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 capacita' 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

Elenco dei metodi - 1

- `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.

Elenco dei metodi - 2

- `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.

Esempio 1

```
import java.util.ArrayList;

public class EsempioArrayList
{
  public static void main(String[] args)
  {
    ArrayList v = new ArrayList();
    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"); // inserisce “ccc” prima di “ddd”
    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); // downcast obbligatorio
  }
}
```

Esempio 2: uno stack senza limiti

```
import java.util.*;
public class Stack
{
  private ArrayList st;

  public Stack()
  { st = new ArrayList(); }

  public void push(Object item)
  { st.add(item); }
  public Object pop()
  {
    if (!st.isEmpty())
      return st.remove(st.size()-1);
    else return null;
  }
  public int getCount()
  { return st.size(); }
}
```

Esempio 2: uso dello stack

```
public class EsempioStack
{
  public static void main(String[] args)
  {
    Stack s = new Stack();
    s.push("Ciao");
    s.push("Arrivederci!");
    String x;
    x = (String)s.pop();
    System.out.println(x);
    x = (String)s.pop();
    System.out.println(x);
  }
}
```

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;
```