MGL7010 Énoncé du TP 1

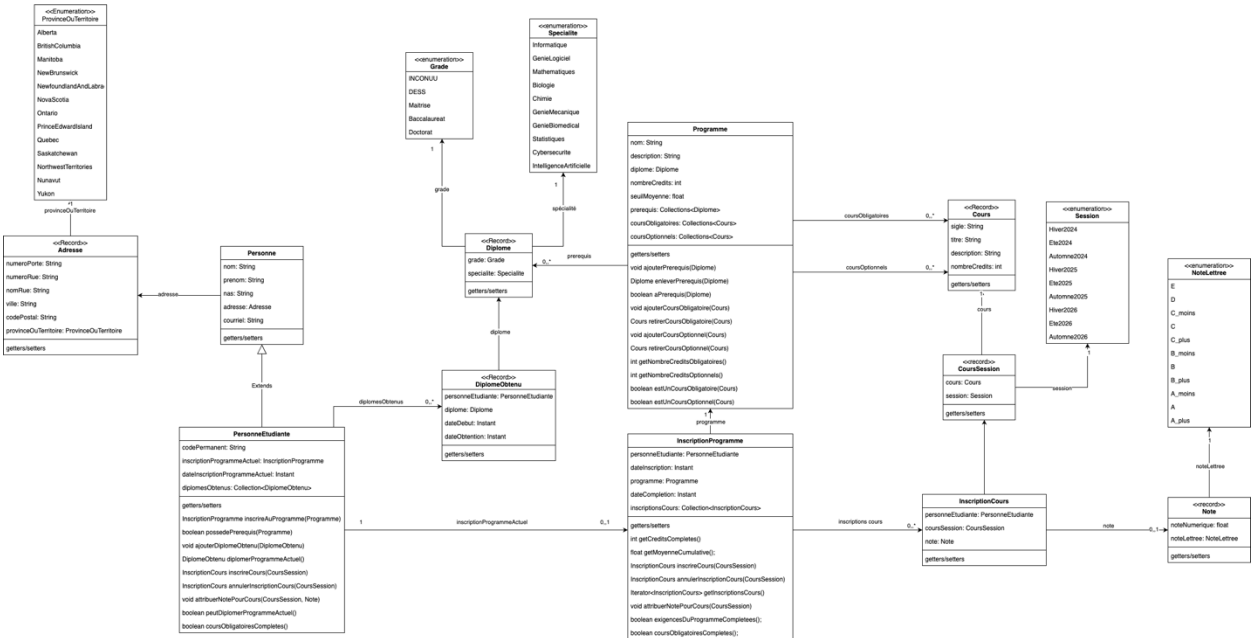
Hiver 2026

Version 0.8

Objectif technique de l'exercice

Le but de ce travail est de développer un embryon d'un système de gestion de dossiers académique (programmes, diplômes, cours, cours-session, inscription).

La figure suivante montre le modèle de classes qui sera expliqué en classe.



Pour le moment, il n'y a pas d'interface graphique. Il « suffit » d'implanter les fonctionnalités permettant de :

1) Créer des **`Cours`**, décrits par un sigle, un titre, une description, et un nombre de crédits
2) Créer des **`Programme`**`'s`, décrits par un nom, une description, l'identification du Diplôme que le programme permet d'obtenir, du nombre total de crédits, du seuil minimal de moyenne pour « graduer »/diplômer, et des listes des cours obligatoires et optionnels
    a. Plein de méthodes pour gérer les attributs et les deux listes de cours
3) Créer une **`PersonneEtudiante`** et :
    a. Vérifier son admissibilité à un programme (`boolean possedePrerequis(Programme);`)

   b. L'inscrire à un programme d'études, retournant une instance de **`InscriptionProgramme`** (`InscriptionProgramme inscrireAuProgramme(Programme);`)
   c. L'inscrire à des cours dans le cadre du programme, donnant lieu à des **`InscriptionCours`** (`InscriptionCours inscrireCours(CoursSession);`):
   d. L'inscription se fait à des **`CoursSession`**`'s` qui sont différentes offres du **`Cours`** à différentes **`Session`**
   e. Lui attribuer une note pour le cours (`void attribuerNotePourCours(CoursSession, Note);`)
   f. Vérifier si elle a satisfait les exigences pour diplômer du programme actuel (`boolean peutDiplomerProgrammeActuel();`)
   g. La « diplômer » pour le programme actuel (`DiplomeObtenu diplomerProgrammeActuel();`)
   h. Ajouter des diplômes « à son CV » (`void ajouterDiplomeObtenu(DiplomeObtenu);`)
   i. Accéder à ses divers attributs (getters/setters)
4) Pour une **`InscriptionProgramme`** donnée (correspondant à l'inscription d'une personne étudiante à une programme donné), je peux :
   a. Obtenir le nombre de crédits complétés à date (`int getCreditsCompletes();`)
   b. Obtenir la moyenne cumulative (`float getMoyenneCumulative();`)
   c. Inscrire la personne étudiante concernée à un cours dans le cadre du programme, donnant lieu à des **`InscriptionCours`** (`InscriptionCours inscrireCours(CoursSession);`):
   d. Annuler l'inscription de la personne concernée à un cours (`InscriptionCours annulerInscriptionCours(CoursSession);`)
   e. Obtenir la liste des **`InscriptionCours`** dans le cadre du programme
   f. Attribuer une note pour un cours (`void attribuerNotePourCours(CoursSession);`)
   g. Vérifier si les exigences du programme ont été complétées par la personne étudiante concernée (`boolean exigencesDuProgrammeCompletees();`)
5) Différentes classes utilitaires dont :
   a. Plusieurs *énumérations* (**`Grade`**, **`Specialite`**, **`Session`**, **`NoteLettree`**, **`ProvinceOuTerritoire`**)
   b. Des classes du type *record*, qui correspondent au struct en C/C++, avec juste des champs de données, et des accesseurs par défaut, mais sans 'comportement métier' (**`Diplome`**, **`DiplomeObtenu`**, **`CoursSession`**, **`Note`**, **`Cours`**)

Nous introduirons plus tard dans le cours (TP2 et TP3) des fonctionnalités plus élaborées.

Votre travail technique

J'ai déjà implanté le logiciel en question, en veillant à utiliser les bonnes pratiques en programmation et conception détaillée orientées-objet.

Entre autres, j'ai codé l'application en :

1) Distinguant entre *interfaces* et *classes* pour les classes les plus importantes, pour maximiser la réutilisation et minimiser les dépendances inutiles entre les différentes parties du code
2) En rédigeant des cas de tests sous la forme de classes de test JUnit 5.

Ce que je vais vous remettre :

1) Les interfaces Java, amplement documentées
2) Les classes utilitaires dont j'ai parlé précédemment
    a. Les *énumérations* **`Grade`**, **`Specialite`**, **`Session`**, **`NoteLettree`**, **`ProvinceOuTerritoire`**.
    b. Les *record* **`Diplome`**, **`DiplomeObtenu`**, **`CoursSession`**, **`Note`**, **`Cours`**
3) Les classes de test

Ce que vous aurez à faire :

1) Implanter les classes correspondantes aux interfaces
    a. Vous devez vous partager le travail de la façon suivante :
        i. Une personne implémente les interfaces **`Cours`**, **`InscriptionCours`**, et **`InscriptionProgramme`**
        ii. L'autre personne implémente les interfaces **`Personne`**, **`PersonneEtudiante`**, et **`Programme`**
    b. Vous devez collaborer via gitlab
2) Faire exécuter et réussir les tests
3) **Il ne faut pas modifier les classes de tests, d'aucune façon.**
    a. Pour exécuter votre code, je recopierai mes classes de tests dans le répertoire src/ca/uqam/mgl7010/tp1/tests, avant de lancer l'exécution

Vous serez évalué-e sur :

1) La qualité de votre implantation en Java
2) La mesure dans laquelle elle passe avec succès les divers cas de tests.
3) Un mini-rapport écrit
4) Votre explication orale sur le code

Modalités de remise :

1) Remettre le code (juste le code source, **pas de code compilé ni de librairies** (fichiers jar) et le rapport dans un fichier zip déposé dans Moodle
2) Remettre la même chose dans un projet gitlab ***privé*** auquel vous me donnerez accès (je ferai un « git clone »).

Barème

Critère	Poids
Du code qui passe réussit les tests en Java	50%
Bonne qualité de conception	25%
Documentation dans le code Java	5%
Qualité du rapport écrit	10%
Présentation orale	10%

Structure du rapport

Le rapport doit contenir trois sections, que voici, avec une page de couverture indiquant vos noms (sans le code permanent) :

1) Décrivez le niveau d'expertise de chacune des deux personnes en Java et en programmation OO, selon une échelle de 5 valeurs :
    1. Débutant : ce travail est le premier travail réalisé dans le langage
    2. Connaissances académiques. Appris dans le cadre d'un ou deux cours, avec quelques travaux de petite envergure réalisés avec le langage, ou expérience industrielle de moins de 3 mois
    3. Plusieurs cours académiques, avec plusieurs projets totalisant des milliers de lignes, ou expérience industrielle de 4 mois à un an.
    4. Expert. Expérience industrielle de plusieurs années, avec dizaines de milliers de lignes
    5. Guru. Expérience industrielle de plus de dix ans. Vous êtes la référence dans votre entreprise pour le langage. Vous avez un blogue. Vous avez inventé le langage, ou contribuez à son évolution.
2) Défis rencontrés dans la réalisation du travail. Les défis peuvent être de différents ordres :
    1. Défis « métiers » : comprendre la fonctionnalité souhaitée
    2. Défis « algorithmiques »
    3. Méconnaissance de Java
    4. Défis de travailler à deux sur le même projet
    5. Autres
3) Ce que vous avez appris de cet exercice, et de quel ordre.
    1. Vous pouvez reprendre les 5 points précédents