Cas d'étude Gestion des TER

On s'intéresse à la mise en place d'une application destinée à aider le responsable des TER de M1 (dans un master pas si lointain) dans sa tâche. Le TER de M1 est réalisé par groupes, et est validé par une soutenance et un rapport évalué par un rapporteur différent de l'encadrant. Au début du semestre, les sujets de TER sont collectés par le responsable : chaque sujet a un titre et un résumé et est donc proposé par un enseignant qui en est l'encadrant. En parallèle, les étudiants réfléchissent à se répartir en groupes d'au maximum 5 étudiants. Il y a en général beaucoup plus de sujets que de groupes.
A partir des sujets collectés (chacun se voyant affecter un code unique), le responsable de TER doit pouvoir générer une page web listant les sujets et leurs caractéristiques.
Les étudiants constituent des groupes, et chaque groupe émet au maximum 5 vœux et au minimum un vœu concernant les sujets qu'ils souhaitent se voir affecter : en voeu 1 le sujet préféré, en voeu 5 un sujet moins préféré ... A partir des vœux des étudiants, le responsable des TER va lancer un processus d'affectation sujets/groupes en 2 passes. A l'issue de cette affectation, tous les groupes ont un sujet. Les sujets qui ont été attribués ont un groupe, mais certains sujets ne sont pas affectés. Les étudiants non intégrés à un groupe ne sont pas concernés par l'affectation des sujets. L'affectation des sujets est enregistrée et une nouvelle page web peut être générée indiquant les affectations des sujets (les sujets non affectés sont indiqués). Lors de la première passe, seules les préférences des groupes servent à leur affecter un sujet. A l'issue de cette première passe, certains sujets sont affectés à certains groupes. Les groupes n'ayant pas pu être affectés doivent refournir une liste de préférences parmi les sujets non affectés restants. L'affectation finale lors de cette deuxième passe attribue quoiqu'il arrive un sujet à un groupe même en dehors de ses préférences.
Vers la fin du semestre, le responsable des TERs peut lancer le processus d'affectation des rapporteurs : les rapporteurs sont choisis (aléatoirement) parmi les enseignants ayant un sujet affecté. Si un enseignant a n sujets affectés, alors il rapportera n autres groupes. Un enseignant ne peut pas rapporter son groupe. Tous les groupes affectés doivent avoir un rapporteur. L'affectation des rapporteurs est enregistrée et une nouvelle page web peut être générée indiquant les affectations des rapporteurs (les sujets non affectés n'ont bien sûr pas de rapporteurs).
La planification des soutenances est générée. Chaque groupe réalise une soutenance d'une heure. Les soutenances ont lieu en séquence, sans pause, sur des créneaux de 9h à midi puis de 13h à 17h, sur une période de temps prédéfinie par le responsable des TERs. Le planning est sauvegardé (horaire de chaque groupe), et également généré au format iCal, de manière à ce que chaque soutenance contienne les informations sur le groupe, le rapporteur, l'encadrant. L'application doit également permettre d'importer du iCal structuré de la même manière. En effet, le responsable des TERs, pour opérer des modifications dans le planning, importera le iCal généré dans un outil pratique extérieur à l'application des TER, puis réimportera le fichier généré par cet outil tierce.
Dans une première version du logiciel, on souhaite :

- que l'application soit développée en Java
- que l'application ne nécessite pas de bases de données, et que la persistance des données soit assurée par des fichiers de format JSON, en utilisant Jackson.
- une interface texte (menu textuel ou différentes commandes accessibles en ligne de commande lors du lancement de l'application)
- ne pas automatiser la collecte des sujets, et des vœux des groupes : le responsable continue de collecter dans un/des fichier(s) de format à définir
- de même la collecte des documents liés au TER est réalisée par Moodle
- l'affectation des sujets aux groupes est réalisée en prenant en compte les vœux des étudiants, sans contrainte du point de vue enseignant. L'affectation sera telle que la somme des préférences des groupes dans les couplages groupe/sujet choisis soit minimale.
- lors de la génération de l'emploi du temps, on prévoit une version sans contrainte de disponibilité, les contraintes sont gérées hors de l'application et à la main par le responsable de TER.

Question 1. Déterminez les fonctionnalités attendues pour cette première version et réalisez un diagramme de cas d'utilisation.

Question 2. Proposez un diagramme d'activité modélisant le déroulement de la gestion du planning de soutenances.

Question 3. Proposez un modèle de classe d'analyse pour l'ensemble de l'application.

Question 4. Proposez une machine à états décrivant les groupes, en vous concentrant sur le processus d'affectation des sujets aux groupes, qui est celui qui nous intéresse dans la suite.

Question 5. Nous allons implémenter le cas d'utilisation relatif à l'affectation sujets/groupes. L'algorithmique combinatoire nécessaire à une affectation équitable sera réalisée par une autre équipe, qui vous fournit l'interface décrite plus loin.

a- Détaillez la modélisation de cette partie dans le diagramme de classes.

b- Implémentez les classes correspondantes (en incluant les mécanismes de sérialisation/désérialisation). Prévoyez une sérialisation dans un fichier pour les groupes, et un autre pour les sujets

Code 1 – Interface pour l'algo d'affectation

```
import java.util.List;
/**
 * Interface de l'algorithme hongrois utilisé pour réaliser le couplage maximal de
↪  poids min
 */

public interface Hongrois {
        /**
         * Permet de positionner la taille (hauteur) de la matrice
         * @param hauteur la hauteur de la matrice avec hauteur>0
         */
        public void setHauteur(int hauteur);

        /**
         * Permet de positionner la taille (largeur) de la matrice
         * @param largeur la largeur de la matrice avec largeur>0
         */
        public void setLargeur(int largeur);
        /**
         * Permet de positionner les valeurs de la matrice
         * @param adjListe la liste d'adjacence définie par (i, j, v)
         * (i, j, v) : la case (i, j) prend la valeur v
         * avec 1<=i<=hauteur ; 1<=j<=largeur ; v>=0
         */
        public void setAdjacenceList(List<List<Integer>> adjListe);
        /**
         * Calcule et retourne les affectations
         * @param phaseNumber : 1 pour une affectation partielle, 2 pour une
↪  affectation globale
         * @return une liste de couples (i, j) qui correspondent aux affectations
↪  calculées
         */
        public List<List<Integer>> affectation(int phaseNumber);
}
```