TD1 : Analyse et conception de service de scolarité

L'objectif de cette étude est de concevoir un système de scolarité d'une université. Nous souhaitons faire une analyse et une conception OO détaillées et sous différents angles de l'ensemble du système qui pourrait par la suite être implémenté dans un langage OO. À cette fin, nous examinerons un système de dépôt de devoir à rendre qui doit être utilisé pour gérer les rendus Les conditions requises pour ce système sont les suivantes :

- Chaque cours est affecté à des enseignants. Ceci est réalisé par un chargé de cours, qui est également un enseignant. Dans le cadre d'un cours, les enseignants peuvent créer un devoir (homework), corriger et attribuer des points ainsi qu'un avis sur les rendus des étudiants.
- Le chargé de cours définit quel enseignant évalue quel rendu. À la fin du cours, le chargé de cours organise également la remise des certificats. La note d'un étudiant est calculée en fonction du nombre total de points obtenus des devoirs remis.
- Les étudiants peuvent suivre des cours et télécharger les devoirs.
- Tous les utilisateurs (étudiants et enseignants) peuvent gérer leurs données utilisateur, afficher les cours et les devoirs définis pour les cours (à condition que l'utilisateur concerné soit impliqué dans le cours), et afficher les rendus ainsi que les points obtenus. Cependant, les étudiants ne peuvent consulter que leurs propres rendus et les notes correspondantes. Les enseignants ne peuvent consulter que les travaux qui leurs ont été attribués et les notes qu'ils ont attribuées. Le chargé de cours a des droits d'accès pour toutes les données.
- Un cours est créé/supprimé par un administrateur.
- Lorsqu'un cours est créé, au moins un chargé de cours y est affecté. D'autres enseignants peuvent être affectés ultérieurement, mais aussi, des affectations aux cours peuvent être supprimées. L'administrateur peut également supprimer des cours entiers.
- Les informations sur les utilisateurs et les administrateurs sont automatiquement transférées d'un autre système. Par conséquent, les fonctions permettant la création de données utilisateur ne sont pas nécessaires.
- Toute personne a un nom, un prénom, une adresse postale et une adresse mail. Un étudiant possède un numéro d'étudiant. Un enseignant possède deux numéros (Numem et Harpège). Un devoir est caractérisé par un nom, une description, deadline et le nombre de points. Un certificat de participation est délivré à la fin de chaque cours avec un total des points et une appréciation.

1 Partie UML

1.1 Analyse

Question 1 • A l'aide d'UML, modélisez les acteurs et les cas d'utilisation de la spécification ci-dessus.

Question 2 • Réalisez un diagramme de classes d'analyse.

1.2 Conception

Les cas d'utilisation et le diagramme de classes issus de la phase d'analyse ne sont généralement pas suffisamment détaillés pour la mise en œuvre réelle du système. Nous allons maintenant mettre en évidence les exigences auxquelles le système doit répondre. Pour cela, nous allons zoomer sur le système et modéliser son comportement.

Question 3 • Représentez les envois des messages pour les cas d'utilisations « *Rendre un devoir* », « *Affecter Enseignant* », « *Consulter Document* », « *Saisir Note* », « *Délivrer Certificat* ».

Question 4 • Reprenez le diagramme de classes d'analyse et ajouter les comportements des objets que vous avez modélisés dans le diagramme de séquence.

Question 5 • Modélisez les différents états-transitions des entités « *Rendu* » et « *Participation* ».

Question 6 • Révisez encore une fois le diagramme de classes en prenant en compte les états-transitions des entités « Rendu » et « Participation ».

Mettons maintenant en place une fabrique d'utilisateur.

Question 7 • Complétez le diagramme avec les classes concrètes *EtudiantCreator*, *EnseignantCreator* et *AdminCreator* qui héritent de *UtilisateurCreator*.

2 Partie Java

L'objectif de cette partie est d'implanter et de tester les différents diagrammes que vous avez réalisés dans la partie UML.

Question 8 • Reprenez vos diagrammes et implantez les différentes classes en commençant par les plus simples. Pour cela, générez automatiquement un premier jet du code à l'aide de `starUML` ; vérifiez le code comparant à votre conception et corriger si besoin ; compléter votre code.

Question 9 • Ajoutez une classe `App` avec un `main` pour effectuer vos premiers tests. Afin de tester, nous vous proposons de créer une instance avec un cours, 2 devoirs, un administrateur, 2 enseignants et 2 étudiants.

Question 10 • Ajoutez des tests unitaires pour la classe "Cours" et "Rendu".