^
1 error
```

Errore: `for`

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

//fase di scrittura
for (int pos = 0; pos < frazioni.length; pos++)
  out.println(frazioni[pos].toString());
```

Errore: `for`

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

//fase di scrittura
for (int pos = 0; pos < frazioni.length; pos++)
  out.println(frazioni[pos].toString());
```

Esecuzione

```
> java UsoErratoArray
Exception in thread "main" java.lang.NullPointerException
        at UsoErratoArray.main(UsoErratoArray.java:14)
```

Errore: for–each

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

//fase di scrittura
for (Frazione f: frazioni)
  out.println(f.toString());
```

Errore: for–each

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

//fase di scrittura
for (Frazione f: frazioni)
  out.println(f.toString());
```

Esecuzione

```
> java UsoErratoArray
Exception in thread "main" java.lang.NullPointerException
        at UsoErratoArray.main(UsoErratoArray.java:14)
```

L'intestazione del metodo `main`

```
public static void main(String[] args)
```

L'intestazione del metodo `main`

```
public static void main(String[] args)
```

- `static`
  È un metodo statico

L'intestazione del metodo `main`

```
public static void main(String[] args)
```

- `static`
  È un metodo statico
- `void`
  Non restituisce alcun valore

L'intestazione del metodo `main`

```
public static void main(String[] args)
```

- `static`
  È un metodo statico
- `void`
  Non restituisce alcun valore
- `String[] args`
  Riceve come argomento il riferimento ad un array di stringhe

Esecuzione del metodo `main`

```
public static void main(String[] args)
```

Quando si manda in esecuzione una classe:

- La JVM cerca nel bytecode della classe un metodo statico con il prototipo descritto sopra

Esecuzione del metodo `main`

```
public static void main(String[] args)
```

Quando si manda in esecuzione una classe:

- La JVM cerca nel bytecode della classe un metodo statico con il prototipo descritto sopra
- Se lo trova, lo invoca passandogli come argomento il riferimento all'array contenente le stringhe specificate nel comando di esecuzione dopo il nome della classe

Esecuzione del metodo `main`

```
public static void main(String[] args)
```

Quando si manda in esecuzione una classe:

- La JVM cerca nel bytecode della classe un metodo statico con il prototipo descritto sopra
- Se lo trova, lo invoca passandogli come argomento il riferimento all'array contenente le stringhe specificate nel comando di esecuzione dopo il nome della classe
- Se non vengono forniti argomenti è l'array vuoto (cioè `args.length` vale `0`)

Esempio

```
import prog.io.*;

class Ripeti {
  public static void main(String[] args) {
    ConsoleOutputManager out = new ConsoleOutputManager();

    for (String s : args)
      out.println(s);
  }
}
```

Esempio

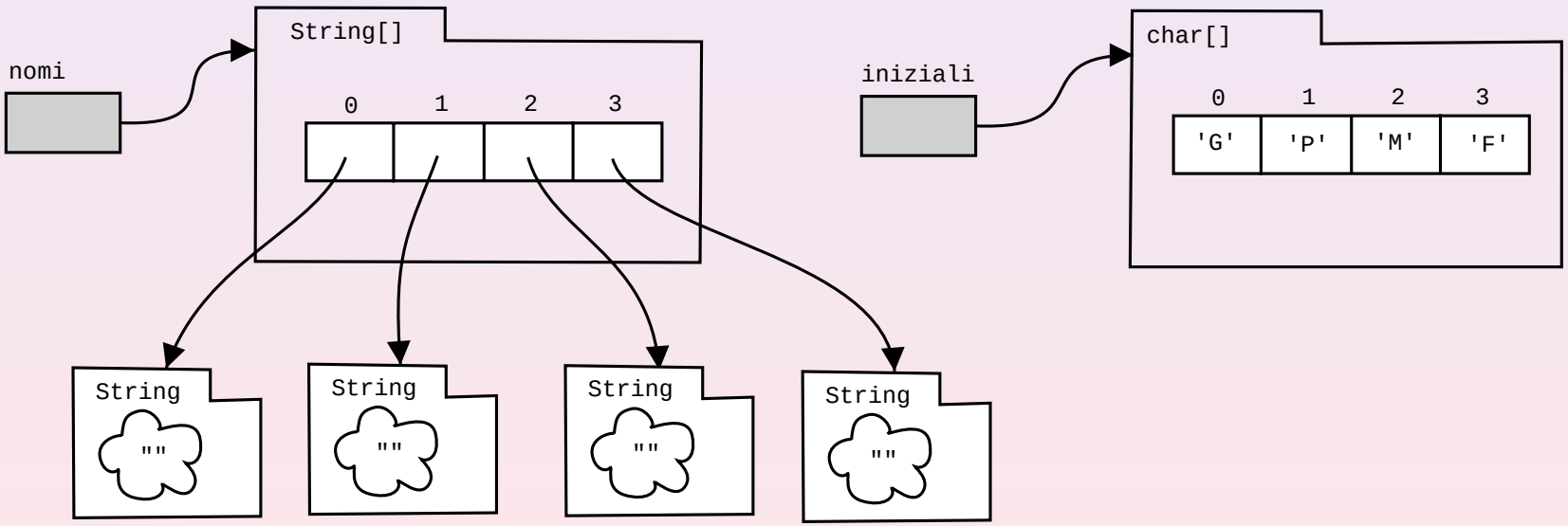
```
import prog.io.*;

class Ripeti {
  public static void main(String[] args) {
    ConsoleOutputManager out = new ConsoleOutputManager();

    for (String s : args)
      out.println(s);
  }
}
```

Esecuzione

```
> java Ripeti PIPPO pluto
PIPPO
pluto
```

Differenza fra array di oggetti e di tipo primitivo

```
String[] nomi = {"", "", "", ""};
char[] inizali = {'G', 'P', 'M', 'F'};
```

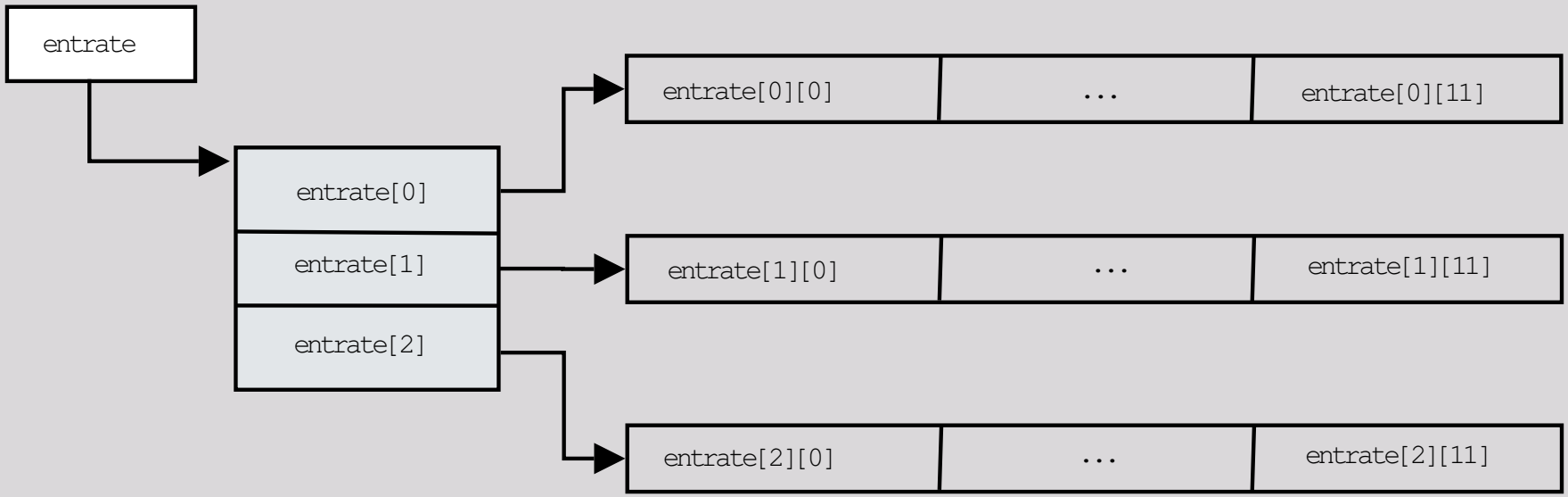


Array multidimensionali

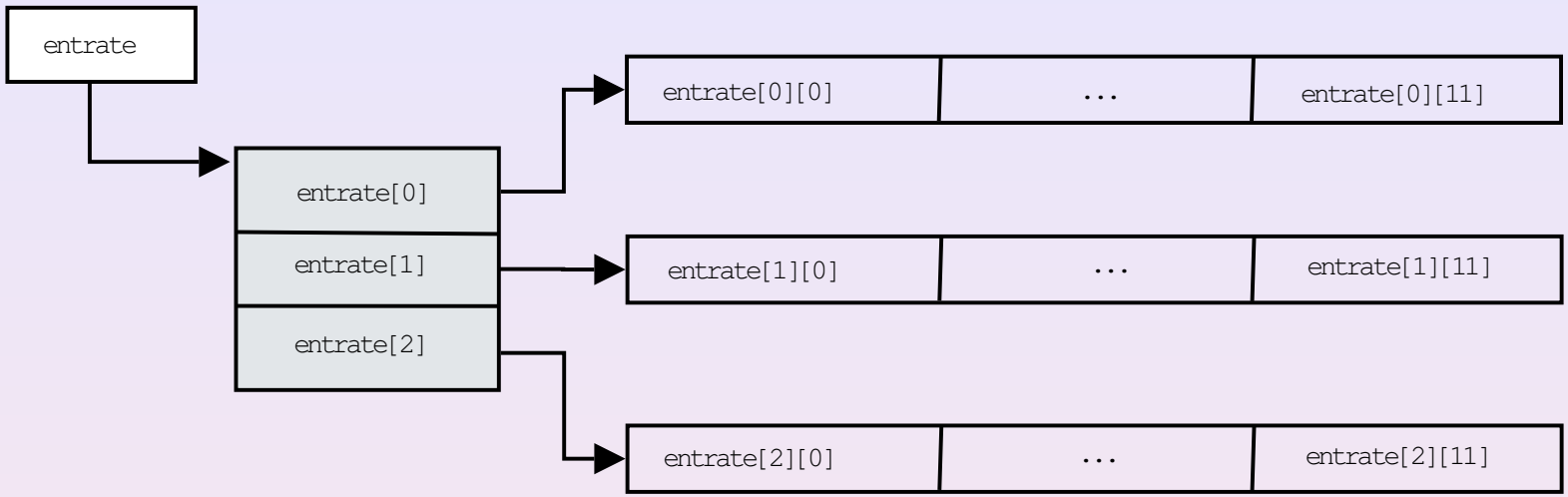
Array i cui elementi sono a loro volta array

Array bidimensionale (matrice)

```
int NANNI = 3;
int NMESI = 12;
Importo[][] entrate = new Importo[NANNI][NMESI];
```



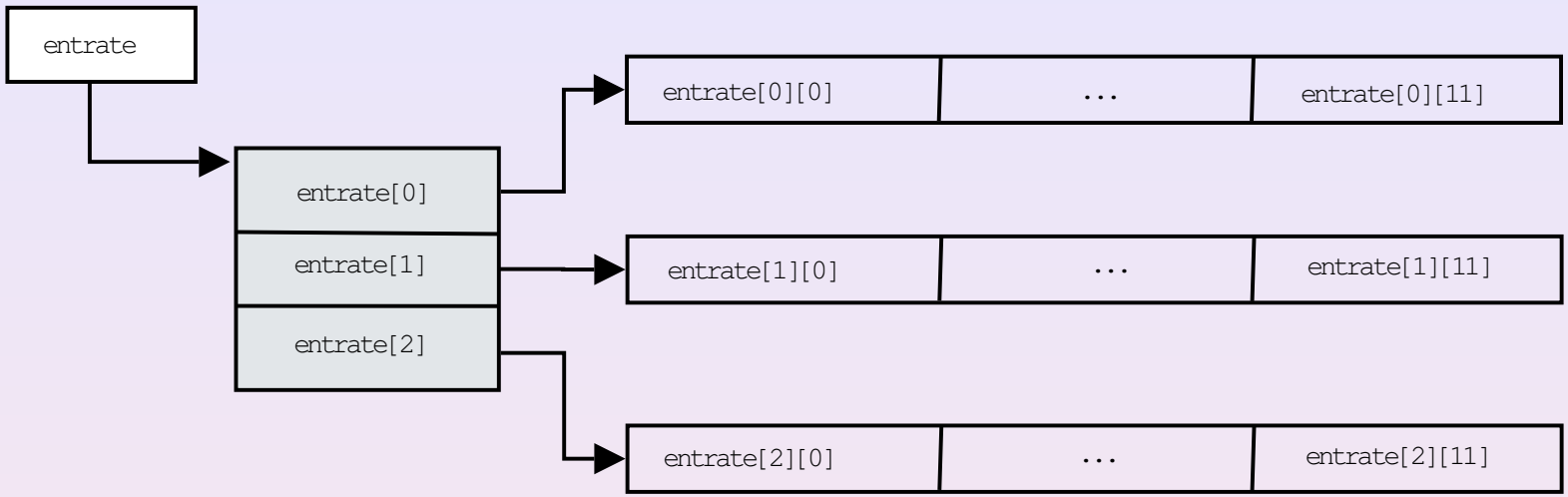
Dichiarazione e accesso



- Accesso all'elemento memorizzato nella seconda riga e terza colonna:

```
Importo i = entrate[1][2];
```

Dichiarazione e accesso



- Accesso all'elemento memorizzato nella seconda riga e terza colonna:

```
Importo i = entrate[1][2];
```

- Dimensioni:
  - Il numero delle righe è `entrate.length`
  - Il numero delle colonne della riga `i` è `matrice[i].length`

Sommario: Array e collezioni

La classe generica Sequenza<E>

Contratto

Le sue istanze rappresentano sequenze di oggetti di un tipo `E`, cioè collezioni di oggetti che possono contenere duplicazioni.
Nella sequenza gli oggetti compaiono nell'ordine in cui sono stati inseriti.

La classe generica Sequenza<E>

Contratto

Le sue istanze rappresentano sequenze di oggetti di un tipo `E`, cioè collezioni di oggetti che possono contenere duplicazioni.
Nella sequenza gli oggetti compaiono nell'ordine in cui sono stati inseriti.

Le classi e i tipi generici:

- Indicati nella documentazione con una notazione del tipo `Sequenza<E>`

La classe generica Sequenza<E>

Contratto

Le sue istanze rappresentano sequenze di oggetti di un tipo `E`, cioè collezioni di oggetti che possono contenere duplicazioni.
Nella sequenza gli oggetti compaiono nell'ordine in cui sono stati inseriti.

Le classi e i tipi generici:

- Indicati nella documentazione con una notazione del tipo `Sequenza<E>`
- `E` viene detto tipo parametro

La classe generica Sequenza<E>

Contratto

Le sue istanze rappresentano sequenze di oggetti di un tipo `E`, cioè collezioni di oggetti che possono contenere duplicazioni.
Nella sequenza gli oggetti compaiono nell'ordine in cui sono stati inseriti.

Le classi e i tipi generici:

- Indicati nella documentazione con una notazione del tipo `Sequenza<E>`
- `E` viene detto tipo parametro
- Possiamo dire che la classe o il tipo è "`Sequenza di E`"

Creazione di un oggetto di una classe generica

```
new Sequenza<String>()
```

Creazione di un oggetto di una classe generica

```
new Sequenza<String>()
```

È un oggetto in grado di memorizzare una sequenza di stringhe

Creazione di un oggetto di una classe generica

```
new Sequenza<String>()
```

È un oggetto in grado di memorizzare una sequenza di stringhe

```
new Sequenza<Frazione>()
```

Creazione di un oggetto di una classe generica

```
new Sequenza<String>()
```

È un oggetto in grado di memorizzare una sequenza di stringhe

```
new Sequenza<Frazione>()
```

È un oggetto in grado di memorizzare una sequenza di frazioni

Memorizzazione di un oggetto di un tipo generico

```
Sequenza<String> seq = new Sequenza<String>()
```

Memorizzazione di un oggetto di un tipo generico

```
Sequenza<String> seq = new Sequenza<String>()
```

```
Sequenza<Frazione> = new Sequenza<Frazione>()
```

Memorizzazione di un oggetto di un tipo generico

```
Sequenza<String> seq = new Sequenza<String>()
```

```
Sequenza<Frazione> = new Sequenza<Frazione>()
```

I tipi `Sequenza<String>` e `Sequenza<Frazione>` vengono detti tipi parametrizzati

Sequenza<E>

Costruttori

- `public Sequenza()`
  Costruisce una sequenza vuota.

Sequenza<E>

Costruttori

- `public Sequenza()`
  Costruisce una sequenza vuota.

Metodi

- `public boolean add(E o)`
  Aggiunge alla fine della sequenza l'oggetto fornito tramite l'argomento e restituisce `true`.
  Nel caso come argomento venga fornito `null`, non modifica la sequenza e restituisce `false`.

Sequenza<E>

Costruttori

- `public Sequenza()`
  Costruisce una sequenza vuota.

Metodi

- `public boolean add(E o)`
  Aggiunge alla fine della sequenza l'oggetto fornito tramite l'argomento e restituisce `true`.
  Nel caso come argomento venga fornito `null`, non modifica la sequenza e restituisce `false`.

- `public int size()`
  Restituisce il numero di elementi presenti nella sequenza.

Sequenza<E>

Costruttori

- `public Sequenza()`
  Costruisce una sequenza vuota.

Metodi

- `public boolean add(E o)`
  Aggiunge alla fine della sequenza l'oggetto fornito tramite l'argomento e restituisce `true`.
  Nel caso come argomento venga fornito `null`, non modifica la sequenza e restituisce `false`.

- `public int size()`
  Restituisce il numero di elementi presenti nella sequenza.

- `public boolean isEmpty()`
  Restituisce `true` se e solo se la sequenza è vuota.

Sequenza<E>

Metodi

- `public boolean contains(E o)`
  Restituisce `true` se e solo se la sequenza contiene un oggetto uguale (sulla base del criterio di uguaglianza fornito dal metodo `equals`) a quello specificato tramite l'argomento.

Sequenza<E>

Metodi

- `public boolean contains(E o)`
  Restituisce `true` se e solo se la sequenza contiene un oggetto uguale (sulla base del criterio di uguaglianza fornito dal metodo `equals`) a quello specificato tramite l'argomento.

- `public E find(E o)`
  Restituisce il riferimento al primo oggetto nella sequenza uguale a quello specificato tramite l'argomento, o `null` se tale oggetto non è presente.

Sequenza<E>

Metodi

- `public boolean contains(E o)`
  Restituisce `true` se e solo se la sequenza contiene un oggetto uguale (sulla base del criterio di uguaglianza fornito dal metodo `equals`) a quello specificato tramite l'argomento.

- `public E find(E o)`
  Restituisce il riferimento al primo oggetto nella sequenza uguale a quello specificato tramite l'argomento, o `null` se tale oggetto non è presente.

- `public boolean remove(E o)`
  Elimina dalla sequenza il primo oggetto uguale a quello specificato tramite l'argomento e restituisce `true`. Nel caso tale oggetto non vi sia, lascia la sequenza immutata e restituisce `false`.

PappagalloConMemoria

```
...
//predisposizione della "memoria"
Sequenza<String> memo = new Sequenza<String>();
```

PappagalloConMemoria

```
...
//predisposizione della "memoria"
Sequenza<String> memo = new Sequenza<String>();

//fase di lettura
String s = in.readLine();
while (!s.equals("")) {
  memo.add(s);
  s = in.readLine();
}
```

PappagalloConMemoria

```
...
//predisposizione della "memoria"
Sequenza<String> memo = new Sequenza<String>();

//fase di lettura
String s = in.readLine();
while (!s.equals("")) {
  memo.add(s);
  s = in.readLine();
}

//visualizzazione della sequenza

...per ogni elemento della sequenza
   visualizzalo
...
```

Sequenza e ciclo `for-each`

È possibile scorrere gli elementi contenuti in un oggetto di tipo `Sequenza<E>`, dal primo all'ultimo, utilizzando un ciclo for–each.

Sequenza e ciclo for-each

È possibile scorrere gli elementi contenuti in un oggetto di tipo Sequenza<E>, dal primo all'ultimo, utilizzando un ciclo for–each.

```
Sequenza<E> sequenza;
...
for (E elemento: sequenza)
  ...usa elemento...
```

Sequenza e ciclo for-each

È possibile scorrere gli elementi contenuti in un oggetto di tipo Sequenza<E>, dal primo all'ultimo, utilizzando un ciclo for–each.

```
Sequenza<E> sequenza;
...
for (E elemento: sequenza)
  ...usa elemento...
```

Esempio

```
Sequenza<String> memo = new Sequenza<String>();
...
//fase di scrittura
for (String x : memo)
  out.println(x);
```

PappagalloConMemoria

```
...
//predisposizione della "memoria"
Sequenza<String> memo = new Sequenza<String>();

//fase di lettura
String s = in.readLine();
while (!s.equals("")) {
  memo.add(s);
  s = in.readLine();
}

//fase di scrittura
for (String x : memo)
  out.println(x);
...
```

La classe generica `SequenzaOrdinata`

Contratto

Le sue istanze rappresentano sequenze ordinate di oggetti di tipo `E`.

La classe generica `SequenzaOrdinata`

Contratto

Le sue istanze rappresentano sequenze ordinate di oggetti di tipo `E`.

Ad esempio:

- Se `E` è il tipo `String`
  la sequenza è ordinata secondo l'ordine alfabetico

La classe generica `SequenzaOrdinata`

Contratto

Le sue istanze rappresentano sequenze ordinate di oggetti di tipo `E`.

Ad esempio:

- Se `E` è il tipo `String`
  la sequenza è ordinata secondo l'ordine alfabetico
- Se `E` è il tipo `Frazione` o il tipo `Integer`
  la sequenze è ordinata in maniera crescente

La classe generica `SequenzaOrdinata`

Contratto

Le sue istanze rappresentano sequenze ordinate di oggetti di tipo `E`.

Ad esempio:

- Se `E` è il tipo `String`
  la sequenza è ordinata secondo l'ordine alfabetico
- Se `E` è il tipo `Frazione` o il tipo `Integer`
  la sequenze è ordinata in maniera crescente
- Se `E` è il tipo `Data`
  la sequenza è ordinata cronologicamente

PappagalloOrdinato

```
...
//predisposizione della "memoria"
SequenzaOrdinata<String> memo = new SequenzaOrdinata<String>();
```

PappagalloOrdinato

```
...
//predisposizione della "memoria"
SequenzaOrdinata<String> memo = new SequenzaOrdinata<String>();

//fase di lettura
String s = in.readLine();
while (!s.equals("")) {
  memo.add(s);
  s = in.readLine();
}
```

PappagalloOrdinato

```
...
//predisposizione della "memoria"
SequenzaOrdinata<String> memo = new SequenzaOrdinata<String>();

//fase di lettura
String s = in.readLine();
while (!s.equals("")) {
  memo.add(s);
  s = in.readLine();
}

//fase di scrittura
for (String x : memo)
  out.println(x);
...
```

Osservazione

- È possibile costruire sequenze di oggetti di un qualunque tipo

> Possiamo usare qualunque tipo riferimento per istanziare il tipo parametro `E` di `Sequenza<E>`

Osservazione

- È possibile costruire sequenze di oggetti di un qualunque tipo

  Possiamo usare qualunque tipo riferimento per istanziare il tipo parametro `E` di `Sequenza<E>`

- Per costruire sequenze ordinate è necessario che il tipo degli elementi sia "ordinabile", cioè che sia definita una relazione di ordine totale tra i suoi elementi.

  Possiamo usare solo tipi riferimento su cui sia definita una relazione di ordine totale per istanziare il tipo parametro `E` di `SequenzaOrdinata<E>`

Osservazione

- È possibile costruire sequenze di oggetti di un qualunque tipo

  Possiamo usare qualunque tipo riferimento per istanziare il tipo parametro `E` di `Sequenza<E>`

- Per costruire sequenze ordinate è necessario che il tipo degli elementi sia "ordinabile", cioè che sia definita una relazione di ordine totale tra i suoi elementi.

  Possiamo usare solo tipi riferimento su cui sia definita una relazione di ordine totale per istanziare il tipo parametro `E` di `SequenzaOrdinata<E>`

- Esistono quindi delle limitazioni sulla genericità della classe `SequenzaOrdinata` che studieremo nel seguito