Collezioni Java

Fondamenti di Informatica A-K

Esercitazione 6

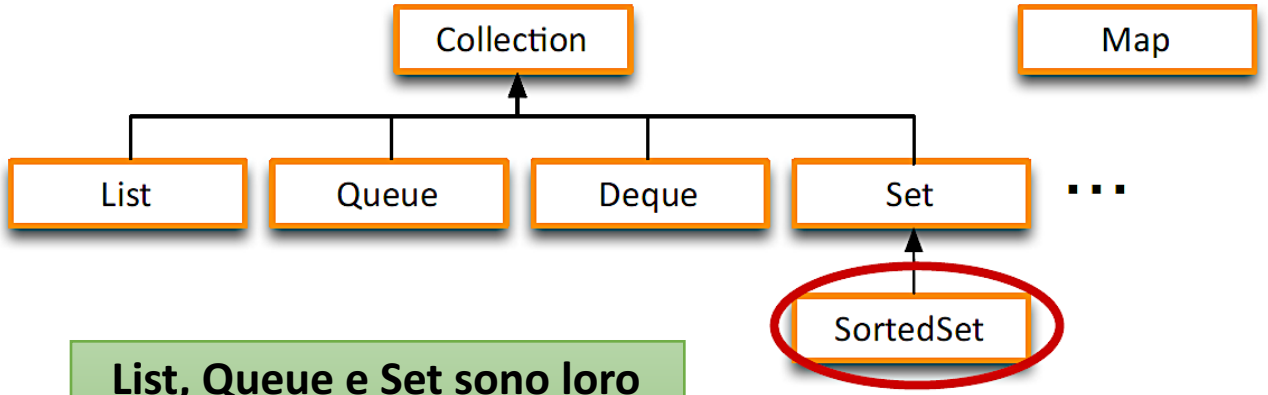
- Introduzione al calcolatore e Java
- Linguaggio Java, basi e controllo del flusso
- I metodi: concetti di base
- Stringhe ed array
- Classi e oggetti, costruttori, metodi statici, visibilità
- Ereditarietà e polimorfismo
- Collezioni Java

Cosa sono le `Collection` di Java

1. `java.util.Collection` è un'interfaccia
2. Una `Collection` rappresenta una qualunque collezione di oggetti, ad esempio:
   - `List` permette l'esistenza di duplicati, mentre `Set` **no**.
   - `List` è una collezione ordinata, mentre `Set` **no**.
   - L'ordinamento viene effettuato secondo il criterio definito nella compareTo(Object)
3. La JDK **non prevede** implementazione alcuna per questa interfaccia: prevede invece classi (come TreeSet ed ArrayList) che implementano interefacce più specifiche da essa derivate (come Set e List).

`interface Collection`

- L'interfaccia Collection definisce alcuni metodi, ad esempio:
  - `boolean add(Element e)` ← per inserire un elemento
  - `boolean remove(Element e)` ← per eliminare un elemento
  - `int size()` ← restituisce il numero di elementi nella collezione
  - `Boolean isEmpty()` ← restituisce `true` se la collezione è vuota



List, Queue e Set sono loro stesse delle interfacce!!

Interfaccia List tipo Array → `ArrayList`

`ArrayList<E>` rappresenta una collezione in cui:

1. Posso inserire l'emento i-esimo ad un indice specifico
   - `add(int index, E element)`
     Inserts the specified element at the specified position in this list.
   - `add(E e)`
     Appends the specified element to the end of this list.
2. In cui possono apparire duplicati.
3. La classe `ArrayList` implementa l'interfaccia `List`

Esempio `ArrayList`

```
List<String> strings = new ArrayList<String>();

strings.add(”B");
strings.add(”C");
strings.add(0, ”A");
strings.add(3, ”A");
strings.add(”C");
strings.add(strings.size() - 1, ”B");

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

```
Output: [A, B, C, A, B, C]
```

Interfaccia Tree tipo Set→ `TreeSet`

`TreeSet<E>` rappresenta una collezione che:

1. Impone ordinamento naturale (interfaccia Comparable)
   - `add(E e)`
     Adds the specified element to this set if it is not already present.
2. Non prevede duplicati → `e1.equals(e2) == false`
3. La classe **`TreeSet`** implementa l'interfaccia **`SortedSet`** che a sua volta implementa l'interfaccia **`Set`**

Esempio `TreeSet`

```
SortedSet<String> sortedSet = new TreeSet<String>();

sortedSet.add(”A");
sortedSet.add(”B");
sortedSet.add(”A");
sortedSet.add(”C");

System.out.println(sortedSet.toString());
```

Output: [A, B, C]

Ciclo `for` per le `Collection`

Nel caso delle collezioni al consueto costrutto di iterazione while o for se ne aggiunge un terzo più semplice nel caso si volesse scorrere tutta la collezione dall'inizio alla fine.

```
List<String> list = new ArrayList<String>();
list.add("Text 1");
list.add("Text 2");
list.add("Text 3");
for (String temp : list) {
     System.out.println(temp);
}
```

Esercizio «Treeset&ArrayList»

- A partire dalla classe Contatore (nella slide successiva)
- **Notare l'implementazione dell'interfaccia comparableTo!**
- Implementare una classe di test *con il main* in cui

1. Istanziare 4 contatori.
2. Creare un `ArrayList` ed un `TreeSet` aggiungendo i quattro contatori ad ognuno di essi.
3. Considerare il seguente *output* che dovrà essere prodotto.

```
Stampo contatori contenuti nell'ArrayList :
3
2
1
1
Stampo contatori contenuti nel Treeset :
1
2
3
```

Classe Contatore.java

```
public class Contatore implements Comparable<Contatore> {
  private int valore;
  public void inc() { valore = valore+ 1; }
  public void reset() { valore = 0; }
  public int getValore() { return valore; }
  @Override
  public int compareTo(Contatore o) { return this.valore-o.getValore();
}
  public String toString(){ return "" + valore; }
}
```

Esercizio «Automobile»

- Si vogliono gestire gli autoveicoli registrati presso un ufficio di motorizzazione Civile. L'ufficio memorizza:
  1. `private String modello;`
  2. `private String targa;`
  3. `private int anno;`
- Si definisca la classe `Automobile` in modo che contenga:
  - `public Automobile(String modello, String targa, int anno)`
  - la definizione dei metodi `set e get` per ogni attributo
  - definisca il metodo `public boolean equals(Automobile a)`
  - implementi l'interfaccia `Comparable<Automobile>` per stabilire la precedenza con un'altra automobile (si utilizzi l'ordine alfabetico sulle targhe)

AutomobileMain.java

```
import java.util.*;

public class AutomobileMain {

  public static void main(String args[]) {
    Set<Automobile> mezzi = new TreeSet<Automobile>();
    mezzi.add(new Automobile("Ibiza", "DS 244 TC", 2009));
    mezzi.add(new Automobile("Polo", "CZ 123 QR", 2006));
    mezzi.add(new Automobile("Passat", "DZ 221 CC", 2010));
    mezzi.add(new Automobile("Punto", "EH 002 AB", 2011));
    mezzi.add(new Automobile("Passat", "DZ 221 CC", 2010));
    for (Automobile a : mezzi)
      System.out.println(a.toString());
  }
}
```

Esercizio «Motorizzazione»

- Per gestire l'ufficio di motorizzazione di una specifica città, si definisca la classe `Motorizzazione` in modo che contenga i seguenti attributi:
  1. `private String citta;`
  2. `private Set<Automobile> mezzi;`
- Definisca i seguenti metodi:
  - un costruttore `public Motorizzazione(String citta)` che azzera la lista delle automobili della motorizzazione.
  - i metodi `set e get` per ogni attributo
  - il metodo `boolean aggiungi(Automobile a)` che permette di registrare una nuova automobile (nel caso la targa sia già presente in registro, il metodo restituisce "false").
  - il metodo `boolean equals(Motorizzazione m)` per stabilire l'uguaglianza con un'altra "Motorizzazione" (l'uguaglianza va verificata solo sulla citta di ubicazione)
  - il metodo `String toString()`
  - il metodo `int quante()` che restituisce il numero di auto registrate
  - il metodo `int quanteDaRevisionare(int annocorrente)` che restituisce il numero di auto più vecchie di un anno.

MotorizzazioneMain.java

```
import java.util.*;
public class MotorizzazioneMain {
  public static void main(String args[]) {
    Motorizzazione m = new Motorizzazione("Bologna");
    mezzi.add(new Automobile("Ibiza", "DS 244 TC", 2016));
    mezzi.add(new Automobile("Polo", "CZ 123 QR", 2006));
    mezzi.add(new Automobile("Passat", "DZ 221 CC", 2010));
    Automobile a = new Automobile("Punto", "DS 244 TC", 2017);
    if (!m.aggiungi(a))
      System.out.println("L'auto è già registrata");
    System.out.println(”Auto registrate: " + m.quante());
    System.out.println(”Auto da revisionare:”+ m.quanteDaRevisionare());
  }
}
```