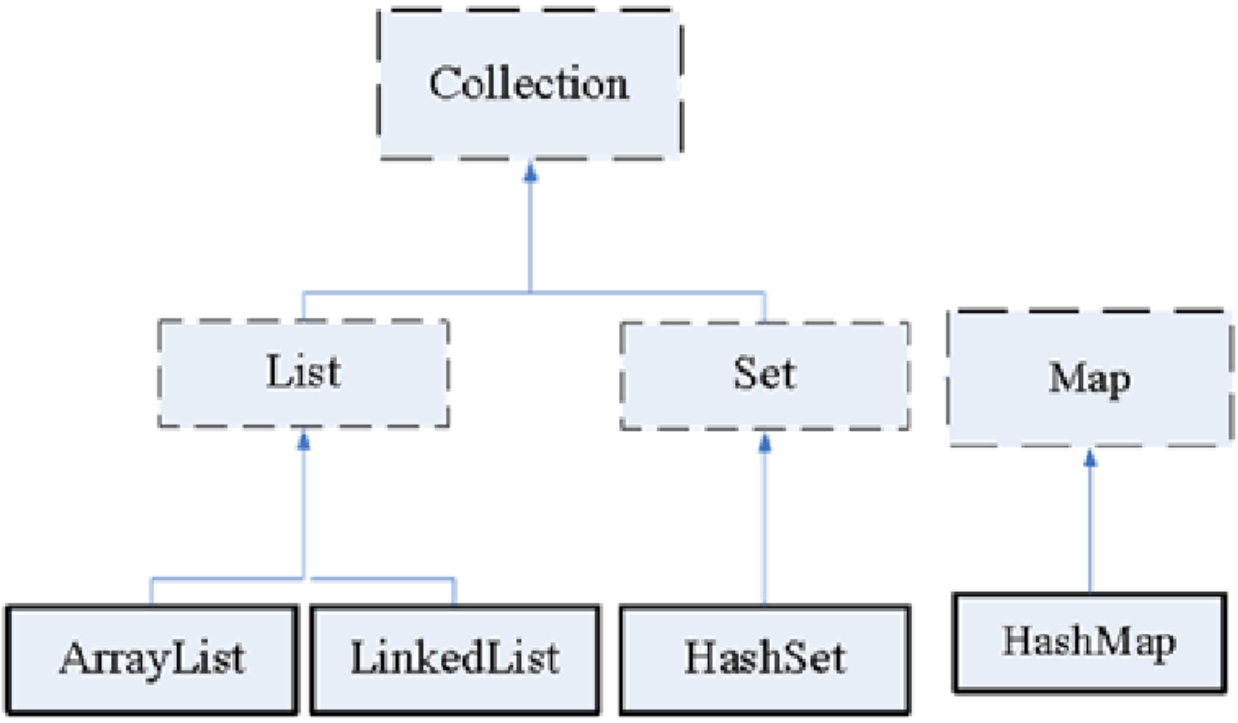
TP10 : Révisions (*Collection)*



Les collections sont des objets qui permettent de gérer des ensembles d'objets. Ces ensembles de données peuvent être définis avec plusieurs caractéristiques : la possibilité de gérer des doublons, de gérer un ordre de tri, etc. ...

Une collection est un regroupement d'objets qui sont désignés sous le nom d'éléments.

L'API Collections propose un ensemble d'interfaces et de classes dont le but est de stocker de multiples objets.

Exercice 1 Modéliser

On rappelle quelques interfaces de collections importantes :

- List<E> : liste d'éléments avec un ordre donné, accessibles par leur indice.
- Set<E> : ensemble d'éléments sans doublons
- Map<K,E> : ensemble d'associations (clé dans K, valeur dans E), tel qu'il n'existe qu'une seule association faisant intervenir une même clé.

Ces interfaces peuvent évidemment être composées les unes avec les autres. Exemple : un ensemble de séquences d'entiers se note Set<List<Integer>>.

Pour les situations suivantes, déclarez le type de la structure de données qui peut les modéliser :

1. Donnée des membres de l'équipe de France de Football.
2. Donnée des membres de l'équipe de France de Football avec leurs rôles respectifs dans l'équipe.
3. Marqueurs de buts lors du dernier match (en se rappelant la séquence).
4. Affectation des étudiants à un groupe de TD.
5. Pour chaque groupe de TD, la "liste" des enseignants.
6. Pour chaque groupe de TD, affectation de l'enseignant pour chaque UE.
7. Étudiants présents lors de chaque séance de TD de POOIG du semestre.

Pour faire le parcours d'une HashMap en Java, il y a deux méthodes : **for** et **iterator** à l'intérieur de la boucle while. Voici un exemple ci-dessous :

```
import java.util.HashMap;
import java.util.Iterator;
import java.util.Map;

public class parcoursHashMap {

    public static void main(String[] args) {

        HashMap<String,Double> map = new HashMap<String,Double>();

        map.put("A",12.0);
        map.put("B",42.1);
        map.put("C",5.6);
        map.put("D",29.7);

        //Boucle for
        System.out.println("Boucle for:");
        for (Map.Entry mapentry : map.entrySet()) {
           System.out.println("clé: "+mapentry.getKey()
                              + " | valeur: " + mapentry.getValue());
        }

        //Boucle while+iterator
        System.out.println("Boucle while");
        Iterator iterator = map.entrySet().iterator();
        while (iterator.hasNext()) {
          Map.Entry mapentry = (Map.Entry) iterator.next();
          System.out.println("clé: "+mapentry.getKey()
                            + " | valeur: " + mapentry.getValue());
        }
    }
}
```

Exercice 2 Bibliothèque

Nous voulons programmer un logiciel de gestion de bibliothèque. Voici la situation dans ses grandes lignes :

- Une bibliothèque possède un certain nombre d'ouvrages, certains en plusieurs exemplaires.
- Une bibliothèque a une certaine "liste" de personnes abonnées.
- Chaque abonné peut emprunter des ouvrages (avec limite du nombre d'ouvrages et de la durée)
- La bibliothèque garde l'historique de tous les emprunts sur un an.

Par ailleurs, pour une gestion efficace, les bibliothécaires souhaitent disposer des fonctionnalités suivantes :

- Mettre à jour l'historique en effaçant les entrées datant de plus d'un an.
- Obtenir les ouvrages disponibles (au moins 1 exemplaire non prété).
- Obtenir la "liste" des abonnés ayant du retard dans leurs emprunts.
- Suggérer des "amis" à un abonné (personnes ayant emprunté un livre en commun avec cet abonné). Stocker (mettre en cache) le résultat de la recherche (pour donner un résultat plus rapidement la prochaine fois).

Modélisez le système de gestion de bibliothèque en UML.

Programmez les fonctionnalités suggérées.

Exercice 3 Pharmacie

On souhaite modéliser le dossier de patients qui sont les clients d'une pharmacie. Chaque patient est représenté par son nom et sa liste de médicaments (chacun de type String) de son ordonnance, **sans doublons**. Les opérations sur le dossier des patients devront garantir que **les noms de patients sont différents, et qu'un médicament** ne figure qu'une fois dans une ordonnance.

- Créer les classes nécessaires pour représenter une pharmacie
- Définir une méthode qui prend en paramètre le nom d'un médicament et retourne une collection contentant tous les patients de la pharmacie ayant pris ce médicament.
- Définir une méthode permettant de supprimer du dossier des patients tous les patients dont l'ordonnance est vide (aucun médicamant).