TD Collections

Exercice 1 : Etudiants, Matières, Groupes et Formations

Une formation (*training*) est définie par un identifiant et la liste des matières qui y sont enseignées avec leurs coefficients. Les coefficients d'une matière peuvent changer d'une formation à l'autre.
Un étudiant est caractérisé par son identité (cf. classe `formation.Identity` ci-dessous), sa formation et ses résultats. Les résultats d'un étudiant sont mémorisés sous la forme d'une liste de notes par matière .
Une matière (*subject*) est définie par son nom. On supposera défini un type énuméré `formation.Subject`.
Un groupe d'étudiants est défini par la liste des étudiants qui le compose et la formation à laquelle ce groupe appartient.
La classe `formation.Identity` est définie ainsi :

formation::Identity
- nip : String - surname : String - firstname : String
+Identity(nip : String, surname : String, firstname :String) +getNip():String +getSurname():String +getFirstname():String +toString():String +equals(o:Object) : boolean +hashCode() : int

On support que le `nip` est unique. Donc deux identités qui ont le même `nip` sont considérées égales.

Q 1 . Définissez la classe `Training`. On veut pouvoir ajouter ou supprimer une matière dans une formation, connaître le coefficient d'une matière, obtenir toutes les matières de la formation.

Quelle structure de données proposez-vous pour gérer les matières de la formation et leur coefficient ?

Q 2 . Définissez la classe `Student`. Il faut pouvoir ajouter une note à un étudiant, calculer sa moyenne pour une matière, sa moyenne générale, modifier sa formation. Lorsque l'on modifie la formation d'un étudiant ses notes sont supprimées.

Quelle structure de données proposez-vous pour gérer les notes d'un étudiant ?

Q 3 . Définissez la classe `StudentGroup`. On doit pouvoir ajouter, supprimer un étudiant du groupe, calculer la moyenne du groupe pour une matière, la moyenne générale.

Q 4 . Pour la classe `StudentGroup` : on veut en plus pouvoir trier les étudiants du groupe selon différents critères. Et donc avoir des méthodes `orerByMerit`, `orderByAlpha`, `orderBySubjectMerit` qui retournent la liste des étudiants du groupe triée selon leur moyenne générale décroissante, leur ordre alphabétique croissant, leur moyenne dans une matière croissante.

On utilisera la méthode de `java.util.Collections` :

```
public static void sort(List<T> list, Comparator<? super T> comp)
```

qui modifie son premier élément et s'appuie sur l'interface : `Comparator<T>` qui impose la méthode :

```
public int compare(T o1, T o2)
```

Q 5 . Proposez un code pour les méthodes `equals` et `hashCode` de `Identity`.