13/10/2025

# CM POO : Collection et table de hachage

Toutes les structures de données vues sont dans le paquetage `java.util`

## Collection

> Les collections prennent des objets en paramètres, pas les types primitifs donc :)

Une collection est un regroupement d'objets (listes, ensembles). Les collections sont typées

Les collections sont définis par une interface `java.util.Collection<E>`, où `E` représente le type des éléments de la collection

Quelques méthodes de `Collection`

- `boolean add(E e)`
- `boolean contains(Object o)`
- `boolean isEmpty()`
- `Iterator<E> iterator()` → Renvoie un itérateur sur les éléments de la collection
- `boolean remove(Object o)`
- `int size()`

> ⚠ Warning
>
> Ne pas confondre l'interface `Collection` avec la classe `Collections`

## Liste

`List<E>` est une interface qui expose une collection ordonnée d'objets

Deux classes l'implémentent :

- `ArrayList<E>`
- `LinkedList<E>` : Liste doublement chaînée

Voici les méthodes que `List` expose :

- `void add(int index, E el)`
- `E get(int index)`

- `E remove(int index)`
- `int indexOf(Object element`

## Ensembles

L'interface `Set` est implémentée par deux classes :

- `HashSet<E>`
- `TreeSet<E>`

D'une manière générale, on préfère typer ces classes par les interfaces, pour simplifier les changements de classes

## Les itérateurs

`java.util.Iterator<E>` définit trois méthodes :

- `boolean hasNext()`
- `E next()`
- `void remove()`

Ne jamais parcourir une liste une get, toujours utiliser un itérateur, c'est beaucoup plus efficace :)

## Tables de hachage

> ⚠ Warning
>
> Les "Map" ne sont pas des `Collection`

Les tables de hachages sont des listes associatives, des dictionnaires

On a :

- `HashMap<K, V>`
- `TreeMap<K, V>`

Les méthodes dessus sont :

- `V get(K key)`
- `void put(k key, V value)`
- `boolean containsKey(Object key)`
- `boolean containsValue(Object value)`
- `Collection<V> values()` …