



Le langage Java

Les sets

Programme détaillé ou sommaire

Les sets

Les sets les plus courants

Ajout d'élément

Parcours d'un set

Gestion des doublons

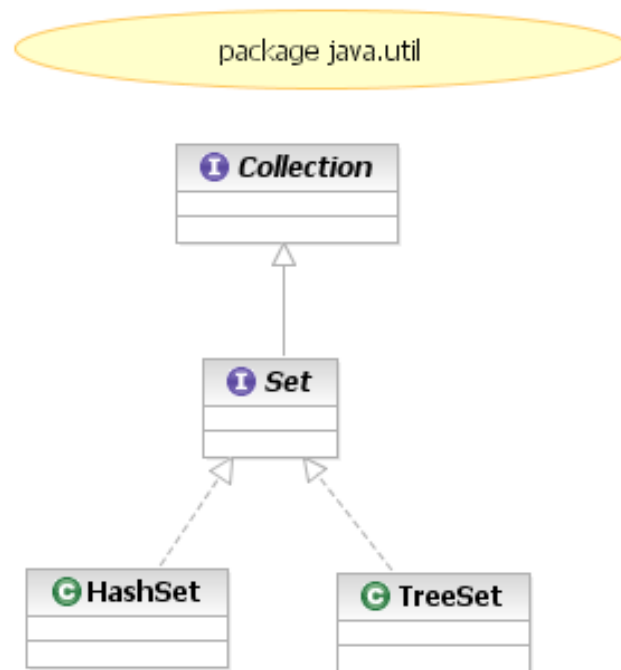
Les sets

Les sets sont des collections

- Implémentent l'interface **java.util.Set** qui hérite de l'interface **java.util.Collection**

Les sets ne sont pas indexés

- il n'existe pas de méthode **get(int i)** permettant de récupérer le $i^{\text{ème}}$ élément du set.



Les sets les plus courants

HashSet

- **Implémentation de base** de l'interface Set
- Ne préserve pas l'ordre d'insertion des éléments
 - Algorithme de stockage basé sur le **hashcode** de l'objet (valeur entière) => clé de hachage
 - Dans un HashSet les objets sont regroupés en agglomérats, selon leur clé de hachage.

TreeSet

- Les éléments sont triés au fur et à mesure de l'insertion

Ajout d'éléments

Les éléments sont ajoutés avec la méthode **add(...)**

- Les éléments ne sont pas indexés.
- L'ordre d'insertion n'est pas préservé (dépend de la clé de hachage)
- Tout type d'objet peut être inséré dans un set.

```
HashSet<String> set = new HashSet<>();  
set.add("le");  
set.add("petit");  
set.add("chat");
```

Suppression d'élément

Méthode **remove(...)**

- Suppression de l'objet passé en paramètre

```
HashSet<User> set = new HashSet<>();  
User u1 = new User("jean", "dupont");  
set.add(u1);  
User u2 = new User("jean", "durand");  
set.add(u2);  
User u3 = new User("jean", "martin");  
set.add(u3);  
  
set.remove(u3);           // suppression de la référence vers u3
```

Parcours d'un set (1/2)

Utilisation d'un Iterator pour le parcours

```
HashSet<User> set = new HashSet<>();  
User u1 = new User("jean", "dupont");  
set.add(u1);  
User u2 = new User("jean", "durand");  
set.add(u2);  
User u3 = new User("jean", "martin");  
set.add(u3);  
  
Iterator<User> iterator = set.iterator();  
while (iterator.hasNext()) {  
    User myUser = iterator.next();  
    System.out.println(myUser);  
}
```

Parcours d'un set (2/2)

Utilisation d'une boucle objet


```
HashSet<User> set = new HashSet<>();  
User u1 = new User("jean", "dupont");  
set.add(u1);  
User u2 = new User("jean", "durand");  
set.add(u2);  
User u3 = new User("jean", "martin");  
set.add(u3);  
  
for (User myUser : set) {  
    System.out.println(myUser);  
}
```


Gestion des doublons

Les Sets n'acceptent pas les doublons

- Si on ajoute un élément plusieurs fois, il ne sera présent qu'une seule fois dans le set.

```
HashSet<User> set = new HashSet<>();  
User u1 = new User("jean", "dupont");  
set.add(u1);  
set.add(u1); //sans effet
```

 La méthode `add(...)` renvoie un booléen. Elle renvoie `false` si l'élément était déjà présent.

Suppression d'élément: ConcurrentModificationException

Exception renvoyée par la méthode remove(...)

- Suppression d'un élément dans une collection en cours de parcours (boucle).

```
Set<User> set = new HashSet<>();
set.add(new User("jean", "dupont"));
set.add(new User("jean", "durand"));
set.add(new User("jean", "martin"));
set.add(new User("marcel", "ferrand"));

for (User user : set) {
    if (user.getNom().equals("durand")) {
        set.remove(user);
    }
}
```



Eviter la ConcurrentModificationException

Parcourir la collection avec un **iterator** et utiliser la méthode **remove()**

```
Set<User> set = new HashSet<>();
set.add(new User("jean", "dupont"));
set.add(new User("jean", "durand"));
set.add(new User("jean", "martin"));
set.add(new User("marcel", "ferrand"));

Iterator<User> iter = set.iterator();
while (iter.hasNext()) {
    User user = iter.next();
    if (user.getNom().equals("durand")) {
        iter.remove();
    }
}
```



Atelier (TP)

Objectifs du TP: manipuler les collections et plus particulièrement les Set et HashSet

Description du TP:

Dans ce TP, vous allez créer divers sets et apprendre à les utiliser.