

Java Collections



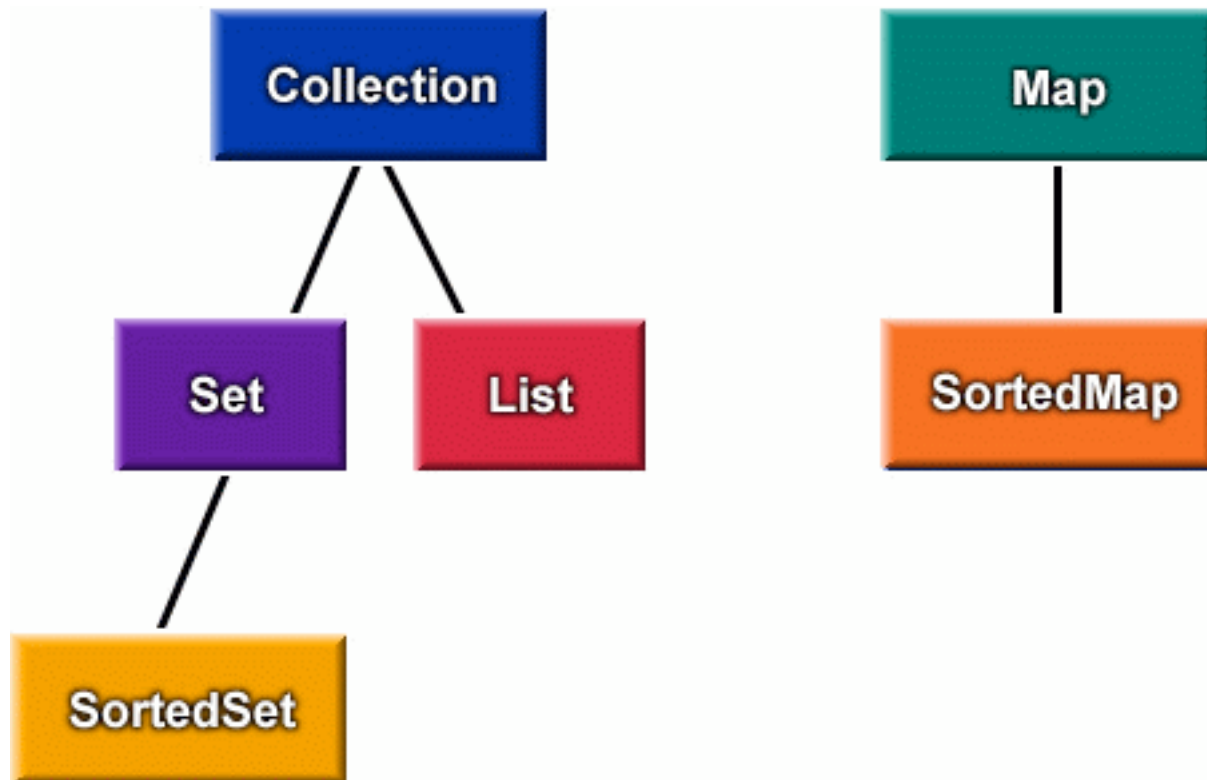
Introduzione

- Una java collection (a volte chiamata anche container) è un oggetto che raggruppa più elementi dello stesso tipo in una singola unità.
- Tipicamente è utilizzata per raggruppare oggetti che nella realtà si trovano spesso insieme (es. i recapiti nella rubrica telefonica, le mail di un casella di posta, etc...)

Metodi di una Collection

```
public interface Collection<E> {  
    ...  
    int size();  
    boolean isEmpty();  
    boolean contains(E element);  
    boolean add(E element);    // Optional  
    boolean remove(E element); // Optional  
    Iterator iterator();  
    ...  
}
```

Uno sguardo d'insieme



Lista

- Una lista (o sequenza) è una collezione ordinata di oggetti.
- In una lista gli ogni elemento è caratterizzato da:
 - una posizione
 - un elemento che lo precede
 - un elemento che lo segue
- Una lista può contenere elementi duplicati.

L'interfaccia `List<E>`

Oltre alle operazioni ereditate dall'interfaccia `Collection`, l'interfaccia `List` include operazioni per l'accesso posizionale: gli elementi possono essere manipolati specificando la loro posizione nella sequenza.

List: Accesso posizionale

```
E get(int index);  
E set(int index, E element);  
void add(int index, E element);  
E remove(int index);  
abstract boolean addAll(int index,  
    Collection c);
```

List: Search

```
int indexOf(Object o);  
int lastIndexOf(Object o);
```


List: Iteration

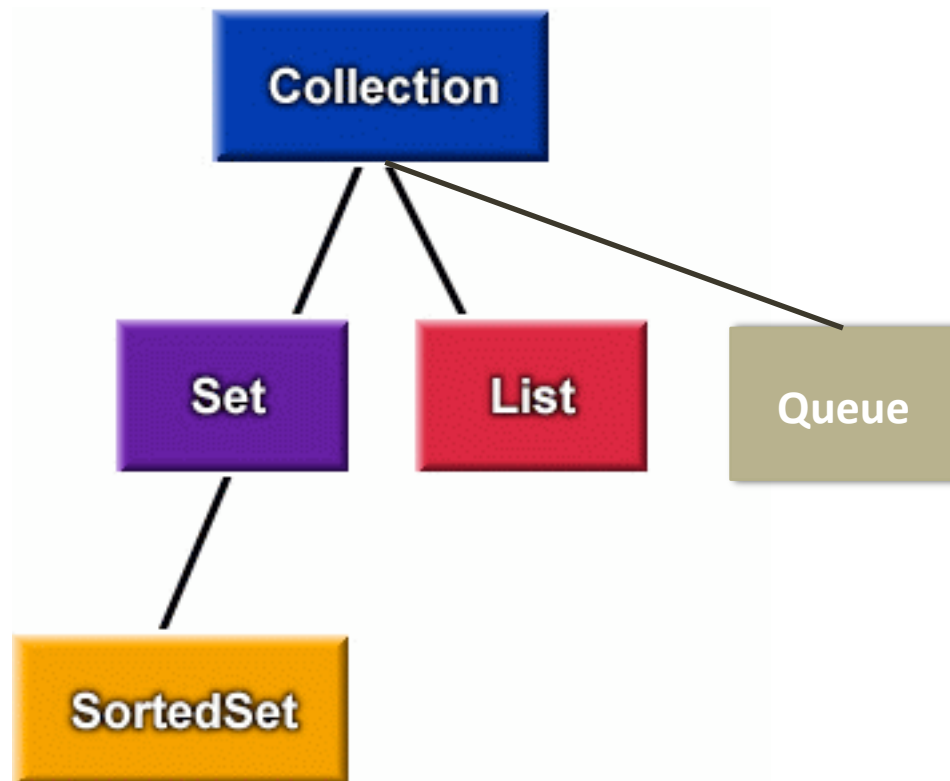
```
ListIterator listIterator();
```

```
ListIterator listIterator(int index);
```

List: Range-view

```
List subList(int from, int to);
```

Uno sguardo d'insieme



Coda

- Una coda è una collezione progettata per mantenere oggetti che devono essere processati.
- Gli elementi in una coda sono mantenuti secondo un certo ordine:
 - FIFO (es. coda alla posta)
 - LIFO (es. stack)
- La testa (o head) della coda è l'elemento che verrà rimosso da una chiamata al metodo `remove()` o `poll()`.

L'interfaccia `Queue<E>`

Oltre alle operazioni ereditate dall'interfaccia `Collection`, l'interfaccia `Queue` include operazioni per

- accedere e/o rimuovere l'elemento che si trova in testa
- aggiungere un elemento in testa o in coda
- Inserire un elemento alla fine (*tail*) della coda (ordinamento FIFO) o in testa (ordinamento LIFO)

Queue<E>

E **element()**

Retrieves, but does not remove, the head of this queue

boolean **offer(E o)**

Inserts the specified element into this queue, if possible.

E **peek()**

Retrieves, but does not remove, the head of this queue, returning null if this queue is empty.

E **poll()**

Retrieves and removes the head of this queue, or null if this queue is empty.

E **remove()**

Retrieves and removes the head of this queue.

DALL'INTERFACCIA
ALL'IMPLEMENTAZIONE

La classe `LinkedList<T>`

- `public class LinkedList<T> ...
 implements List<T>, Deque<T>, ...`
- La `LinkedList` implementa la struttura dati facendo uso di reference:

```
/**  
 * The first element in the list.  
 */  
Entry<T> first;
```

```
/**  
 * The last element in the list.  
 */  
Entry<T> last;
```


La classe `ArrayList<T>`

```
public class ArrayList<E> ...
```

```
    implements List<E>, RandomAccess, ...
```

Implementa le operazioni di una lista utilizzando un array.

Conclusioni

- Differenza tra interfaccia e implementazione
- LinkedList vs. ArrayList