Java EE Spring, prêt à l'emploi

Collections Java

Guillaume Dufrêne – Lionel Seinturier – Julien Wittouck

Université de Lille

Stocker des données en Java

- Dans un fichier
- En mémoire
  - Variable par ex. `String s = "Hello world!"`
  - Tableau par ex. `int[] tab = new int[10]`
  - Liste, ensemble (List, Set)

Stocker des données en Java

- En mémoire
  - Liste, ensemble (List, Set) => collection
  - Table de hachage (Map)
- Définition dans `java.util`
- Depuis Java 5, utilisation des types génériques

```
java.util.Collection<E>
java.util.Map<K,V>
```

java.util.Collection

Extrait de Javadoc

A **collection** represents a group of objects.

The JDK does not provide any direct implementations of this interface: it provides implementations of more specific subinterfaces like Set and List.

java.util.Collection

Méthodes principales

```
public interface Collection<E> {
	boolean add( E e );
	boolean remove( Object o );
	boolean contains( Object o );
	int size();
	Iterator<E> iterator();
	...
}
```

+ addAll, clear, removeAll, removeIf, toArray, ...

java.util.List

Extrait de Javadoc

An ordered collection.

The user can access elements by their integer index (position in the list), and search for elements in the list.

java.util.List

Méthodes principales

```
public interface List<E> extends Collection<E> {
	void add(int index, E element);
	E get(int index);
	E remove(int index);
	int indexOf(Object element);
	...
}
```

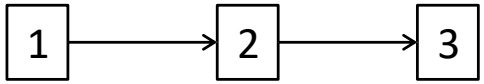
+ sort, subList, ...

java.util.List

2 classes principales implémentant `List`

- `ArrayList` : implémentée avec un tableau redimensionnable

- `LinkedList` : implémentée avec une chaîne de références



java.util.List

Pourquoi plusieurs implémentations ?

- Selon les opérations, l'efficacité n'est pas la même
- Accès au k-ème élément de la liste
  - rapide avec `ArrayList`
  - plus coûteux avec `LinkedList`
- Insertion d'un élément au milieu de la liste
  - rapide avec `LinkedList`
  - plus coûteux avec `ArrayList`

⇨ Choix dépend des fréquences d'utilisation de ces opérations

java.util.Set

Extrait Javadoc

A collection that contains no duplicate elements.

Remarque : différence entre `List` et `Set`

Classes principales implémentant `Set`

- `HashSet`
- `TreeSet` : les éléments sont triés

java.util.Map

Table de hachage (Map)

- stocker des couples < clé , valeur >
- clé doit être unique
- recherche efficace à partir de la clé
- parcours des clés et des valeurs

java.util.Map

Méthodes principales

```
public interface Map<K,V> {
        V put(K key, V value);
        V get(Object o);
        boolean containsKey(Object key);
        boolean containsValue(Object key);
        int size();
        Set<K> keySet();
        Collection<V> values();
        ...
}
```

+ entrySet, putAll, remove, replace, clear, ...

java.util.Map

Exemple d'utilisation : parcours d'une map

```
Map<Integer,String> m = new HashMap<>();
m.put(1,"one");
m.put(42,"forty two");

for (Integer key : m.keySet()) {
   String value = m.get(key);
   ...
}
```

Compléments

- multi-threading
  - par défaut collections non synchronisées
  - synchronisées
    - protection accès concurrents
    - `Collections.synchronisedSet`

- modifications
  - par défaut modifiables
  - non modifiables
    - en lecture seule
    - "garantie" pour éviter les erreurs
    - `Collections.unmodifiableList, List.of`

Différentes implémentations des collections et des maps

General purpose implementations

Interface	Hash Table	Resizable Array	Balanced Tree	Linked List	Hash Table + Linked List
Set	HashSet		TreeSet		LinkedHashSet
List		ArrayList		LinkedList	
Deque		ArrayDeque		LinkedList	
Map	HashMap		TreeMap		LinkedHashMap

https://docs.oracle.com/en/java/javase/12/docs/api/java.base/java/util/doc-files/coll-overview.html

API Streams

- Introduite avec Java 8
- Programmation fonctionnelles sur les éléments des collections
  - Filtrage, convertion, comptage, tri, MapReduce
  - Parcours en séquence, en parallèle

```
interface Collection<E> {
    Stream<E> stream();
    ...
}
```

En résumé

- Stocker les données d'un programme Java
- Listes (`List`), ensembles (`Set`), tables de hachage (`Map`)
- Différentes implémentations avec différentes propriétés
- Utilise les types génériques Java
- Synchronisées ou non
- Modifiables ou non