MGL7460
Rapport du TP 1

Membres de l'équipe

- Emmanuelle LIARD
- Alireza NEZAMI
- Djamel HADJ MOHAMMED

URL des dépôts GIT-lab

Partie Java :
https://gitlab.info.uqam.ca/mgl7460/tp1_hadj_mohammed_liard_nezami

Partie Python :
https://gitlab.info.uqam.ca/mgl7460/tp1_hadj_mohammed_liard_nezami_python

I. Niveau d'expertise de l'équipe

Membre	Niveau expertise Java (1 à 5)	Niveau expertise Python (1 à 5)	Commentaires
Emmanuelle Liard	3	2	Dispose de très bonnes compétences techniques et d'expériences académiques et professionnelles en Java. Ces compétences ont pu être transférées aisément en Python, langage très peu utilisé jusque-là.
Alireza Nezami	2	2	Je n'ai fait pas beaucoup de python en orienté objet (je fais Python sur Data mining) et c'était intéressant OOP en Python. Pour Java OOP, je l'ai fait il y a 12 ans (un projet en industrielle) et maintenant, après beaucoup d'années, je me souviens et apprends beaucoup de choses.
Djamel Hadj Mohammed	2	2	

II. Facilité d'implanter le modèle

A. Évaluation pour Java : {Faiblement, Moyennement, Fortement}

Résultat de l'implémentation en Java : **Fortement**.

Il a été fortement possible d'implémenter le modèle. Le diagramme de classe ayant déjà été fourni, il a été rapidement et efficacement implémenté.

Toutes les exigences et aspects du modèle ont été abordées et ont pu être traitées, notamment les interfaces, les listes, les associations entre les classes, et en particulier celle entre les classes <Etudiant-Inscription-GroupeCours>.

L'utilisation des types statiques de ce langage pour définir les variables a été d'une grande aide (chaînes de caractères, entiers, booléens, « ArrayList », « Hashmap », etc).

En ce qui concerne les tests unitaires, l'utilisation de *JUnit* et des composants qu'il offre a été suffisant pour la réalisation de nos tâches.

B. Évaluation pour Python : {Faiblement, Moyennement, Fortement}

Résultat de l'implémentation en Python : **Fortement**.

Dans l'ensemble, il a été possible d'implémenter le modèle, qui a été fourni dans la partie Java, avec ce langage.

La particularité de Python par rapport à Java est son typage dynamique. Pour cette raison, l'initialisation de listes dans les constructeurs des méthodes et l'ajout d'éléments dans celles-ci a été plus complexe à réaliser.

Les éléments ont pu être transposés d'un langage à un autre et le modèle a pu être implémenté le plus fidèlement possible.

De même qu'en Java, l'utilisation des types prédéfinis a été suffisante. Les classes concrètes ont été uniquement implémentées, conformément aux directives, bien qu'il soit possible de réaliser des interfaces via des modules existants.

En ce qui concerne les tests, l'utilisation de *Unittest* et de ses composants à également été suffisante. Les tests sont la transposition en Python de ceux de la partie Java.

III. « Expérience développeur »

A. Le langage Java

1. Rapidité de codage

Ayant de l'expérience en Java, il a été facile d'implémenter les interfaces fournies. En une semaine la majorité des implémentations ont été réalisées.

Toutefois, la méthode la plus complexe implémentée a été la liaison entre *Etudiant*, *GroupeCours* et *Inscription*. C'est-à-dire, le fait que la classe *Etudiant* possèdent une méthode permettant d'inscrire un étudiant à son groupe cours, et inversement, en passant par la classe *Inscription*.

La subtilité qui a été induite a compliqué la tâche : en inscrivant *l'Etudiant*, la liaison avec *GroupeCours* devait s'opérer automatiquement et en l'inscrivant via *GroupeCours*, l'inscription devait figurer parmi les attributs de cet(te) étudiant(e). Il a fallu trouver un stratagème pour éviter les boucles infinies et accomplir cette tâche.

Parmi toutes les solutions, possibles, celle qui a été implémentée répond parfaitement au besoin. Néanmoins, elle est perfectible car sa complexité peut être diminuée.

2. Lisibilité

Nous avons trouvé que le code était facile à lire et à comprendre. Que ce soit en s'aidant de la documentation présente sur Internet, que des méthodes déclarées via la Javadoc et les commentaires du code.

La logique du code en elle-même est facilement compréhensible et la nomenclature des méthodes (celles déjà implémentées par le langage et celles que nous avons réalisées) est un soutien à la compréhension du but recherché.

B. Le langage Python

1. Rapidité de codage

Cette partie du projet a pris plus de temps à être réalisée que l'autre.
Cela est dû en majorité au manque d'expérience des membres du groupe dans ce langage et à ses subtilités : indentations, manque de points-virgules en bout de lignes de code, différents noms de méthodes prédéfinies, etc.

Le caractère non typé des variables n'a pas, non plus, été d'une grande aide lors de l'implémentation des constructeurs des classes. Par exemple, l'erreur « *'str' object has no attribute 'append'* » a souvent été rencontrée. Il a été possible de la résoudre mais elle est restée une erreur récurrente.

Un autre problème rencontré de façon régulière a été lié à l'import de méthodes d'autres classes situées dans d'autres packages. Faire comprendre au langage/IDE à quelle méthode nous faisions allusion n'a pas été aussi intuitif que voulu.

Bien que la logique d'implémentation soit la même, la réalisation du modèle en Python a été deux fois plus longue que celle en Java.

2. Lisibilité

La lisibilité du code est facilitée avec la nomenclature des méthodes. Sans elle, nous pensons qu'il aurait été plus compliqué de lire et de comprendre le programme dans ce langage. La présence des indentations pour marquer les différents blocs de code est perturbante. En effet, s'il en manque une, ou qu'il y en a une en trop, le code a une tout autre signification.

De plus, comparé à Java, certaines méthodes n'ont pas de nom suffisamment explicite si on tente de les comprendre sans documentation. Par exemple, la méthode « *__eq__(Object)* » est l'équivalent en Java de « *equals(Object)* ». Selon nous cette appellation est plus ambiguë et nécessite une plus grande vigilance concernant la compréhension.

Pour conclure, nous trouvons que le langage Java est plus lisible que Python.

IV. Sécurité du code.

A. Le langage Java

Quelques erreurs ont été trouvées à la compilation et l'exécution du programme en Java. Étant un langage que nous trouvons intuitif, il a été aisé de les comprendre et de les résoudre.

B. Le langage Python

En revanche, cela a été tout le contraire avec Python. Beaucoup d'erreurs ont été trouvées à la compilation, à l'exécution, et il a été compliqué de les résoudre.

Bien que la logique d'implémentation soit la même qu'en Java, encore une fois les subtilités du langage python ne nous ont pas été d'un grand secours. L'une des erreurs récurrentes que nous avons rencontrées était : « *'X' has not attribute 'Y'* ». Alors que logiquement si… De même, l'import des méthodes d'autres dossiers n'a pas été chose simple. Elles ont toutes été résolues, toutefois avec plus de temps qu'il n'en a fallu en Java.

Remarque : la répartition du travail n'a pas été équitable tout au long du projet.