Programmazione ad Oggetti

ArrayList

A.A. 2022/2023

Docente: Prof. Salvatore D'Angelo
Email: salvatore.dangelo@unicampania.it

Università degli Studi della Campania *Luigi Vanvitelli*
Dipartimento di Ingegneria

ArrayList

- La classe **ArrayList** è una classe parametrica (programmazione generica)
- In genere un ArrayList ha lo stesso scopo di un array
- A differenza degli array, che una volta creati hanno dimensione fissa, un ArrayList può cambiare la propria dimensione durante l'esecuzione del programma
- Risulta meno efficiente degli array

ArrayList

Gli elementi contenuti sono di un solo tipo: Object (utilizzo di wrapper per i tipi semplici).

Questo vincolo è meno restrittivo 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.

ArrayList aList = new ArrayList();

ArrayList

- E' contenuto nel package **java.util.***
  Array di tipo T
  ArrayList<T> aList = new ArrayList<T>();

```
ArrayList<BankAccount> accounts =
                    new ArrayList<BankAccount>();
accounts.add(new BankAccount(1001));
accounts.add(new BankAccount(1015));
accounts.add(new BankAccount(1022));
```

- Il tipo base <T> può essere solo un tipo strutturato (per i tipi semplici bisogna usare delle apposite classi *wrapper*

ArrayList

Come per gli array

- gli indici iniziano da 0
- errore se l'indice è fuori range
- posizioni accessibili: 0 .. size() – 1.

- **size():** il metodo che calcola il numero di elementi nella struttura

Accesso tramite il metodo **get()**

```
BankAccount anAccount = accounts.get(2);
   // il terzo elemento dalla arraylist
```

ArrayList

set() sovrascrive un valore esistente

```
BankAccount anAccount = new BankAccount(1729);
accounts.set(2, anAccount);
```

add() aggiunge un nuovo valore nella posizione i

```
accounts.add(i, a)
```

oppure all'ultima posizione

```
accounts.add(a)
```

remove() rimuove l'elemento all'indice i

```
accounts.remove(i)
```

Vector<T>

- La classe **Vector**<T> è simile alla classe ArrayList<T>
- E possibile fare le stesse cose della classe ArrayList, ma ha qualche metodo in più
- E' più vecchia come classe e meno efficinente in temini di prestazione

Mappe

- **HashMap<k,v>** Utizza una tabella hash (chiave, valore) e non fornisce alcuna garanzia sull'ordine degli elementi contenuti nella mappa
- **TreeMap<k,v>** che organizza i propri elementi in una struttura ordinata ad albero (chiave,valore)

- Tutte queste classi (ed altre) appartengono alla ***famiglia delle collection***, cioè una classe che contiene oggetti, ed implementano tutti l'interfaccia ***Collection<T>***

HashMap<k,v>

- HashMap è un'implementazione dell'interfaccia Map che fornisce una struttura dati per archiviare i dati in coppie chiave-valore
- Non esiste un ordinamento degli elementi

```
Map<KeyType, ValueType> myMap = new
HashMap<KeyType, ValueType>();
```

```
Map<String,Integer> myMap = new
HashMap<String,Integer>();
```

HashMap<k,v>

- Inserire dei valori

```
myMap.put("key1", 1);
myMap.put("key2", 2);
```

- Ottenere dei valori

```
myMap.get("key1"); //return 1 (class Integer)
```

- Controllare se la chiave è nella mappa o no.

```
myMap.containsKey(varKey);
```

- Controlla se il valore è nella mappa o no.

```
myMap.containsValue(varValue);
```

HashMap vs Hashtable

HashMap non è sincrono, cioè se usati in un contest multithread, più thread possono accedere e modificare/processare HashMap simultaneamente.

Hashtable è simile alla HashMap (chiave, valore), ma è sincrona

TreeMap<k,v>

E’ utilizzata in situazioni in cui è richiesto un ordinamento sulle chiavi

Esempio, realizziamo una TreeMap che memorizzi le coppie cognome-nome di una persona ordinandole alfabeticamente sul cognome

Si tratta di una classe la cui complessità di tempo per le varie operazioni è logaritmica nella dimensione della mappa.

Tipi generici

Le classi e i metodi possono usare un **tipo parametrico** al posto di uno specifico tipo di dato.

Quando il programmatore usa una tale classe o metodo, specifica il tipo di classe per il tipo parametrico

```
public class Esempio <T> {
     private T dati;
     public void setDati(T nuovoValore) { …
};
     public T getDati() ) { … };
}
```