Exercice 1. Membres universitaires **Objectif(s) :** *Héritage, abstraction*

On souhaite réaliser un programme permettant de gérer les membres universitaires, tant le personnel que les étudiants. On considère qu'un membre du personnel peut être soit un administratif, soit un personnel enseignant, et qu'un personnel enseignant peut être soit un enseignant, soit un enseignant-chercheur. On considère également les règles de gestion de base suivantes :

1. toute personne membre de l'université est identifiée par un numéro Harpège, et possède prénom, nom, adresse postale de domicile, adresse mail institutionnelle ;
2. un membre du personnel possède également, éventuellement, plusieurs autres adresses mail ;
3. un membre du personnel est rattaché à un établissement de référence ;
4. un enseignant-chercheur peut également être rattaché à un établissement de recherche, différent de l'établissement de référence ;
5. les enseignants et enseignant-chercheurs ont un service annuel d'enseignement à effectuer, chiffré en nombre d'heures ; ce service est calculé de la façon suivante :
   service = service statutaire de base - décharge
   où le service statutaire de base est de 192 h pour un enseignant-chercheur, et 384 h pour un enseignant ; une décharge peut prendre n'importe quelle valeur, à concurrence du service de base ;
6. le nombre d'heures effectuées par un enseignant ou un enseignant-chercheur au cours d'un semestre est rapporté et archivé sous la forme d'un ensemble de couples (UV, nombre d'heures) ;
7. le service effectué chaque année universitaire par un enseignant ou un enseignant-chercheur est calculé à partir du nombre d'heures rapporté ; on doit pouvoir comparer le service effectif avec le service attendu ;
8. un étudiant est inscrit dans un cursus d'étude (DUT, L, M, D), pour une spécialité (maths, info, lettres moderne, etc.) et un semestre particuliers (tous les cursus n'ayant pas nécessairement le même nombre de semestres) ;
9. les évaluations d'un étudiants sont enregistrées pour chaque semestre ; une évaluation est vue comme un ensemble de couples (note, coef.) ;
10. un étudiant est admis dans le semestre suivant si sa moyenne sur le semestre est supérieure à 10/20 ;
11. un étudiant est admis au diplôme concerné par son cursus s'il a obtenu la moyenne à l'ensemble des semestres du cursus.

Question 1.1. Créer la structure de classes adéquate pour traiter le problème.

Question 1.2. Munir chaque classe des constructeurs, accesseurs et mutateurs utiles. Attention à l'accessibilité de chacuns.

Question 1.3. Redéfinir les méthodes `toString()` et `equals()` pour chaque classe. Tester ces méthodes dans le programme principal, notamment en affichant les informations connues pour chaque personne.

Question 1.4. Écrire une méthode de calcul d'un nouvel identifiant Harpège. Utiliser cette méthode lors de la construction d'un nouveau membre universitaire.

Question 1.5. Écrire les méthodes nécessaires à l'implémentation des règles de gestion ci-dessus : caclul de moyenne d'un étudiant, cacul d'admissibilité, calcul de service effectif d'un enseignant, différence entre service attendu et service effectif, etc. Cela inclus, bien entendu, la programmation de toute méthode-outil qui s'avererait utile.

Question 1.6. Créer une classe `Promotion` représentant un ensemble d'étudiants. Dans le programme principal, créer un objet de type `Promotion` et l'initialiser avec plusieurs étudiants. Vérifier les comportements du calcul de moyenne et du calcul d'admission pour chacun des étudiants de la promotion.

Question 1.7. Dans la classe `Universite`, créer un champ destiné à représenter l'ensemble du personnel enseignant de l'université, et munir la classe des méthodes utiles habituelles associées à ce champ. Dans le programme principal, créer des instances de personnel enseignant (enseignants et enseignant-chercheurs) pour une université donnée. Tester les différents utilitaires de calcul de service pour chaque personnel enseignant de cette université.

Exercice 2. Médiathèque **Objectif(s) :** *Encapsulation, héritage simple*

TODO 1 — *Renommer* `Copy.borrow()` *en* `Copy.setAsBorrowed()` *pour éviter la confusion avec le* `LoanRecord.borrow()` *à venir. Mettre également à jour code source*

On souhaite réaliser un programme permettant de gérer les documents d'une médiathèque et leurs emprunts. On considère les règles de gestion suivantes :

- les documents gérés sont de deux types : livre ou CD ;
- un document est identifié par un numéro ;
- un document a un titre ;
- un CD est un document qui a un interprète et un compositeur ;
- un livre est un document qui a un auteur ;
- certains documents sont *sortables* (i.e. on peut les sortir de l'enceinte de la médiathèque), d'autres non (i.e. ils ne peuvent être que consultés sur place).
- la médiathèque peut posséder dans ses rayons plusieurs exemplaires d'un même document ;
- un exemplaire est identifié par un numéro ;
- un exemplaire peut être emprunté, mais uniquement si le document correspondant est sortable et non déjà emprunté.

Question 2.8. Proposer un diagramme de classes qui modélise la situation exposée ci-dessus.

Question 2.9. Implémenter les classes du modèle, en pensant à respecter le principe d'*encapsulation* (cf. cours). Redéfinir la méthode `toString()` lorsque nécessaire. On rappelle que cette méthode retourne un objet de type `String` qui décrit l'*état* de l'objet au moment de l'appel.

Question 2.10. Construire une classe `Mediatheque` qui permet de gérer un ensemble d'exemplaires de documents. Cette classe doit être exécutable (i.e. munie d'une méthode `main`) et munie dans un premier temps des fonctionnalités publiques suivantes :

- ajout d'un exemplaire de document quelconque
- ajout de plusieurs exemplaires

Tester.

Question 2.11. Ajouter les méthodes suivantes (attention au choix de la classe de définition) :

- `public boolean estEmpruntable()`
- `public boolean emprunter()` qui retourne `true` ou `false` selon que l'emprunt a pu être effectué ou pas
- `public boolean retourner()` qui retourne `true` ou `false` selon que le retour a pu être effectué ou pas.

Tester.

Question 2.12. (subsidiaire) Ajouter à la classe `Mediatheque` les fonctionnalités publiques suivantes :

- recherche de document(s) par titre
- recherche de tous les livres d'un auteur donné
- recherche de tous les CD d'un interprète ou compositeur donné