UE Programmation Orientée Objet

TD Collections

Exercice 1 : Etudiants, Matières, Groupes et Formations

Un étudiant est caractérisé par son identité, sa formation et ses résultats. L'identité est définie par un NIP (unique), un nom et un prénom (3 chaînes de caractères). Les résultats d'un étudiant sont mémorisés sous la forme d'une liste de notes par matière .

Une formation est définie par un identifiant et la liste des matières qui y sont enseignées avec leurs coefficients.

Une matière est définie par son nom.

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.Identite` est définie ainsi :

formation::Identite
- nip : String - nom : String - prenom : String
+Identite(nip : String, nom : String, prenom :String) +getNip():String +getNom():String +getPrenom():String +toString():String +equals(o:Objet) : boolean +hashCode() : int

On décide que les matières qui existe pour l'ensemble des formations sont définies dans un type énuméré `Matiere`.

Q 1 . Définissez la classe `Formation`. On veut pouvoir ajouter ou supprimer une matière dans une formation. Connaître le coefficient d'une matière.

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

Q 2 . Définissez la classe `Etudiant`. 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 `Groupe`. 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 `Groupe` : on veut en plus pouvoir trier les étudiants du groupe selon différents critères. Et donc avoir des méthodes `triParMerite`, `triAlpha`, `triParMatière` 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 `Identite`.