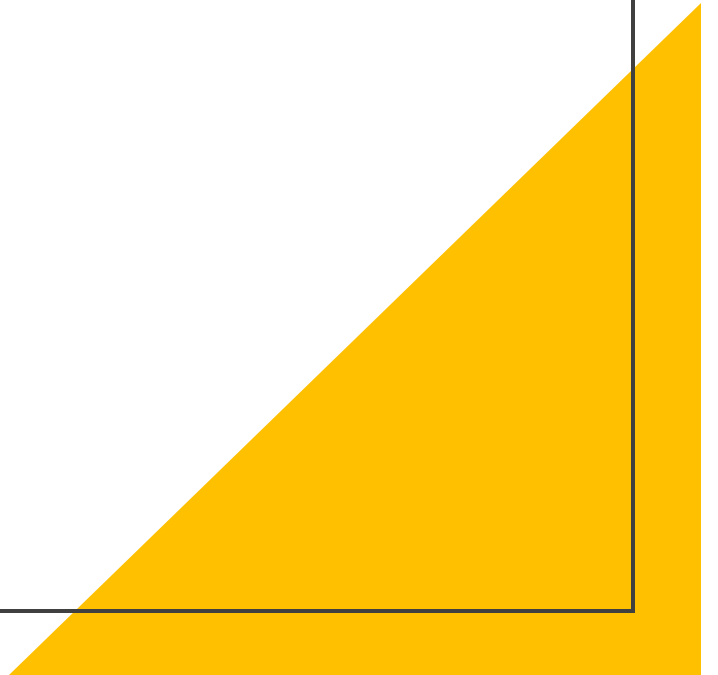


Array List



Array bidimensionali

Una disposizione composta da righe e colonne di valori → Chiamata anche matrice.

Esempio: conteggio delle medaglie delle gare di pattinaggio artistico alle Olimpiadi invernali 2014.

Possiamo utilizzare un'array bidimensionale per archiviare dati tabulari.

Quando si costruisce un array bidimensionale, si deve specificare quante righe e colonne sono necessarie:

```
final int COUNTRIES = 8;  
final int MEDALS = 3;  
int[][] counts = new int[COUNTRIES][MEDALS];
```

	Gold	Silver	Bronze
Canada	0	3	0
Italy	0	0	1
Germany	0	0	1
Japan	1	0	0
Kazakhstan	0	0	1
Russia	3	1	1
South Korea	0	1	0
United States	1	0	1

Array bidimensionali

- Puoi dichiarare e inizializzare l'array raggruppando ogni riga:

```
int[][] counts = {  
    { 0, 3, 0 },  
    { 0, 0, 1 },  
    { 0, 0, 1 },  
    { 1, 0, 0 },  
    { 0, 0, 1 },  
    { 3, 1, 1 },  
    { 0, 1, 0 },  
    { 1, 0, 1 } };
```

- Non è possibile modificare la dimensione di una matrice bidimensionale una volta dichiarata

Array bidimensionali

Diagram illustrating the declaration of a 2D array:

```
double[][] tableEntries = new double[7][3];
```

Annotations:

- Name: `tableEntries`
- Element type: `double`
- Number of rows: `7`
- Number of columns: `3`

All values are initialized with 0.

Diagram illustrating the declaration of a 2D array with initial values:

```
int[][] data = {  
    { 16, 3, 2, 13 },  
    { 5, 10, 11, 8 },  
    { 9, 6, 7, 12 },  
    { 4, 15, 14, 1 },  
};
```

Annotations:

- Name: `data`

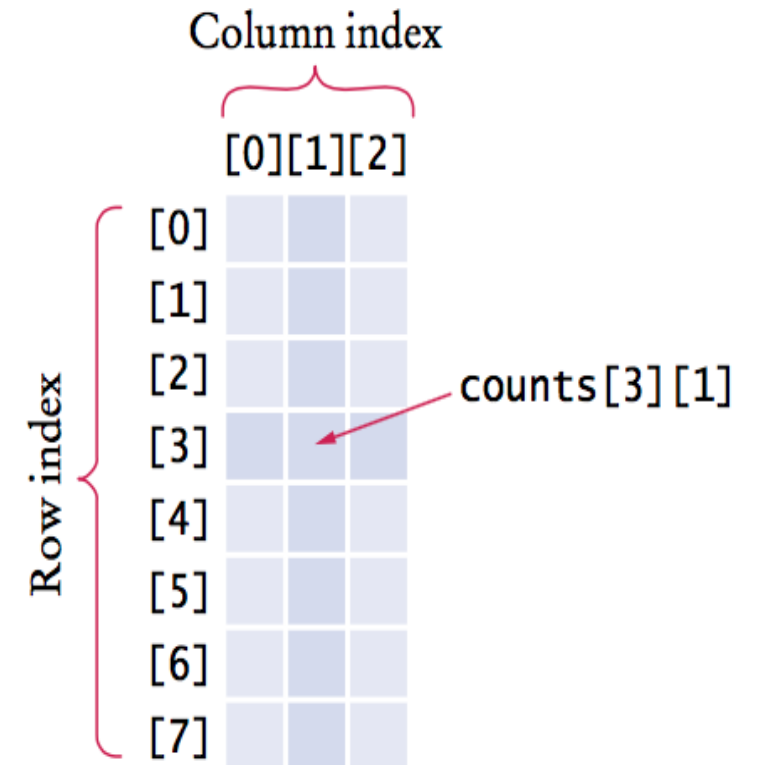
List of initial values

Accesso array bidimensionali

L'accesso? Utilizzando due valori di indice,
`array[i][j]`

```
int medalCount = counts[3][1];
```

Si deve usare i loop nidificati per accedere a tutti gli elementi in una matrice bidimensionale.



Accesso array bidimensionali

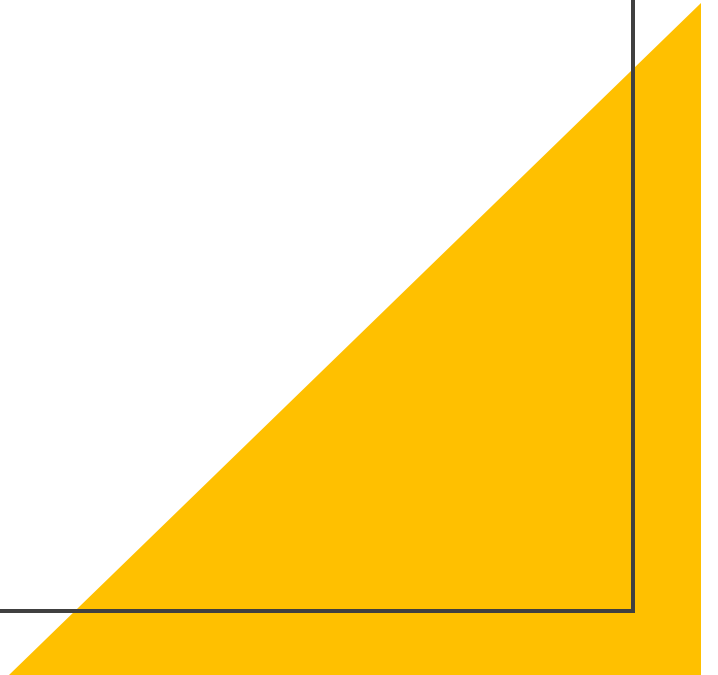
Esempio: stampa tutti gli elementi dell'array counts

```
for (int i = 0; i < COUNTRIES; i++)
{
    // Process the ith row
    for (int j = 0; j < MEDALS; j++)
    {
        // Process the jth column in the ith
        row    System.out.println(counts[i][j]);
    }
    System.out.println();
}
```

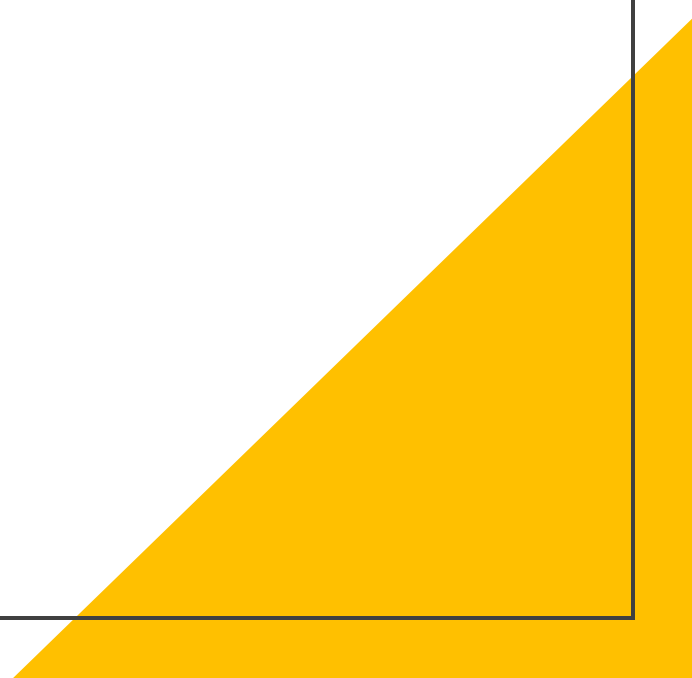
Accesso array bidimensionali

Numero di righe: `counts.length`

Numero di colonne: `conta[0].length`



Array list

- Un array list memorizza una sequenza di valori la cui dimensione può cambiare.
 - Un array list può crescere e ridursi secondo necessità.
 - La classe ArrayList fornisce metodi per molte attività comuni, come l'inserimento e la rimozione di elementi.
 - Un array list si espande per contenere tutti gli elementi necessari.
- 
- A large yellow triangle is positioned in the bottom right corner of the slide, pointing towards the top right.

Array list

Syntax

To construct an array list:

```
new ArrayList<typeName>()
```

To access an element:

```
arraylistReference.get(index)
```

```
arraylistReference.set(index, value)
```

Variable type

Variable name

An array list object of size 0

```
ArrayList<String> friends = new ArrayList<String>();
```

Use the
get and set methods
to access an element.

```
friends.add("Cindy");  
String name = friends.get(i);  
friends.set(i, "Harry");
```

The add method
appends an element to the array list,
increasing its size.

The index must be ≥ 0 and $< \text{friends.size}()$.

Dichiarare e usare un array list

- Dichiarare un vettore di stringhe

```
ArrayList<String> names = new ArrayList<String>();
```

- Per usarlo però:

```
import java.util.ArrayList;
```

- ArrayList è una classe generica
- Le parentesi angolari denotano un parametro del tipo
- Basta sostituire String con qualsiasi altra classe per ottenere un tipo di array list diverso

Dichiarare e usare un array list

- `ArrayList<String>` viene prima costruito, ha dimensione 0
- Utilizzare il metodo `add` per aggiungere un oggetto alla fine dell'array list:

```
names.add("Emily"); // Now names has size 1 and element "Emily"
```

```
names.add("Bob"); // Now names has size 2 and elements "Emily", "Bob"
```

```
names.add("Cindy"); // names has size 3 and elements "Emily", "Bob",  
and "Cindy"
```

- Il metodo `size` fornisce la dimensione corrente dell'array list.
- La size è 3

Dichiarare e usare un array list

- Per ottenere un elemento dell'elenco di array, bisogna utilizzare il metodo `get`

- L'indice inizia da 0

- Per recuperare il nome con l'indice 2:

```
String name = names.get(2);
```

- L'ultimo indice valido è `names.size() - 1`

- Un errore di bound comune:

```
int i = names.size();  
name = names.get(i); // Error
```

- Per impostare un elemento dell'array list su un nuovo valore, utilizzare il metodo `set`: `names.set(2, "Carolyn");`

Dichiarare e usare un array list

- Un array list ha metodi per aggiungere e rimuovere elementi nel mezzo
- Questa istruzione aggiunge un nuovo elemento alla posizione 1 e sposta tutti gli elementi con indice 1 o maggiore di una posizione

```
names.add(1, "Ann");
```

- Il metodo di rimozione, rimuove l'elemento in una determinata posizione sposta tutti gli elementi dopo l'elemento rimosso in basso di una posizione e riduce la dimensione dell'array list di 1.

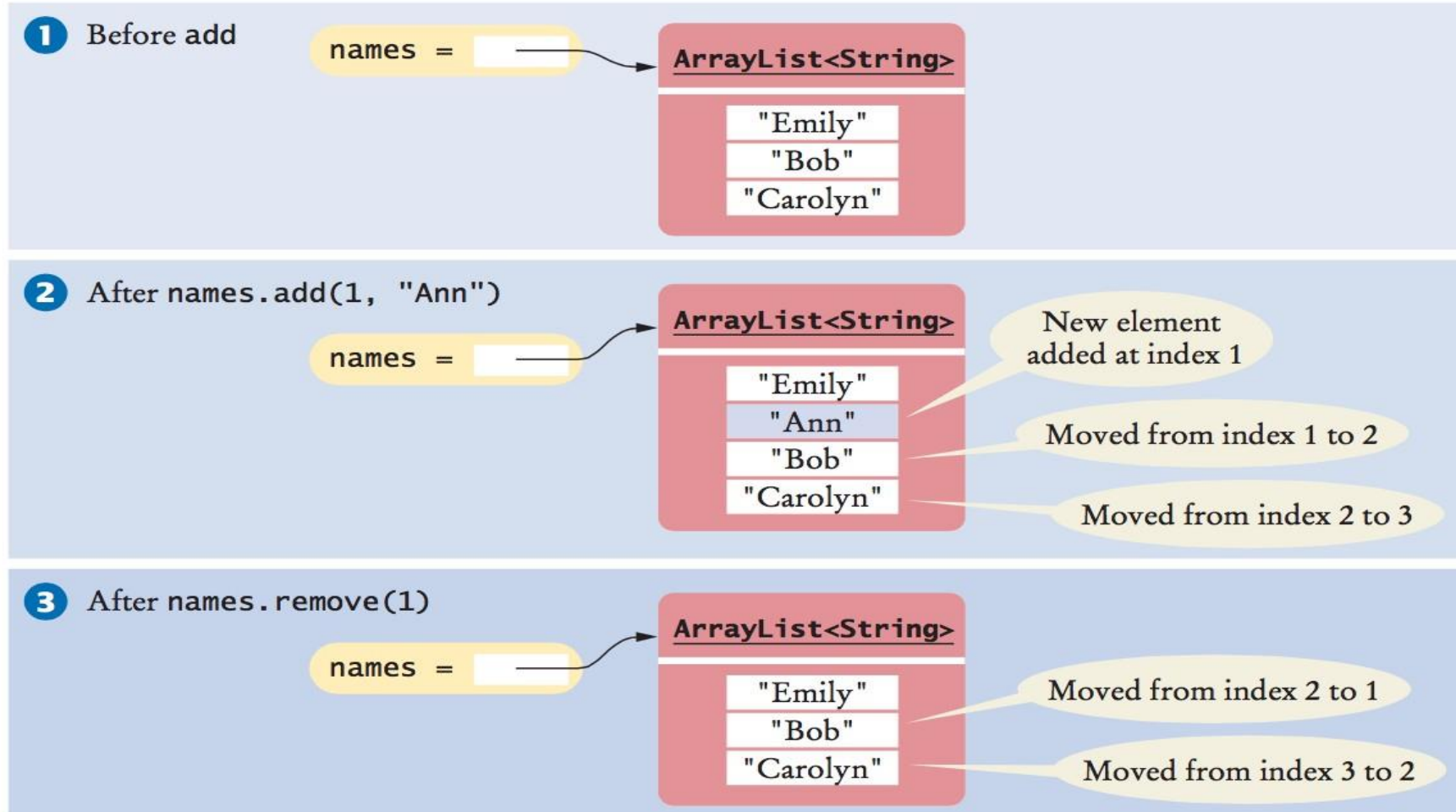
```
names.remove(1);
```

- Per stampare:

```
System.out.println(names);
```

```
// Stampa [Emily, Bob, Carolyn]
```

Dichiarare e usare un array list



For esteso

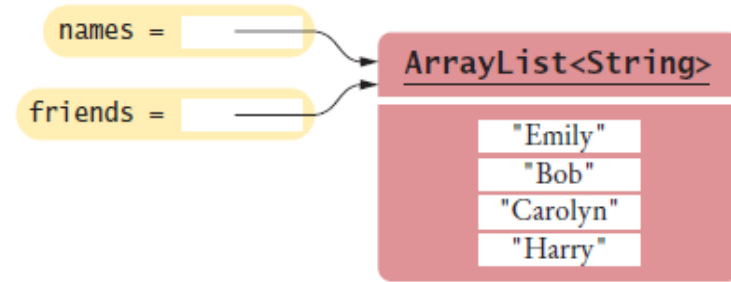
- È possibile utilizzare il ciclo for esteso per visitare tutti gli elementi di un array list

```
ArrayList<String> names = . . . ;  
for (String name : names)  
{  
    System.out.println(name);  
}
```

Equivalente:

```
for (int i = 0; i < names.size(); i++)  
{  
    String name = names.get(i);  
    System.out.println(name);  
}
```

Copiare array list



La copia di un riferimento a un array list produce due riferimenti allo stesso array list.

Dopo che il codice seguente è stato eseguito

Sia names sia friends fanno riferimento allo stesso array list a cui è stata aggiunta la stringa "Harry".

```
ArrayList<String> friends = names;  
friends.add("Harry");
```

Se voglio fare una copia vera:

```
ArrayList<String> newNames = new ArrayList<String>(names);
```



```
ArrayList<String> names = new ArrayList<String>();
```

```
names.add("Ann");  
names.add("Cindy");
```

```
System.out.println(names);
```

```
names.add(1, "Bob");
```

```
names.remove(0);
```

```
names.set(0, "Bill");
```

```
String name = names.get(i);
```

```
String last = names.get(names.size() - 1);
```

```
ArrayList<Integer> squares = new ArrayList<Integer>();  
for (int i = 0; i < 10; i++)  
{  
    squares.add(i * i);  
}
```

1. Costruisce un array list di String vuoto
2. Aggiunge elementi alla fine dell'array list
3. Stampa l'array list
4. Inserisce un elemento all'indice 1
5. Rimuove l'elemento all'indice 0
6. Sostituisce un elemento con un valore diverso.
7. Ottiene un elemento
8. Ottiene l'ultimo elemento
9. Costruisce un array list contenente i primi dieci quadrati.

Classi involucro (wrapped)

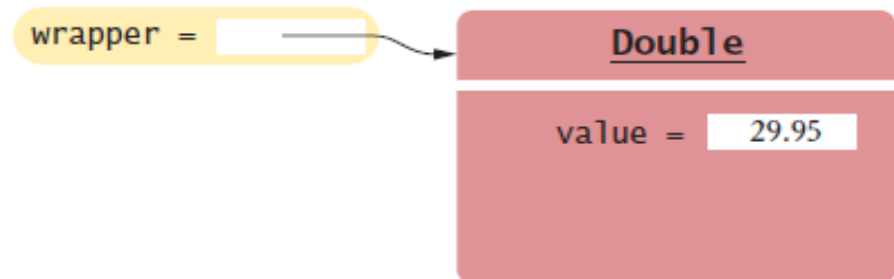
Non è possibile
inserire direttamente
valori di tipo primitivo
negli array list.

Un numero deve
essere inserito in un
«involucro» per essere
archiviato in un array
list

Primitive Type	Wrapper Class
byte	Byte
boolean	Boolean
char	Character
double	Double
float	Float
int	Integer
long	Long
short	Short

Classi involucro (wrapped)

- Per raccogliere valori double in un array list, si deve usare un `ArrayList<Double>`.
- se si assegna un valore double ad una variabile Double, il numero viene automaticamente “messo in una casella”
- Tutto ciò è chiamato auto-boxing:
 - Conversione automatica tra tipi primitivi e le classi wrapper corrispondenti:
 - `Double wrapper = 29.95;`
 - I valori del wrapper vengono automaticamente "unboxed" ai tipi primitivi
`double x = wrapper;`



Classi involucro (wrapped)

Gli algoritmi che abbiamo visto negli array possono essere convertiti in array list semplicemente utilizzando i metodi dell'array list anziché la sintassi dell'array.

Codice per trovare l'elemento più grande in un array e in un array list:

```
double largest = values[0];
for (int i = 1; i < values.length; i++)
{
    if (values[i] > largest)
    {
        largest =
        values[i];
    }
}
```

```
double largest = values.get(0);
for (int i = 1; i < values.size(); i++)
{
    if (values.get(i) > largest)
    {
        largest =
        values.get(i);
    }
}
```

Rimuovere specifici valori

- Per rimuovere elementi da un array list, bisogna richiamare il metodo remove.
- Errore: saltare l'elemento dopo l'elemento spostato

```
ArrayList<String> words = ...;  
for (int i = 0; i < words.size(); i++)  
{  
    String word = words.get(i);  
    if (word.length() < 4)  
    {  
        Remove the element at index i.  
    }  
}
```

i	words
0	"Welcome", "to", "the", "island"
1	"Welcome", "the", "island"
2	

Non dovrebbe incrementare i
quando un elemento viene rimosso

Rimuovere specifici valori

```
int i = 0;
while (i < words.size())
{
    String word = words.get(i);
    if (word.length() < 4)
    {
        words.remove(i);
    }
    else
    {
        i++;
    }
}
```

If the element at index *i* matches the condition

Remove the element.

Else

Increment *i*.

Array o array list?

- Per la maggior parte delle attività di programmazione, gli array list sono più facili da usare rispetto agli array
- Gli elenchi di array possono crescere e ridursi.
- Gli array hanno una sintassi migliore.
- Raccomandazioni:
 - Se la dimensione di una raccolta non cambia mai, utilizzare un array.
 - Se raccogli una lunga sequenza di valori di tipo primitivo e sei preoccupato per l'efficienza, usa un array.
 - In caso contrario, utilizzare un elenco di array.

Array o array list?

Operation	Arrays	Array Lists
Get an element.	<code>x = values[4];</code>	<code>x = values.get(4);</code>
Replace an element.	<code>values[4] = 35;</code>	<code>values.set(4, 35);</code>
Number of elements.	<code>values.length</code>	<code>values.size()</code>
Number of filled elements.	<code>currentSize</code> (companion variable, see Section 7.1.4)	<code>values.size()</code>
Remove an element.	See Section 7.3.6.	<code>values.remove(4);</code>
Add an element, growing the collection.	See Section 7.3.7.	<code>values.add(35);</code>
Initializing a collection.	<code>int[] values = { 1, 4, 9 };</code>	No initializer list syntax; call <code>add</code> three times.

ArrayList recap

- Gli array non possono cambiare la propria dimensione: il numero di elementi contenuti viene stabilito al momento della creazione e rimane immutato.
- Per superare questa limitazione Java mette a disposizione la classe `ArrayList`, contenuta nel package `java.util`, che permette di rappresentare sequenze di oggetti di lunghezza variabile.
- Ciascun oggetto in un'istanza di `ArrayList` viene identificato da un numero intero, detto indice, che ne indica la posizione.
- L'accesso ad una posizione inesistente provoca un errore (viene lanciata un'eccezione `IndexOutOfBoundsException` (se l'indice è fuori dal range (`index < 0 || index > size()`))).

ArrayList recap

L'ArrayList è quindi simile ad un array.

- Le differenze principali sono due:
- La dimensione può variare durante l'esecuzione di un programma
- Gli elementi contenuti sono di un solo tipo: Object.
- ArrayList è una classe come tutte le altre, non ha alcuna sintassi particolare

Costruttori

- La classe `ArrayList` definisce due costruttori:
- `ArrayList()`: crea un vettore vuoto in cui la capacità iniziale non è specificata (costruttore di default).
- `ArrayList(int initialCapacity)`: crea un vettore con la capacità iniziale indicata. Si utilizza quando si ha un'idea, anche approssimata, della dimensione massima che la lista raggiungerà.

Metodi

I metodi definiti dalla classe consentono tra l'altro di:

- Leggere o scrivere un elemento in una certa posizione (operazioni analoghe a quelle sugli array)
- Aggiungere uno o più elementi, in varie posizioni
- Eliminare uno o più elementi, in varie posizioni
- Cercare un oggetto contenuto
- Trasformare l'ArrayList in un array

Metodi

- `Object get(int index)`
Restituisce l'elemento di indice `index`.
- `Object set(int index, Object obj)`
Sostituisce `obj` all'oggetto di posizione `index`.
- `boolean add (Object obj)`
Aggiunge `obj` dopo l'ultimo elemento (restituisce sempre `true`).
- `void add (int index, Object obj)`
Inserisce `obj` nella posizione `index` e sposta tutti gli elementi, da `index` in poi, di una posizione.
- `int size()`
Restituisce il numero di elementi contenuti.
- `boolean isEmpty()`
Dice se la lista è vuota

Metodi

- `Object remove(int index)`
Rimuove l'oggetto presente nella posizione `index` e sposta all'indietro di una posizione tutti gli elementi successivi a quello rimosso.
- `int indexOf(Object elem)`
Restituisce la prima posizione dell'oggetto '`elem`' nel vettore, - 1 se non esiste.
- `String toString()`
Restituisce una stringa con l'elenco degli elementi contenuti: "`[el1, el2,... eln]`".
- `void clear()`
Svuota completamente la lista eliminando tutti gli elementi contenuti.
- `public Object[] toArray()` Restituisce un array con l'intero contenuto.

Esercizio

- Si realizzi un programma JAVA che faccia uso di un'istanza della classe ArrayList per memorizzare una lista di parole di un abecedario: albero, banana, cuscino, denti, elevatore.
- Si usino le diverse versioni del metodo add.
- Si stampi poi a video la lista di parole e il primo e l'ultimo elemento.

Esercizio

- Si realizzi una classe MyStack che, facendo uso di un'istanza della classe ArrayList, implementi il comportamento di uno stack senza nessuna limitazione in memoria.
- La classe deve implementare, oltre ai metodi push e pop, anche un costruttore (senza parametri) e un metodo getCount che restituisce il numero di elementi contenuti nello stack.
- Si implementi un metodo main che inserisca nello stack due stringhe e poi le estragga dallo stack per stamparle a video.