Introduction programmation Java

Cours 8: Map et Notion d'ordre

Stéphane Airiau

Université Paris-Dauphine

Map

un `Map` représente une relation binaire : chaque élément d'un `Map` est une paire entre une clé et une valeur.

Dans un `Map`, chaque clé est unique, mais on peut avoir des doublons pour les valeurs.

Attention, `Map` n'est pas une sous interface de `Iterable`, donc on ne peut pas parcourir un `Map` avec une boucle `for each` !

On peut obtenir l'ensemble des clés, l'ensemble des valeurs, et l'ensemble des paires (clé,valeur) grace aux méthodes suivantes :

- `Set<K> keySet()`
- `Set<Map.Entry<K,V» entrySet()`
- `Collection<V> values()`

`Map.Entry` désigne une classe `Entry` qui est interne à la classe `Map`. On peut créer des classes à l'intérieur de classes, mais je n'en parlerai pas plus aujourd'hui.

Exemple parcours d'une `Map`

```
1  Map<Personnage,Region> origines = new HashMap<>();
2  ...
3  for (Map.Entry<Personnage,Region> paire: origines.entrySet()){
4     Personnage p = paire.getKey();
5     Region r = paire.getValue();
6     if (r.getName().equals("Ibère"))
7       System.out.println(p);
8   }
```

Dans cet exemple, on part une `Map` qui associe à chaque personnage sa région d'origine et on affiche seulement les personnages qui sont des ibères.

Ordre

Interface `Comparable` contient une seule méthode :

```
public int compareTo(T o)
```

Cette méthode retourne

- un entier négatif si l'objet est plus petit que l'objet passé en paramètre
- zéro s'ils sont égaux
- un entier positif si l'objet est plus grand que l'objet passé en paramètre.

les classes `String`, `Integer`, `Double`, `Date`, `GregorianCalendar` et beaucoup d'autres implémentent toutes l'interface `Comparable`.

Exemple

```
1  public class Gaulois extends Personnage
2                  implements Comparable<Gaulois>{
3    String nom;
4    int quantiteSanglier;
5      ...
6
7    public int compareTo(Gaulis ixis) {
8      return this.quantiteSanglier - ixis. quantiteSanglier;
9    }
10 }
```

interface `Comparator`

Une classe qui implémente l'interface comparator représente une notion d'ordre.
A priori, on n'a besoin d'implémenter une seule méthode, la méthode pour comparer deux éléments.

```
1 public interface Comparator<T> {
2    int compare(T o1, T o2);
3    boolean equals(Object obj);
4 }
```

Pour comparer des `Gaulois`, et même tous les `Personnage` selon leur taille, on peut écrire la classe suivante :

```
1 public class OrdreHauteur implements Comparator<Personnage> {
3     public int compare(Personnage gauche, Personnage droit){
4       return gauche.hauteur < droite.hauteur ? -1:
5             (gauche.hauteur == droite.hauteur ? 0 : 1);
5     }
6   }
```

Trier les éléments d'une collection

Collections

Deux méthodes de tri sont implémentées dans Javaet se trouvent dans la classe Collections (attention avec un **s**).

- La première méthodea un seul argument : une collection d'instance d'une classe qui implémente l'interface Comparable. Le tri se fait donc en utilisant la méthode compareTo codée dans la classe T.
- La seconde méthode nécessite deux arguments : la collection d'instance d'une classe T et une notion d'ordre sur la classe T. Le tri se fera donc en utilisant la méthode compare codée dans la classe implémentant Comparator.

```
// classe Collections
public static <T extends Comparable<? super T»
                void sort(List<T> list)
public static <T> void sort
                (List<T> list, Comparator<? super T> c)
```

Exemple

```
1  public static void main(String[] args){
2
3    List<Personnage> personnages =new ArrayList<>();
4    personnages.add(new IrreductibleGaulois("Obelix", 1.81));
5    personnages.add(new IrreductibleGaulois("Astérix", 1.60));
6    personnages.add(new Personnage("César", 1.75));
7
8    for (Personnage p: personnages)
9      System.out.println(p.presentation());
10
11   Collections.sort(personnages);
12   for (Personnage p: personnages)
13     System.out.println(p.presentation());
14
15   Comparator<Personnage> ordre = new OrdreHauteur();
16   Collections.sort(personnages, ordre);
17   for (Personnage p: personnages)
18     System.out.println(p.presentation());
```