IUT Lannion **R2.01**
Département informatique

TP5 : Les collections

Consultez la documentation en ligne pour obtenir plus d'information sur l'utilisation des classes *HashMap* et *HashSet*.
Pour tout ajout, modification ou suppression d'un élément dans une collection, vérifiez resp. que l'élément n'existe pas, qu'il existe avant de réaliser l'opération correspondante.

Vous parcourrez les HashSet et les HashMap de trois manières différentes. Penser aux Iterator au moins une fois.

I. Exercice 1

Question 1/ Proposez une classe pour représenter un cours (une seule variable d'instance, une instance de `String` pour son intitulé). Prévoir un constructeur, l'encapsulation des variables d'instance et la redéfinition des méthodes `equals`() et `toString()`.

Question 2/ On souhaite gérer un ensemble d'étudiants. Chaque étudiant sera défini par une classe *Etudiant* caractérisée par les informations suivantes :

- Un matricule (numéro d'inscription) qui est entier, identifiant unique de chaque étudiant ;
- Un nom ;
- Un prénom ;
- Une instance de *HashMap* conserve pour chaque cours, la note obtenue par l'étudiant : une seule note est associée à matière.

Les attributs de la classe *Etudiant* sont encapsulés. Prévoir les méthodes suivantes :

- Un constructeur ;
- La redéfinition des méthodes `equals`() et `toString()`.
- Deux méthodes permettent d'ajouter et de modifier une note pour un cours (le cours et la note sont passés en paramètre) ;
- Une méthode affiche les notes des candidats pour toutes les matières ;
- Une méthode supprime une note étant donné le libellé du cours ;
- Une méthode calcule la moyenne générale d'un étudiant ;
- Une méthode délivre la note d'une matière (l'identifiant de la matière est passé en paramètre) ;
- Une méthode retourne sa meilleure note ;
- Une méthode retourne sa moins bonne note.

Question 3/ La classe *Groupe* permet de gérer les étudiants d'un groupe. Les attributs de la classe *Groupe* sont :

- Nom du groupe ;
- Une *Collection* d'étudiants (un *HashSet*);

Ces attributs sont également encapsulés. Prévoir les méthodes suivantes : elles utiliseront l'interface *Collection* :

- Un constructeur ;
- Une méthode pour ajouter un étudiant (l'étudiant étant passé en paramètre) ;
- Une méthode pour supprimer un étudiant (le matricule est passé en paramètre) ;
- Une méthode affiche tous les étudiants du groupe ;
- Une méthode pour connaitre le nombre d'étudiants d'un groupe ;
- Une méthode pour calculer la moyenne du groupe ;
- Une méthode pour afficher les noms du premier et dernier du groupe.

Question 4/ Prévoir un jeu d'essai complet pour tester toutes les méthodes précédemment écrites.

Question 5/ Si vous avez le temps, créez un menu complet dans le programme principal pour la gestion des étudiants d'un groupe. Affichez les statistiques des étudiants et du groupe.

II. Exercice 2

Question 1/ Proposer une classe pour représenter une compétition équestre (une seule variable d'instance, une instance de `String` pour son intitulé). Prévoir un constructeur, l'encapsulation de la variable d'instance et la redéfinition des méthodes `equals`() et `toString()`.

Question 2/ On souhaite établir le classement d'un ensemble de cavaliers pour toutes les compétitions. Chaque cavalier sera défini par une classe *Cavalier* caractérisée par les informations suivantes :

- Un numéro de licence, entier, identifiant unique pour chaque cavalier ;
- Un nom ;
- Un prénom ;
- Une instance de *HashMap* conserve pour chaque compétition, la place obtenue par le cavalier à cette compétition.

Les attributs de la classe Cavalier sont encapsulés. Prévoir les méthodes suivantes :

Un constructeur ;

- La redéfinition des méthodes `equals`() et `toString()`.
- Deux méthodes permettent d'ajouter et de modifier une place à une compétition (la compétition étant passé en paramètre) ;
- Une méthode supprime une place étant donné le nom de la compétition ;
- Une methode affiche tous les classements du cavalier avec sa compétition et sa place :

- Une méthode délivre la place d'une compétition (la compétition est passé en paramètre) ;
- Une méthode retourne sa meilleure place et la compétition concernée ;
- Une méthode retourne sa moins bonne place.

Question 3/ La classe C*lub* permet de gérer les cavaliers d'un club. Les attributs de la classe *Club* sont :

- Nom du club ;
- Une *Collection* de cavaliers (un *HashSet*);

Ces attributs sont également encapsulés. Prévoir les méthodes suivantes : elles utiliseront l'interface *Collection* :

- Un constructeur ;
- Une méthode pour ajouter un cavalier (le cavalier étant passé en paramètre) ;
- Une méthode pour supprimer un cavalier (le cavalier est passé en paramètre) ;
- Une méthode pour afficher tous les cavaliers du club ;

- Une méthode pour afficher le classement d'un club à une compétition donnée : le nom de la compétition sera passée en paramètre. Il faudra afficher le nom et le club de chaque cavalier placé.

Question 4/ Prévoir un jeu d'essai complet pour tester toutes les méthodes précédemment écrites.

.