Yohann Manseau-Glémot (20217138) Alex Chevrier (20216495) 7/12/2022

IFT1025 Devoir 1: Conception d'un système de création d'horaire universitaire

Classes:

- **Cours**:
  - La classe **Cours** contient la définition d'attributs relatifs à un cours universitaire, c'est-à-dire: le **numéro du cours**, la **matière**, le **jour**, **l'heure de début**, **l'heure de fin** ainsi que le nombre de **crédits** du cours.
  - Cette classe contient un constructeur **Cours** prenant en paramètre les attributs décrit ci-dessus. Ce constructeur sera utile pour créer des objets **cours** avec les entrées de l'utilisateur.
  - Finalement, la classe cours contient des méthodes **get** et **set** pour chacun des attributs mentionnés si-haut. Ces méthodes permettent d'accéder et de modifier les attributs initialisés au début. Exemple: méthodes **getMatiere** et **setMatiere** pour la matière d'un cours.
- **BuildHoraire:**
  - La classe **BuildHoraire** permet de vérifier la présence de conflits d'horaire, de calculer le nombre de crédits total et de s'assurer qu'il ne dépasse pas 15 et de construire l'horaire (graphiquement).
  - Les méthodes **BuildHoraire** et **build** permettent de créer un tableau en deux dimensions de 5x14 (5 jours semaine 14h par jour) et de remplir chaque espace du tableau avec le boolean false.
  - la méthode **calculCredits** permet de vérifier que le total du nombre de crédits pour la liste de cours de l'utilisateur ne dépasse pas 15 crédits. Si le cours courant dépasse 15, l'indice du cours précédent est conservé pour la suite (le cours qui

dépasse 15 n'est pas inclus)

- La méthode **jourEnInt** permet de transformer les différents jours de la semaine qui sont en string en int et chaque jour est associé à un numéro

- La méthode **afficherHoraire** permet de remplir toutes les lignes et colonnes, sauf les deux premières qui seront les heures verticalement et les jours horizontalement. Elle permet aussi de concaténer la matière et le numéro du cours (EX: IFT1025) et de placer les cours vis-à-vis de la bonne journée et pour le bon nombre d'heures.

- La méthode **remplirHoraire** permet de gérer les conflits d'horaire. Si la liste de cours contient un seul cours, la valeur false associée à la plage horaire du cours est remplacée par true. Si la liste contient plus que 1 cours, on vérifie si la valeur true est déjà présente dans cette plage horaire. S'il s'agit de la valeur true, il y a un conflit d'horaire. Sinon, on continue de parcourir la liste.

- **Main**

  - La classe **Main** permet de demander à l'utilisateur d'**ajouter** des cours à sa liste de cours, **modifier** sa liste de cours et **supprimer** des cours. Un objet scanner est créer pour récolter les entrées de l'utilisateur, c'est-à-dire les attributs de la classe **Cours**. Des méthodes static propres à chaques attributs sont utilisés pour faciliter la gestion des erreurs

  - Un scanner static est déclaré au début de la classe et est utilisé dans tout le reste de la classe

  - Lorsque tous les attributs nécessaires d'un cours sont entré par l'utilisateur, un nouvel objet cours est créé grace au constructeur de la classe Cours.

  - Chaque objet **Cours** créé par l'utilisateur est ajouter à une **ArrayList** (**liste**)

  - L'utilisateur doit s'inscrire à tous les cours dont il est intéressé avant de pouvoir en retirer ou en modifier

- Lorsque l'utilisateur est satisfait de son horaire, la classe BuildHoraire est appelée. Si un conflit d'horaire est présent, l'utilisateur devra modifier le cours entrant en conflit. On demande à nouveau d'entrer les détails du cours (Nouvelle journée, Nouvelle heurede début et Nouvelle heure de fin).

- La gestion d'erreur d'entrés est gérée avec plusieurs méthodes "try-catch". Par exemple, la matière du cours ne doit contenir que trois caractères et aucun chiffre. Si l'utilisateur entre une valeur erronée, on "trow" une exception décrite par la classe **InputException** ou une exception déjà définie dépendamment des cas.

- **InputException**

  - Contient le constructeur **InputException**

  - Retourne la string spécifique à une erreur. Elle permet d'afficher un message d'erreur (string) unique pour chacun des attribus de la classe cours demandé à l'utilisateur .

Mode d'emploi du programme:

Dans un premier temps, l'utilisateur se fait demander de créer un premier cours pour son horaire. On lui demande la matière, le numéro, la journée, l'heure de début, l'heure de fin et le nombre de crédits. L'utilisateur doit respecter les conditions spécifié entre paranthèse pour chaque attribus, sinon une erreur surviendra et la question sera demandée à nouveau. Celui-ci peux ajouter au maximum 10 cours dans son horraire, il ne peut pas dépasser le nombre de crédit maximal donné et il doit éviter les conflits d'horaire. Si un conflit d'horaire survient lors de la création de l'horaire, alors l'utilisateur sera encourager à changer le cours qui pose problème. Si le nombre de crédit total de la liste de cours dépasse 15, l'horaire sera consruit seulement avec les cours précédant celui qui fait dépasser 15. L'utilisateur peut également retirer des cours, mais seulement lorsqu'il a terminer d'ajouter des nouveau cours. Il peut

également modifier son horaire s'il n'est pas satisfait du résultat final.

Bilan qualitatif du travail, difficultés rencontrées, critiques et suggestions

Le but de ce travail était de construire un programme permettant de créer et de modifier un horraire étudiant. Nous avons créé 4 classes pour pouvoir faire ce travail, c'est-à-dire la classe Main (classe permettant de d'obtenir les entrées de l'utilisateur et de gérer les erreurs), BuildHoraire (permet de gérer les conflits d'horraire et d'afficher l'horaire adéquatement), Cours (permet de créer des objets cours grace aux entrées de l'utilisateur) et finalement la classe InputException (Contient le constructeur et permet de retourner un message d'erreur spécifique). La partie la plus difficile de ce travail était la gestion des conflits d'horaire et les exceptions. En effet, la gestion de conflits d'horaire posait problème, puisque nous ne savions pas comment procéder pour vérifier si un cours était déja présent dans la plage horaire spécifée. La gestion d'erreur d'entrées était également difficile, puisqu'il faut penser à chaque erreur possible que l'utilisateur peut comettre. Finalement, une suggestion pouvant améliorer la conception de ce travail est de donner un peu plus de détails sur le résultat final attendu et sur le fonctionnement désiré du travail.