Langages de programmation – TP7

Objectifs:

- Generics et collections en Java
  - Une collection permet à un groupe d'objets à traiter comme une seule unité. Les objets peuvent être stockés, récupérés et manipulés comme des éléments d'une collection. Le "Java Collections Framework" (en java.util paquet) fournit un ensemble de classes d'utilitaires standard pour la gestion des différents types de collections.
- Iterators
  - Une collection fournit un itérateur qui permet un accès séquentiel aux éléments d'une collection. Un itérateur peut être obtenu en appelant la method suivante de l'interface Collections: Iterator<E> iterator()

```
Vector<Integer> v = new Vector();
    v.add(2);
    v.add(8);
    v.add(4);
Iterator it = v.iterator();
    while (it.hasNext())
    {
       System.out.println(it.next());
    }
```

- Static import

```
import static java.lang.System.out;
……
out.println("Hola");
```

- Comparer des objets: mise en œuvre des metodes *equals*, *hashCode*, *compareTo* et *compare*

Problème en classe:

Implémentez une classe *VersionNumber* (avec les attributs *releaseNumber*, *revisionNumber*, *patchNumber* de type *Integer*). Créez 5 objets de cette classe avec 2 objets ayant des valeurs égales.

- Mettez-les dans un tableau de type *VersionNumbe*r et recherchez d'un certain objet, qui est dans le tableau.
- Transformez le tableau en *List* et recherchez d'un certain objet, qui est dans la *List*.
- Transformez la *List* en *TreeSet* et sortez-le. Recherchez d'un certain objet, qui est dans le *TreeSet*.
- Créez une *HashMap* :
  Map<VersionNumber, Integer> versionStatistics = new HashMap<>();
  Recherchez d'un certain objet, qui est dans la *HashMap*.

Devoir:

1. Pensez à une façon différente de la modélisation d'une classe Polynom, en utilisant une *List*. Mettez en œuvre les méthodes pour ajouter et soustraire deux polynômes.

2. Étendez (Extend) la classe ArrayList <Integer> avec une clase MyArrayList en fournissant une méthode qui trie les éléments de la liste dans l'ordre naturel inverse - MySort. Testez la méthode.

3. Ecrivez un programme qui lit à partir du clavier deux ensembles de nombres entiers et affichez la réunion, intersection, différence symétrique et produit cartésien pour les ensembles. Utilisez la classe HashSet pour représenter des ensembles.

Attachments

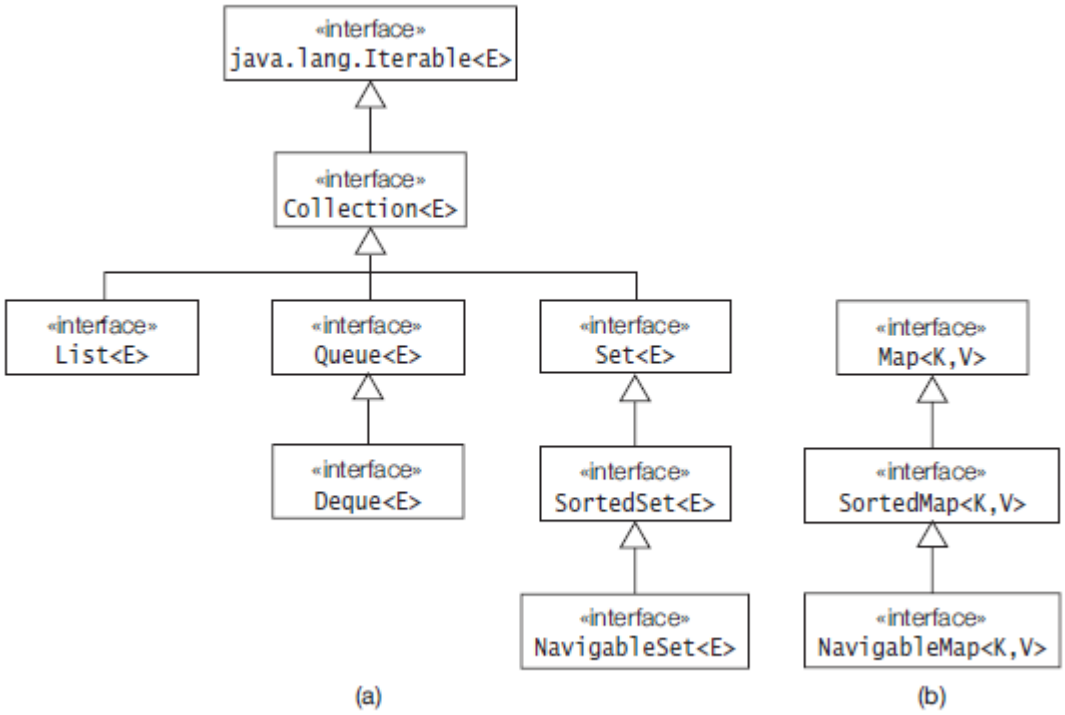


Table 15.1 *Core Interfaces in the Collections Framework*

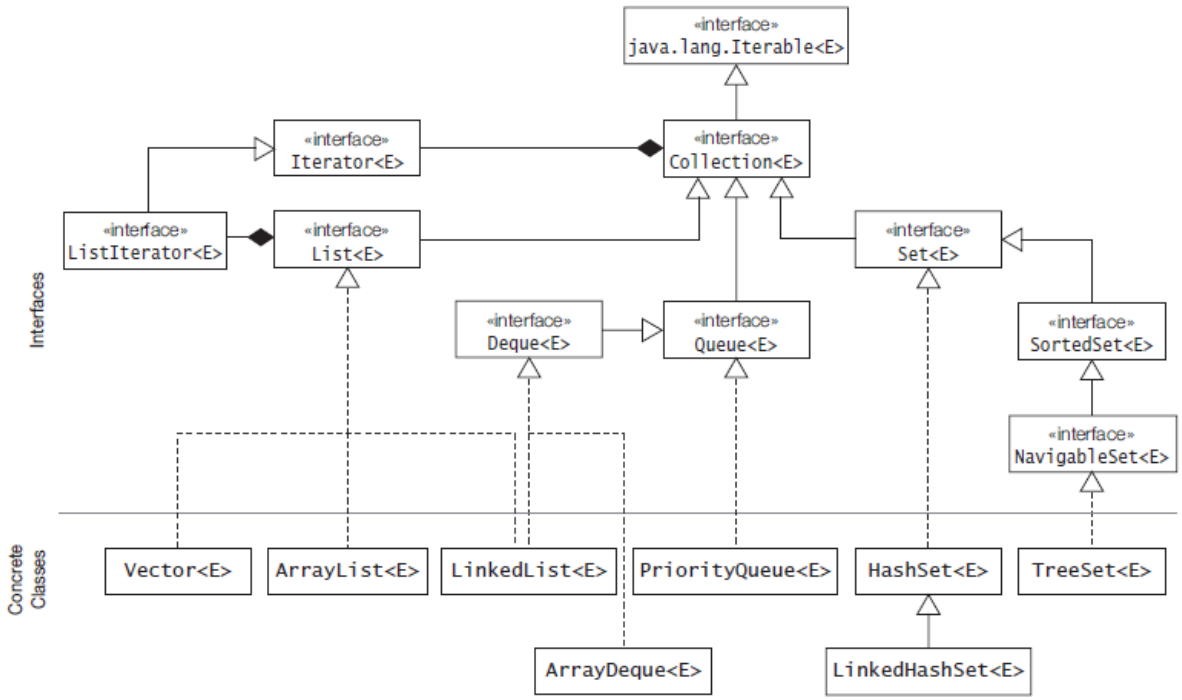
Interface	Description	Concrete Classes
`Collection<E>`	A basic interface that defines the normal operations that allow a collection of objects to be maintained or handled as a single unit.	
`Set<E>`	The `Set` interface extends the `Collection` interface to represent its mathematical namesake: a *set* of unique elements.	`HashSet<E>` `LinkedHashSet<E>`
`SortedSet<E>`	The `SortedSet` interface extends the `Set` interface to provide the required functionality for maintaining a set in which the elements are stored in some sorted order.	`TreeSet<E>`
`NavigableSet<E>`	The `NavigableSet` interface extends and replaces the `SortedSet` interface to maintain a sorted set, and should be the preferred choice in new code.	`TreeSet<E>`

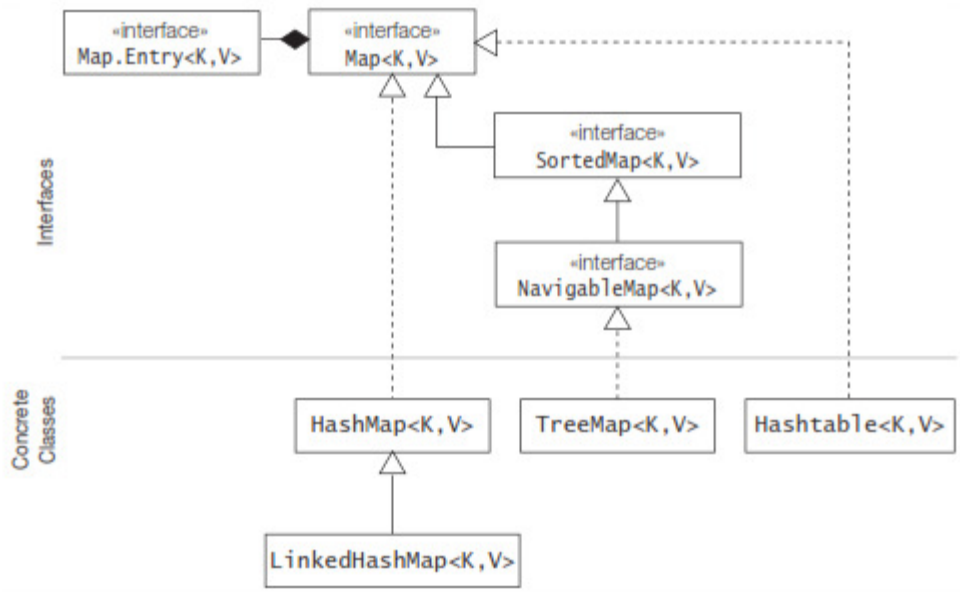
`List<E>`	The `List` interface extends the `Collection` interface to maintain a sequence of elements that can contain duplicates.	`ArrayList<E>`
		`Vector<E>`
		`LinkedList<E>`

`Queue<E>`	The `Queue` interface extends the `Collection` interface to maintain a collection whose elements need to be processed in some way, i.e., insertion at one end and removal at the other, usually as FIFO (First-In, First-Out).	`PriorityQueue<E>` `LinkedList<E>`
`Deque<E>`	The `Deque` interface extends the `Queue` interface to maintain a queue whose elements can be processed at both ends.	`ArrayDeque<E>` `LinkedList<E>`

`Map<K,V>`	A basic interface that defines operations for maintaining mappings of keys to values.	`HashMap<K,V>`
		`Hashtable<K,V>`
		`LinkedHashMap<K,V>`

`SortedMap<K,V>`	The `SortedMap` interface extends the `Map` interface for maps that maintain their mappings sorted in key order.	`TreeMap<K,V>`
`NavigableMap<K,V>`	The `NavigableMap` interface extends and replaces the `SortedMap` interface for sorted maps.	`TreeMap<K,V>`


«interface»
java.lang.Iterable<E>
«interface»
Iterator<E>
«interface»
Collection<E>
«interface»
ListIterator<E>
«interface»
List<E>
«interface»
Set<E>
«interface»
Deque<E>
«interface»
Queue<E>
«interface»
SortedSet<E>
«interface»
NavigableSet<E>
Interfaces
Concrete Classes
Vector<E>
ArrayList<E>
LinkedList<E>
PriorityQueue<E>
HashSet<E>
TreeSet<E>
ArrayDeque<E>
LinkedHashSet<E>


«interface»
Map.Entry<K,V>
«interface»
Map<K,V>
«interface»
SortedMap<K,V>
«interface»
NavigableMap<K,V>
Interfaces
Concrete Classes
HashMap<K,V>
TreeMap<K,V>
Hashtable<K,V>
LinkedHashMap<K,V>