



Esempio su strutture dati dinamiche: ArrayList

ArrayList

- **Abbiamo detto che 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).**

ArrayList e array

- **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**

Il contenuto

- **Come abbiamo detto le istanze di ArrayList possono contenere solo istanze della classe Object.**
- **Questo vincolo è meno restrittiva di quanto sembrerebbe: in virtù del subtyping possiamo infatti mettere in un ArrayList istanze qualunque discendente di Object, ovvero qualunque oggetto Java**
- **Possiamo memorizzare oggetti di classi completamente scorrelate (come String, Rectangle, Persona) nella stessa istanza di ArrayList**
- **Quando li estraiamo dobbiamo però usare un downcast per passare dal tipo Object al tipo voluto**

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

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**

Elenco dei metodi - 1

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

Elenco dei metodi - 2

- Rimuove l'oggetto presente nella posizione `index` e sposta all'indietro di una posizione tutti gli elementi successivi a quello rimosso:
`void remove (int index)`
- Rimuove l'oggetto `obj` se presente restituendo `true`, oppure restituisce `false`:
`boolean remove (Object obj)`
- Restituisce la prima posizione dell'oggetto '`elem`' nel vettore, -1 se non esiste:
`int indexOf (Object elem)`
- Restituisce una stringa "[`e1`, `e2`,... `eN`]":
`String toString ()`

Esempio

```
import java.util.ArrayList;

public class EsempioArrayList
{
    public static void main(String[] args)
    {
        ArrayList v = new ArrayList (3);
        System.out.println("n.elementi di v: "+v.size());
        v.add("aaa"); v.add("bbb");
        v.add("ddd"); v.add("eee");
        v.add(2, "ccc");
        System.out.println("n. elementi di v: "+v.size());
        for (int i=0; i<v.size(); i++)
            System.out.println("elemento "+ i+": "+v.get(i));
        System.out.println("primo: "+v.get(0));
        System.out.println("ultimo: "+v.get(v.size()-1));
        String s = (String)v.get(0); // Serve un downcast è
    }
}
```

Memorizzare dati di tipi primitivi in ArrayList

- I tipi primitivi in Java non sono oggetti, quindi non è possibile inserirli direttamente in un vettore.
- Per memorizzare sequenze di numeri interi, numeri in virgola mobile, o valori di tipo boolean in un vettore, si devono usare le classi wrapper.

```
ArrayList dati = new ArrayList();

int n = 30;
Integer numero = new Integer(n);
dati.add(numero);

Integer numero = (Integer)dati.get(0);
int n = numero.intValue();
```

Implementazione

- L'implementazione è basata sugli array.
- Al momento della creazione di un vettore, viene utilizzato un array di dimensioni predefinite.

```
private Object[] v;
```

```
...
```

```
v = new Object[x];
```

Il valore di x
viene stabilito
dal sistema Java

- Successivamente, se la capacità dell'array non è più sufficiente, viene creato un nuovo array più grande nel quale vengono copiati tutti gli elementi del vecchio.

```
Object[] v2 = new Object[2*v.length];
```

```
System.arraycopy(v, 0, v2, 0, v.length);
```

```
v = v2;
```