Le langage Java

Les maps

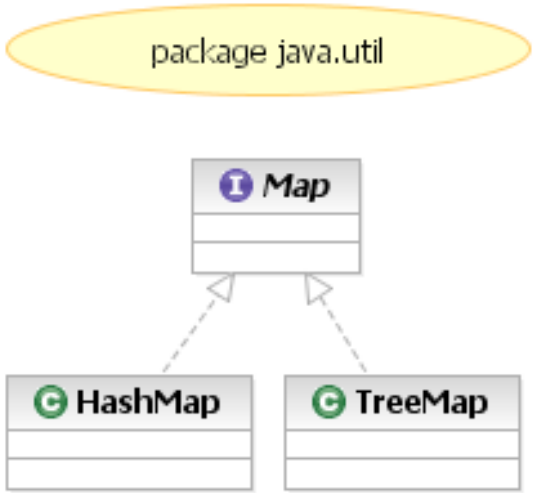
Programme détaillé ou sommaire

Présentation
Les Maps
Les Maps les plus courantes
Ajout et récupération d'éléments
Suppression d'élements
Parcours d'une map
Gestion des doublons

Les Maps

Associations clé-valeur

- Chaque élément est stocké avec **une clé**
- Conceptuellement, une Map est aussi un ensemble d'éléments.



Les maps les plus courantes

HashMap

- **Implémentation de base** de l'interface Map

TreeMap

- Les éléments sont triés en fonction de la clé de stockage

LinkedHashMap

- Les éléments sont stockés dans l’ordre d’insertion

Ajout et récupération d'éléments

Pour ajouter un élément:

put(Object key, Object value)

Pour extraire un élément:

get(Object key)

```
User u1 = new User("jean", "dupont");
User u2 = new User("marcel", "durand");

HashMap<String, User> map = new HashMap<>();
// association d'une clé à chaque élément
map.put("aaa", u1);
map.put("bcd", u2);

// récupération en fonction de la clé
User u1bis = map.get("aaa");
```

Suppression d'élément

Pour supprimer un élément:

remove(Object key)

```
User u1 = new User("jean", "dupont");
User u2 = new User("marcel", "durand");

HashMap<String, User> map = new HashMap<>();
// association d'une clé à chaque élément
map.put("aaa", u1);
map.put("bcd", u2);

// suppression
map.remove("aaa");
```

Attention à ne pas passer l'objet mais la clé en paramètre

Parcours d'une Map avec un Iterator (1/3)

Problématique particulière

- S'agit-il de parcourir les **clés** ou les **valeurs** ?
  - Méthode **Map.keySet()** : retourne les clés dans un **Set**
  - Méthode **Map.values()** : retourne les valeurs dans une **Collection**

```
HashMap<Integer, User> map = new HashMap<>();

Iterator<Integer> keysIte = map.keySet().iterator();

Iterator<User> valuesIte = map.values().iterator();
```

Parcours d'une Map avec un Iterator (2/2)

Parcours des valeurs avec un Iterator:

```
HashMap<String, User> map = new HashMap<>();
User user1 = new User("jean", "dupont");
User user2 = new User("marcel", "durand");

map.put("aaa", user1);
map.put("bcd", user2);

Iterator<User> valuesIte = map.values().iterator();
while (valuesIte.hasNext()) {
    User user = valuesIte.next();
    // …
}
```

Parcours d'une Map avec un Iterator (3/3)

Parcours des clés avec un Iterator:

```
HashMap<String, User> map = new HashMap<>();
User user1 = new User("jean", "dupont");
User user2 = new User("marcel", "durand");

map.put("aaa", user1);
map.put("bcd", user2);

Iterator<String> keysIte = map.keySet().iterator();
while (keysIte.hasNext()) {
    String key = keysIte.next();
    // …
}
```

Parcours d'une Map avec une boucle (1/2)

Parcours des valeurs avec une boucle objet:

```
HashMap<String, User> map = new HashMap<>();
User user1 = new User("jean", "dupont");
User user2 = new User("marcel", "durand");

map.put("aaa", user1);
map.put("bcd", user2);

for (User user : map.values()) {

}
```

Parcours d'une Map avec une boucle (2/2)

Parcours des clés avec une boucle objet:

```
HashMap<String, User> map = new HashMap<>();
User user1 = new User("jean", "dupont");
User user2 = new User("marcel", "durand");

map.put("aaa", user1);
map.put("bcd", user2);

for (String cle : map.keySet()) {

}
```

Gestion des doublons

Les maps acceptent les doublons :

- **Seulement** s'ils sont référencés par des **clés différentes**.
- Dans le cas de l’ajout d'un élément avec une clé existant déjà :

  Si la clé était déjà présente dans la map, la précédente valeur est écrasée.

Atelier (TP)

Objectifs du TP: manipuler les collections et plus particulièrement les Map et HashMap

Description du TP:

Dans ce TP, vous allez créer diverses maps et apprendre à les utiliser.