LES COLLECTIONS

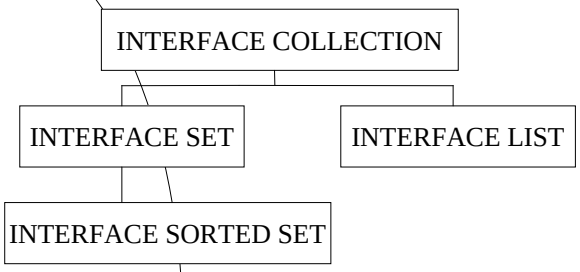
Ludovic Liétard

INTRODUCTION

- Une collection est un objet qui est un rassemblement d'autres objets
- Une collection est utilisée pour stocker, retrouver et manipuler des données
- Les collections sont organisées en interfaces et en classes
- Penser en terme d 'interfaces plutôt qu 'en terme d 'implémentations

LES INTERFACES (1)

- Hiérarchie des interfaces (a)



LES INTERFACES (2)

- Interface Collection : racine de la hiérarchie (pas d 'implémentation directe)
- Interface Set : les éléments ne peuvent être dupliqués (class HashSet)
- Interface SortedSet : les éléments ne peuvent être dupliqués et sont triés (class TreeSet)

LES INTERFACES (3)

- Interface List : les éléments sont ordonnés et peuvent être dupliqués (class Vector)

LES INTERFACES (4)

```
public interface Collection {
   int size();
   boolean isEmpty();
   boolean contains(Object element);
   boolean add(Object element);
   boolean remove(Object element);
   Iterator iterator();
}
```

LES INTERFACES (5)

- Pour connaître la taille d 'une collection :

```
public int size();
```

- Pour connaître la vacuité d ' une collection:

```
public boolean isEmpty();
```

LES INTERFACES (6)

- pour connaître l 'appartenance d 'un objet :

```
public boolean contains(Object o);
```

(`contains(Object o)` se base sur `equals`)

- Pour rajouter un objet :

```
public boolean add(Object o);
```

renvoie `true` si la collection a été modifiée et `false` sinon

LES INTERFACES (7)

- Pour supprimer un objet :

```
public boolean remove(Object o);
```

cette méthode se base sur `equals` et renvoie `true` si la collection a été modifiée et `false` sinon

- Pour parcourir une collection :

```
public Iterator iterator();
```

cette méthode renvoie un `Iterator` sur la collection

LES INTERFACES (8)

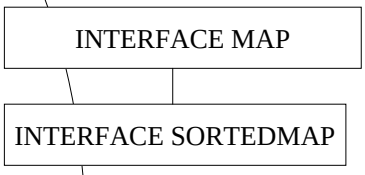
- Interface Iterator

```
public interface Iterator {
        boolean hasNext();
        Object next();
        void remove();
}
```

- La méthode `hasNext()` délivre true si l'on a pas atteint la fin de l 'itération
- La méthode `next()` délivre l'objet courant (en fin d'itération, émet l'exception `NoSuchElementException`)
- La méthode remove() supprime le dernier élément accédé

LES INTERFACES (9)

- Hiérarchie des interfaces (b)



LES INTERFACES (10)

- Interface Map : associe une clé à un objet (à une clé ne peut être associé qu 'un seul objet)
- Interface SortedMap : maintient les clés par ordre trié (ascendant)

LES INTERFACES (11)

```
public interface Map {
   int size();
   boolean isEmpty();
   Object put(Object key, Object value);
   Object get(Object key);
   Object remove(Object key);
   boolean containsKey(Object key);
   boolean containsValue(Object value);
}
```

LES INTERFACES (12)

- Interface Enumeration (class StringTokenizer)

```
public interface Enumeration {
        boolean hasMoreElements();
        Object nextElement();
}
```

- La méthode `hasMoreElements()` délivre true si l'on a pas atteint la fin de l'Enumération
- La méthode `nextElement()` délivre l'objet courant
- Si l'on est en fin de l'énumération, la méthode `nextElement()` émet l'exception

  `NoSuchElementException`

LES INTERFACES (13)

- Exemple :

  Si e est une énumération :

```
while (e.hasMoreElements()) {
    System.out.println(e.nextElement());
}
```

LA CLASSE HASHMAP (1)

- Implémente l 'interface `Map`
- Une instance de la classe `HashMap` est un dictionnaire c'est-à-dire un tableau associatif d'Objet dont la taille est dynamique
- Un dictionnaire se compose donc d'association (clé, valeur) :

  clé étant un identifiant (donc unique)
  valeur étant la valeur associée

LA CLASSE HASHMAP (2)

- Pour rajouter une association dans une instance de `HashMap` :

```
public Object put(Object key,
                  Object value);
```

si la `key` était déjà présente :

l'ancienne association est perdue mais la méthode retourne l 'ancienne valeur associée (retourne `null` sinon)

LA CLASSE HASHMAP (3)

- Pour supprimer une association:

```
public Object remove(Object key);
```

retourne la valeur associée à `key` et `null` si aucune valeur n 'était associée.

LA CLASSE HASHMAP (4)

- Pour consulter une valeur associée à une clé:

```
public Object get(Object key);
```

(retourne `null` si pas d'association avec cette clé)

LA CLASSE HASHMAP (5)

- Pour obtenir les clés :

```
public Set KeySet();
```

- Pour obtenir les valeurs :

```
public Collection values();
```