

# Programmazione ad Oggetti

## Collections



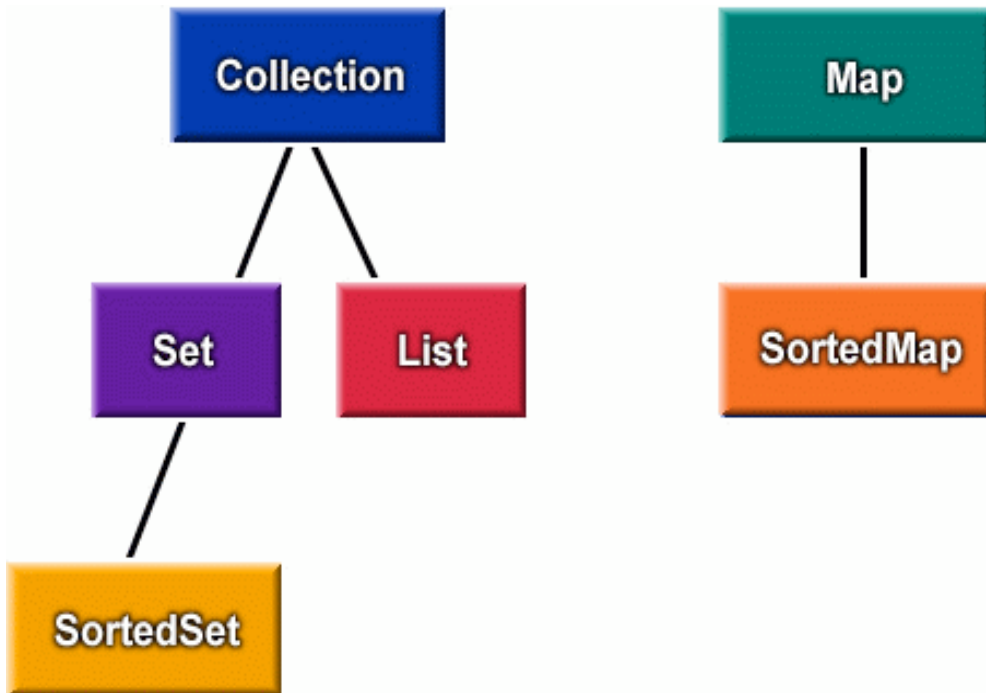
V 1.2 © Marco Torchiano 2005

## Collection

- Collezione di elementi
    - ◆ reference ad oggetti
  - Non specificano se
    - ◆ ordinati/non ordinati
    - ◆ con duplicati / senza duplicati
-

# Interfaces

---



# Iterator

---

- Astrae il problema di scandire tutti gli elementi di una collection
  - Data una collection si ottiene tramite
    - ◆ `Iterator iterator()`
  - Mantiene memoria di collection corrente e di elemento corrente in collection
    - ◆ `boolean hasNext()`
    - ◆ `T next()`
-

# Uso

---

```
Collection<Studente> c = new
    LinkedList<Studente>();
c.add(new
    Studente("10", "Rossi", "Mario"));
c.add(new
    Studente("11", "Verdi", "Giuseppe"));
Iterator it1 = c.iterator();
Studente primo = it1.next();
Studente secondo = it1.next();
```

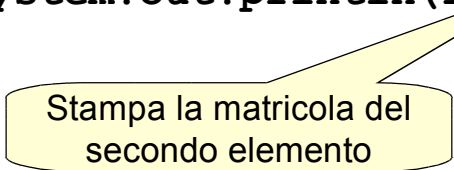
---

# Uso degli iterator

---

- Dentro un ciclo chiamare il metodo next() una sola volta
  - ◆ Ad ogni invocazione si sposta sull'elemento successivo

```
for (Iterator<Studente> iter = c.iterator();
     iter.hasNext();) {
    Studente element = iter.next();
    System.out.println(iter.next().matricola);
}
```



Stampa la matricola del secondo elemento

---

# Collection - contenimento

---

```
Collection<Studente> c;  
Studente s1 = new  
    Studente("10", "Rossi", "Mario");  
c.add(s1);  
Studente s2 = new  
    Studente("10", "Rossi", "Mario");  
if(c.contains(s2)) {  
    System.out.println("contiene");  
}else{  
    System.out.println("NON contiene");  
}
```



True o False?

---

# Collection contenimento

---

- `contains()` è vero se la collezione contiene un elemento **uguale** al parametro
  - Uguale: metodo `equals()`
  - Come funziona equals?
    - ◆ Default in Object
      - true se stesso oggetto (come ==)
    - ◆ Ridefinito nelle singole classi
      - Solitamente confronto fra attributi
-

# Metodo equals

---

```
public boolean equals(Object obj) {
    Studente other = (Studente) obj;
    return this.matricola
        .equals(
            other.matricola);
}
```

---

# Ordinamento

---

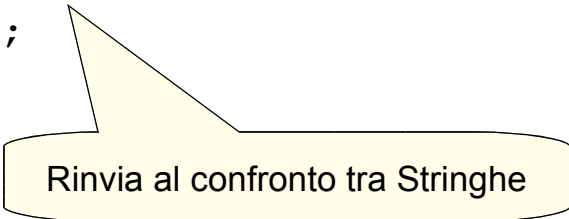
- Metodo statico: **Collections.sort()**
  - Utilizza delle liste per ordinare
  - Due approcci:
    - ◆ Uso dell'interfaccia Comparable<T> (con il metodo compareTo())
      - Collections.sort(l);
    - ◆ Uso di un Comparator<T>
      - Comparator cmp=new
      - Comparator<Studente>() { ... }
      - Collections.sort(l, cmp);
-

# CompareTo

---

- Definisce una relazione di ordine totale
- Restituisce:
  - ♦ Se `this < other` → numero negativo
  - ♦ Se `this == other` → 0 (zero)
  - ♦ Se `this > other` → numero positivo

```
public int compareTo(Studente other) {  
    return this.matricola  
        .compareTo(  
            other.matricola);  
}
```



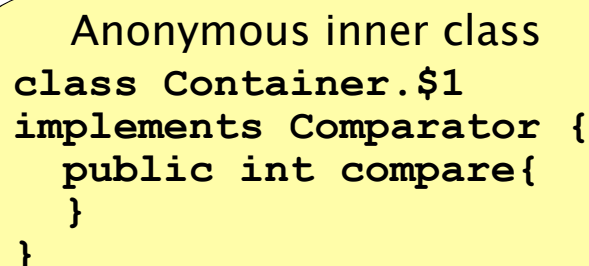
Rinvia al confronto tra Stringhe

---

# Comparator

---

```
Comparator cmp=new Comparator(){  
    public int compare(Studente s1,  
        Studente s2) {  
        return - s1.compareTo(s2);  
    }  
};
```



```
Anonymous inner class  
class Container.$1  
implements Comparator {  
    public int compare{  
    }  
}
```

---

# Ordinamento di Collection

---

- Una Collection di per se non è ordinabile
- Occorre “*convertirla*” in una List

```
Collection<Studente> coll;
```

...

```
List<Studente> list = new  
    ArrayList<Studente>(coll);  
Collections.sort(list)
```

---

# Inserimento ordinato

---

- Per avere una lista ordinata
- Soluzione 1:

- ♦ Inserimento casuale + Ordinamento

```
- List<Studente> l = new ArrayList<Studente>();  
- l.add(element)  
- Collections.sort(l);  
- return l;
```

- Soluzione 2:

- ♦ Inserimento ordinato

```
- TreeSet<Studente> s =new TreeSet<Studente>();  
- s.add(element)  
- return s;
```

N.B. i “tree” usano  
`compareTo()` per  
ordinare gli elementi

---

# Map

---

- Si usano le mappe quando data una chiave si vuole ottenere il valore (oggetto) corrispondente

- Inserimento

```
studenti.put(s1.matricola, s1);
```

- Ricerca

```
Studente s = studenti.get(matricola);
```

---

## Map – Implementazioni

---

- L'interfaccia Map è implementata da:

- ♦ TreeMap
- ♦ HashMap

- Dichiarazione di variabili Map

```
Map<String, Studente> studenti =  
    new HashMap<String, Studente>();
```

- Elenco oggetti nella map:

```
Collection<Studente> st =  
    studenti.values();
```

– Già ordinata se si usa una TreeMap

---