Les collections

1. Introduction

Une collection, qu'on appelle aussi un conteneur, représente un groupe d'objets et fournit un ensemble de méthodes qui facilite la gestion de ces objets (ajout, suppression, recherche, etc.).

Nous avons déjà abordé la notion de collection quand nous avons utilisé les tableaux. Mais les tableaux ont le défaut d'être de taille fixe : on ne peut pas modifier leur taille. Ce qui pose deux problèmes : s'ils sont trop petits, on risque une erreur faute de place ; s'ils sont trop grands, de l'espace mémoire est gaspillé.

On les remplacera avantageusement par des listes qui peuvent être implémenter de différentes manières.

Les collections étant très utilisées, Java fournit en standard un ensemble de classes permettant de les gérer.

2. Définitions

Les collections sont organisées, de manière assez complexe, avec des interfaces et des classes (comme par exemple *List*, *ArrayList*, *LinkedList*, *Set*, *SortedSet*, *NavigableSet*, *Map*, *SortedMap*, *HashTable*). Seules quelques unes d'entre elles nous seront utiles : *ArrayList*, *HashTable*.

On entend par :

- *set* : un ensemble d'éléments (un même élément ne peut pas être présent plusieurs fois dans l'ensemble);
- *list* : un groupe ordonnées d'éléments (un même élément peut apparaître plusieurs fois);
- *map* : un groupe de couples (clé, valeur) (chaque clé doit être unique, à chaque clé correspond au plus une valeur).

3. Exemples

Les collections utilisent les génériques. Ainsi, il faut préciser entre des chevrons le type des éléments que l'on veut utiliser dans la collection.

Exemples :
Liste de chaîne :

```
ArrayList<String> l1 = new ArrayList<String>();
```

Liste de personnes :

```
ArrayList<Personne> l2 = new ArrayList<Personne>();
```

Dictionnaire (ou tableau associatif) :

```
HashTable<String, Date> anniversaires = new HashTable<String, Date>(); // à chaque prénom, on associera une date de naissance
```

4. Parcours d'une collection

Voir l'exemple.

Il existe deux manières pour parcourir une collection :

- utilisation de la nouvelle boucle *for* (*for each*) ;
- utilisation d'un itérateur.

Avec un itérateur on peut parcourir un collection mais aussi retirer un élément de la collection, ce qui n'est pas possible avec une boucle for.

5. Trier une liste

Pour trier une liste d'élément il faut que le type des éléments implémente l'interface *Comparable* ce qui revient à donner un corps à la méthode *compareTo*.
À ce moment-là, on peut utiliser la méthode *sort* de la classe *Collections* (avec un s).

Voir l'exemple.

6. Application

On souhaite disposer d'une application en mode CLI qui permet de gérer un répertoire téléphonique. On entend par gérer : mettre à jour, ajouter, supprimer, consulter (MASC[1]) des personnes avec leurs numéros de téléphones.

Dans une première version, on associe à chaque personne (identifié par une chaîne de caractères) un unique numéro de téléphone.

Dans une deuxième version, on doit pouvoir associer à chaque personne un voire plusieurs numéros de téléphone.

1 Les Anglo-saxons disent CRUD (Create, Read, Update, Delete) pour parler des quatre opérations de base que l'on peut réaliser sur des données.