Rapport TP1 IFT1025

Abdel Rahman Ibrahim 20260406

I. Introduction

Ce rapport présente le travail réalisé pour développer un programme de gestion d'horaire d'étudiant. Le programme permet la création, la modification et la suppression de cours et d'examens, ainsi que la visualisation de l'emploi du temps généré. Il est conçu selon une approche orientée objet et comprend plusieurs classes interagissant pour fournir les fonctionnalités requises.

Nous mettrons en évidence les classes utilisées, ensuite nous décrirons leurs membres et nous expliquerons leur rôle dans le programme. Enfin, un bilan qualitatif du travail sera présenté, incluant les difficultés rencontrées et les suggestions d'amélioration.

II. Les classes du programme

1. La classe Main

Cette classe continent méthode principale main qui lance le programme. Elle gère le menu principal du programme et appelle les méthodes appropriées en fonction du choix de l'utilisateur, elle contient aussi la méthode runTestsUnitaires, qui est la méthode utilisée pour exécuter les tests unitaires du programme.

2. La classe Cours

La classe Cours présente un cours individuel, elle contient les membres suivants :

- Attributs privés : nom, numero, credits, horairesCours, horairesTP, et horairesExamen.
- Constructeurs :
    - Le premier constructeur : (Cours(String nom, String numero, int credits, List<Horaire> horairesCours, List<Horaire> horairesTP, List<HoraireExamen> horairesExamen) avec tous les attributs est utilisé pour créer un objet Cours en spécifiant toutes les informations nécessaires,
    - Le deuxième constructeur avec les attributs essentiels est utilisé lorsque seules les informations de base du cours, telles que le nom, le numéro et les crédits, sont disponibles.
- Les getters et setters : Ces méthodes permettent d'accéder et de modifier les attributs de la classe Cours. Comme par exemple getNom(), setNom(String Nom)…

Ces membres de classe permettent de stocker et de manipuler les informations relatives à un cours, telles que son nom, son numéro, ses crédits, ses horaires de cours, ses horaires de TP et ses horaires d'examen.

3. La classe Horaire

La classe Horaire représente un horaire pour un cours magistral ou un TP. Elle contient les attributs jour de la semaine, heure de début et heure de fin.

- Attributs privés : jourSemaine, heureDebut, heureFin
- Constructeur :
    - Horaire(String jourSemaine, LocalTime heureDebut, LocalTime heureFin)
    - Ce constructeur constitue un horaire de TP ou de cours magistral
- Getters et setters : Permettent de modifier et d'accéder aux attributs en respectant le principe d'encapsulation

4. La classe HoraireExamen

Rapport TP1 IFT1025

Abdel Rahman Ibrahim 20260406

La classe HoraireExamen représente un horaire pour un examen, qui est composé d'une date, une heure de début/fin. Les dates et les temps sont modélisés par LocalTime et LocalDate

- Attributs : date, heureDebut, heureFin
- Constructeur :
    - HoraireExamen(LocalDate date, LocalTime heureDebut, LocalTime heureFin)
    - Crée l'objet HoraireExamen
- Les getters et setters permettent de manipuler les attributs en respectant le concept d'encapsulation.

Cette classe est non seulement utilisée pour représenter les horaires des examens , mais aussi pour stocker et manipuler les infos liées à celles-ci.

5. La classe Affichage

La classe Affichage contient des méthodes pour afficher les horaires des cours, des TP et des examens. Elle fournit également des méthodes pour créer de nouveaux horaires à partir de la saisie de l'utilisateur. Elle compote les méthodes suivantes :

- afficherHoraire(List<Horaire> horaires, String type) : Afiche les horaires des cours ou des TP en format texte.
- afficherHoraireExamen(List<HoraireExamen> horairesExamen) : Affiche les horaires des examens en format texte.
- creerHoraire(Scanner scanner) : Permet à l'utilisateur de saisir les horaires des cours ou des TP et renvoie une liste d'objets Horaire.
- creerHoraireExamen(Scanner scanner) : Permet à l'utilisateur de saisir les horaires des examens et renvoie une liste d'objets HoraireExamen.
- jourValide(String jourSemaine) : Valide si un jour de la semaine est valide (lundi, mardi, mercredi, jeudi, vendredi).
  convertirHeure(String heureStr) : Convertit une chaîne de caractères représentant une heure en un objet LocalTime.
- voirEmploiDuTemps(List<Cours> emploiDuTemps) : Affiche l'emploi du temps en parcourant les objets Cours et leurs détails.

6. La Classe GestionHoraireEtudiant

La classe GestionHoraireEtudiant est responsable de la gestion de l'emploi du temps de l'étudiant. Elle permet d'ajouter, supprimer et modifier des cours dans l'emploi du temps, ainsi que de gérer les conflits d'horaire.

- ajouterCours(List<Cours> emploiDuTemps, Scanner scanner) : Ajoute un nouveau cours à l'emploi du temps de l'étudiant en demandant les informations à l'utilisateur et en vérifiant les conflits d'horaire grâce aux méthodes ci-dessous.
- modifierCours(List<Cours> emploiDuTemps, Scanner scanner) : Permet de modifier les horaires d'un cours existant dans l'emploi du temps en demandant à l'utilisateur de choisir le type d'horaire à modifier.
- modifierHoraires(List<Horaire> horaires, String type, List<Cours> emploiDuTemps, Scanner scanner) : modifie les horaires des cours et tp
- modifierHorairesExamen(List<HoraireExamen> horairesExamen, List<Cours> emploiDuTemps, Scanner scanner) : modifie les horaires des examens
- supprimerCours(List<Cours> emploiDuTemps, Scanner scanner) : Supprime un cours de l'emploi du temps en demandant à l'utilisateur de fournir le numéro du cours.
- isConflictNewCours(List<Horaire> newHoraires, List<Cours> emploiDuTemps) : Vérifie s'il y a un conflit d'horaire entre les horaires d'un nouveau cours et les horaires des cours existants.
-

Rapport TP1 IFT1025

Abdel Rahman Ibrahim 20260406

- **isModificationConflit(Horaire newHoraire, List<Cours> emploiDuTemps)** : Vérifie s'il y a un conflit d'horaire lors de la modification d'un horaire de cours existant.
- **horaireConflict(Horaire horaire1, Horaire horaire2)** : Vérifie si deux horaires sont en conflit en comparant les jours de la semaine et les heures de début et de fin.
- **isConflictMemeCours(List<Horaire> magistralHoraires, List<Horaire> tpHoraires)** : Vérifie s'il y a un conflit d'horaire entre les horaires d'un cours magistral et les horaires d'un TP.
- **hasExamDateConflict(List<HoraireExamen> horaireExamen)** : Vérifie s'il y a un conflit d'horaire entre les dates des examens.
- **isExamConflictNewCours(List<HoraireExamen> horaireExamen, List<Cours> emploiDuTemps)** : Vérifie si un nouvel examen crée un conflit d'horaire avec d'autres examens dans l'emploi du temps.
- **isExamModifConflit(List<Cours> emploiDuTemps, LocalDate examDate**) : Vérifie si la modification d'un examen pour une date donnée crée un conflit d'horaire avec d'autres examens dans l'emploi du temps.

III. Mode d'emploi du programme

Le programme est conçu pour gérer l'emploi du temps d'un étudiant. Voici un mode d'emploi pour utiliser les fonctionnalités du programme :

Au démarrage du programme, l'utilisateur pouvez effectuer différentes actions en entrant les commandes correspondantes grâce à un menu qui est présenté à l'utilisateur. « 1 » pour ajouter....
Lorsque vous choisissez d'ajouter un cours, vous serez invité à saisir les informations nécessaires dans le format demandé, telles que le nom du cours, le numéro du cours, le nombre de crédits, les horaires du cours magistral, du TP et des examens. Il y a aussi la possibilité d'avoir plusieurs jours de cours magistraux, de TP, d'examens...
Lorsque vous choisissez de modifier un cours, vous devrez entrer le numéro du cours que vous souhaitez modifier. Si le cours est trouvé dans l'emploi du temps, vous pourrez choisir quel type d'horaire vous souhaitez modifier : cours magistral, TP ou examen. Bien sûr, les conflits entrainés par les modifications sont pris en compte.
Lorsque vous choisissez de supprimer un cours, vous devrez entrer le numéro du cours que vous souhaitez supprimer. Si le cours est trouvé dans l'emploi du temps, il sera supprimé de la liste.
Lorsque vous choisissez d'afficher l'emploi du temps, il sera affiché à l'écran, montrant tous les cours actuellement ajoutés, ainsi que leurs horaires correspondants.
Le programme effectue également des vérifications pour éviter les conflits d'horaire. Par exemple, il vérifie si un nouveau cours entre en conflit avec un cours existant ou si deux examens sont prévus pour la même date. Si un conflit est détecté, le programme affiche un message d'erreur et vous demande de choisir une autre option.
Lorsque votre tâche choisie est terminée, vous retournez au menu principal, où vous pouvez toujours quitter avec la touche « 6 »

IV. Bilan du travail

Ce TP n'était pas simple. Nous avons passé beaucoup de temps à réfléchir à la conception d'un cours, à sa structure, afin que la gestion de conflits soit la plus simple et efficace possible, car il y a beaucoup de possibilités : Conflits d'horaires entre cours magistraux, cours magistral et TP, TP et TP...au sein d'un même cours, entre cours différents...

Le résultat auquel nous avons abouti semble marcher (d'après les tests unitaires et les nombreux tests fait de notre côté), même si ce n'est peut-être pas la conception la plus efficace d'un programme pareil. En particulier, les méthodes de gestion de conflit.