Introduction

Dans ce sixième laboratoire de POO, le but était de réaliser un programme en suivant un schéma UML qui nous était donné. Ce schéma modélisait des personnes qui pouvaient être soit des étudiants, soit des enseignants. Ces étudiants pouvaient être en groupe. Les groupes, ainsi que les professeurs ont des leçons qui composent leur horaire.

Ce rapport a pour but de présenter nos choix les plus importants concernant ce laboratoire.

Choix d'implémentation

Cette section présentera brièvement les choix réalisés.

Visibilités de certaines fonctions

Les fonctions qui étaient sur le schéma avaient déjà leur visibilité indiquée. Cependant, nous avons rajoutés certaines fonctions supplémentaires qui n'étaient pas sur le schéma. Cela concerne notamment les fonctions "setter" qui nous permettent de garder une intégrité dans les données. Par exemple, lorsque l'on crée un groupe d'étudiants, il faut que le groupe référence les étudiants qui le compose et les étudiants doivent aussi connaitre leur groupe.

On ne pouvait pas utiliser d'autres méthodes publiques comme indiqué dans la consigne, alors nous avons décidé que ces "setter" allaient avoir la visibilité par défaut dite "package". De cette manière, la fonction ne sera pas publique à toutes les autres classes, mais on pourra tout de même l'utiliser dans le package qui compose notre application.

Méthode d'affichage de l'horaire

Pour ce point, on aborde la méthode affichant la grille d'horaire dans la classe "Lecon". On se focalise notamment sur la taille de cette fonction qui est tout de même conséquente. L'explication d'une telle taille s'explique car on ne pouvait pas rajouter d'attributs supplémentaires dans la classe ou d'autres classes pour avoir les caractères et les spécificités de l'affichage comme nous l'a proposé M.Krähenbühl.

La séparation de la fonction en plusieurs plus petites fonctions empirait nettement la lisibilité du programme et n'était pas du pratique à l'utilisation. Pour ces raisons, nous avons décidé de garder la fonction comme elle est.

Vérification des contraintes d'intégrité

Par contraintes d'intégrité, on parle notamment de la vérification des entrées lors de la création d'une leçon ou d'autres objets. Dans le cas d'une leçon, ça serait de vérifier que le jour, heure ou encore durée donnée soit cohérent. Pareil pour la vérification que deux leçons ne se chevauchent pas.

En parlant avec les enseignants, nous avons eu l'information que cette partie-là n'était pas en accord avec ce qui était réellement demandé dans le laboratoire. Nous avons donc décider de ne réaliser aucune vérification d'intégrité à ce niveau.

La seule vérification que l'on a réalisée est qu'il est obligatoire d'avoir au moins un étudiant lors de la création d'un groupe. Ainsi, on respecte totalement le diagramme UML qui nous était fourni.

Choix des types de tableaux

Au début, nous étions partis sur des tableaux à taille fixe. Cependant, nous nous sommes retrouvés bloqués lorsque nous avons essayés de garder la cohérence au sein des données comme expliqué dans le premier point des choix d'implémentation.

Nous avons donc modifié notre programme afin d'avoir des tableaux à taille variable. De cette manière, on pouvait simplement ajouter dans les instances voulues les informations manquantes avec nos "setter".