Spiegone Recap relazioni tra oggetto originale e oggetti ritornati da metodi.

Per tutti, le copie sono shallow copies, copio il puntatore, non creo un'istanza separata dell'oggetto. Questo viene espresso con "the X is backed by the Y".

Dal punto di vista strutturale la cosa e' un po' piu' variegata.

Map

entrySet

`public Set entrySet()`

Returns a set view of the mappings contained in this map. Each element in the returned set is a `Map.Entry`. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress, the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Set.remove`, `removeAll`, `retainAll` and `clear` operations. It does not support the `add` or `addAll` operations.

In soldoni:

1) Se la mappa e' modificata mentre sto scorrendo il set -> risultato indefinito (che vuol dire decide l'implementazione quello che succede MA deve essere documentato).
2) se rimuovo dal set -> rimuovo anche dalla mappa
3) Non sono supportate le aggiunte al set (ne add ne addAll)

keySet

`public Set keySet()`

Returns a set view of the keys contained in this map. The set is backed by the map, so changes to the map are reflected in the set, and vice-versa. If the map is modified while an iteration over the set is in progress, the results of the iteration are undefined. The set supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Set.remove`, `removeAll` `retainAll`, and `clear` operations. It does not support the add or `addAll` operations.

Returns:

a set view of the keys contained in this map.

In soldoni:

1) Se la mappa e' modificata mentre sto scorrendo il set -> risultato indefinito (che vuol dire decide l'implementazione quello che succede MA deve essere documentato).

2) se rimuovo dal set -> rimuovo anche dalla mappa LA ENTRY (coppia chiave valore)
3) Non sono supportate le aggiunte al set (ne add ne addAll)

values

```
public Collection values()
```

Returns a collection view of the values contained in this map. The collection is backed by the map, so changes to the map are reflected in the collection, and vice-versa. If the map is modified while an iteration over the collection is in progress, the results of the iteration are undefined. The collection supports element removal, which removes the corresponding mapping from the map, via the `Iterator.remove`, `Collection.remove`, `removeAll`, `retainAll` and `clear` operations. It does not support the add or `addAll` operations.

Returns:

a collection view of the values contained in this map.

In soldoni:

1) Se la mappa e' modificata mentre sto scorrendo la collection -> risultato indefinito (che vuol dire decide l'implementazione quello che succede MA deve essere documentato).
2) se rimuovo dalla collection -> rimuovo anche dalla mappa LA ENTRY (coppia chiave valore)
3) Non sono supportate le aggiunte al set (ne add ne addAll)

N.B. la similitudine nel comportamento documentato potrebbe suggerire qualcosa per l'implementazione.

List

subList

```
public List subList(int fromIndex,
                    int toIndex)
```

Returns a view of the portion of this list between the specified `fromIndex`, inclusive, and `toIndex`, exclusive. (If `fromIndex` and `toIndex` are equal, the returned list is empty.) The returned list is backed by this list, so non-structural changes in the returned list are reflected in this list, and vice-versa. The returned list supports all of the optional list operations supported by this list.

This method eliminates the need for explicit range operations (of the sort that commonly exist for arrays). Any operation that expects a list can be used as a range operation by passing a subList view instead of a whole list. For example, the following idiom removes a range of elements from a list:

```
list.subList(from, to).clear();
```

Similar idioms may be constructed for `indexOf` and `lastIndexOf`, and all of the algorithms in the `Collections` class can be applied to a subList.

The semantics of the list returned by this method become undefined if the backing list (i.e., this list) is *structurally modified* in any way other than via the returned list. (Structural modifications are those that change the size of this list, or otherwise perturb it in such a fashion that iterations in progress may yield incorrect results.)

Parameters:

`fromIndex` - low endpoint (inclusive) of the subList.

`toIndex` - high endpoint (exclusive) of the subList.

Returns:

a view of the specified range within this list.

Throws:

IndexOutOfBoundsException - for an illegal endpoint index value (fromIndex < 0 || toIndex > size || fromIndex > toIndex).

In soldoni:

1) Sempre shallow copy (backed by the orginating list)
2) Le modifiche strutturali devono essere propagate dalla sublist alla lista di provenienza
3) Le modifiche strutturali sulla lista di provenienza -> comportamento indefinito per la sublist, che, come prima, vuol dire implementate come volete e documentate quello che succede.

toArray

`public Object[] toArray()`

Returns an array containing all of the elements in this list in proper sequence. Obeys the general contract of the `Collection.toArray` method.

Specified by:

toArray in interface Collection

Returns:

an array containing all of the elements in this list in proper sequence.

In soldoni:

1) Vediamo cosa dice il metodo di Collection.

Collection

toArray

`public Object[] toArray()`

Returns an array containing all of the elements in this collection. If the collection makes any guarantees as to what order its elements are returned by its iterator, this method must return the elements in the same order.

The returned array will be "safe" in that no references to it are maintained by this collection. (In other words, this method must allocate a new array even if this collection is backed by an array). The caller is thus free to modify the returned array.

This method acts as bridge between array-based and collection-based APIs.

Returns:

an array containing all of the elements in this collection

In soldoni:

1) Shallow copy
2) Stesso ordine della collezione originale (se itero l'originale e se scorro l'array vedo la stessa sequenza)
3) Modifiche strutturali all'array non si propagano sull'originale
4) Modifiche strutturali all'originale non si propagano sull'array