Introduction

L'école Infomaniak s'agrandit d'année en année, avec toujours plus de campus s'ouvrant de part le monde. Un des inconvénients majeurs de cet agrandissement est l'apparition de nouveaux campus, qu'il est parfois difficile de gérer en termes d'effectifs et de professeurs.

Afin de palier cet inconvénient, la direction technique de l'école a décidé de développer une application qui permettra de faciliter la gestion du nombre d'étudiants au sein de chaque campus, ainsi que la gestion des professeurs. Cette application devra contenir la liste des campus existants à Infomaniak, ainsi que la liste des étudiants inscrits et des professeurs donnant cours dans chaque campus.

Caractéristiques de l'application

Cette application devra répondre aux contraintes suivantes :
- Gestion de la liste des campus de l'école Infomaniak
- Gestion de la liste des étudiants inscrits dans chaque campus
- Limitation du nombre d'étudiants par campus

Fonctionnalités attendues

La direction technique de l'école vous demande de développer la base de l'application représentée par ses entités.

Modélisation d'un campus

Un campus sera représenté par la classe Campus, qui contiendra une ville, une région et une capacité :

- La ville sera une chaîne de caractères, contenant le nom de la ville où se situe le campus
- La région sera une chaîne de caractères, contenant le nom de la région où se situe le campus
- La capacité sera un entier, contenant le nombre maximum d'étudiants pouvant s'inscrire dans le campus. Une capacité de zéro correspondra à un campus sans limite de places.

Modélisation d'un étudiant

Un étudiant sera représenté par la classe Student, qui contiendra un ID, un prénom, et un nom :

- L'ID sera un entier, contenant l'ID d'un étudiant.
- Un ID de 0 correspondra à un étudiant n'ayant pas encore d'ID
- Le prénom et le nom seront des chaînes de caractères.

Modélisation d'un professeur

Un professeur sera représenté par une classe abstraite, Teacher, qui aura deux classes filles : InternalTeacher et ExternalTeacher.

La classe Teacher contiendra un ID, le nom et le prénom du professeur :

- L'ID sera un entier
- Le prénom et le nom seront des chaînes de caractères

La classe ExternalTeacher contiendra les propriétés de la classe Teacher, puis une propriété supplémentaire :

- Le salaire du professeur externe, qui sera un entier

La classe InternalTeacher contiendra les propriétés de la classe Teacher. Cependant, le salaire étant le même pour tous les InternalTeachers, il vous est demandé de trouver une façon de pouvoir le renseigner pour tous, tout en pouvant le modifier facilement.

Liste des étudiants par campus

Chaque campus devra également contenir la liste des étudiants inscrits au sein de ce campus pour l'année scolaire en cours.

Pour cela, trois méthodes seront à créer au sein de la classe Campus :

- addStudent(Student s) : Cette méthode ajoute un étudiant dans le campus. Si le campus est plein, cette méthode devra lever une exception.
- removeStudent(Student s) : Cette méthode supprime un étudiant du campus.
- getStudents() : Cette méthode renvoie la liste des étudiants du campus. La liste renvoyée par cette méthode ne doit pas être modifiable

Liste des professeurs par campus

En plus de la liste des étudiants, chaque campus contiendra la liste des professeurs donnant cours dans ce campus.

Pour cela, trois nouvelles méthodes seront à ajouter :

- addTeacher(Teacher t) : Cette méthode ajoute un professeur dans le campus
- removeTeacher(Teacher t) : Cette méthode supprime un professeur du campus
- getTeachers() : Cette méthode renvoie la liste des professeurs du campus. La liste renvoyée par cette méthode ne doit pas être modifiable.

Fonctionnalités avancées

Tri de la liste des étudiants

La liste des étudiants au sein de chaque campus doit être triée. Les étudiants seront triés par IDcroissants. Les étudiants n'ayant pas d'ID seront affichés en premiers.

Egalité entre campus et entre étudiants

L'application sera amenée à comparer des étudiants et des campus entre eux, afin de savoir si ces objets sont égaux ou différents.

Les critères de comparaisons seront les suivants :

Concernant les campus :
- Deux campus étant dans la même ville et la même région sont égaux (et ce, même si leurs capacités sont différentes)
- Deux campus étant dans deux villes différentes ou deux régions

Concernant les étudiants :
- Si les deux étudiants ont un ID, et que leurs ID sont identiques, alors ils sont égaux (et ce, même si leurs noms et prénoms sont différents)
- Si deux étudiants n'ont pas d'ID, ils seront égaux si et seulement si leurs noms et prénoms sont égaux.
- Si un étudiant a un ID et un autre n'en a pas, alors ils sont différents

Sauvegarde des données

Les objets que vous créez durant le développement de cette application seront écrits dans un fichier afin de les sauvegarder lors de l'arrêt de l'application. Vous devrez donc vous assurer que vos objets pourront être écrits dans un fichier. Vous pouvez utiliser la sérialisation sur les classes Campus, Student et Teacher, mais un export XML serait plus ouvert.

Gestion des exceptions

La méthode addStudent de la classe campus doit lever une exception. Afin d'en faciliter la gestion, vous devrez lever une exception de type personnalisé. La méthode addStudent devra lever une exception de type FullCampusException si le campus est déjà complet. De plus, cette exception devra être non vérifiée, c'est-à-dire étendre de RuntimeException.

Jeu de test

Pour la présentation de votre projet à l'ensemble de la direction d'Infomaniak, il vous est demandé de fournir un simple jeu de test en ligne de commande (predéfini, ou non) permettant de tester toutes les méthodes et création d'entités.