 Université BEN YOUCEF BEN KADDA

Faculté des sciences

Département Mathématique et informatique

|  |  |
| --- | --- |
| **Projet TP 02 :**   |  | | --- | | **Système de gestion d’une bibliothèque numérique** | |

Année universitaire : 2021/2022

**Réalisé par** : ZOUAI SERINE MARIA

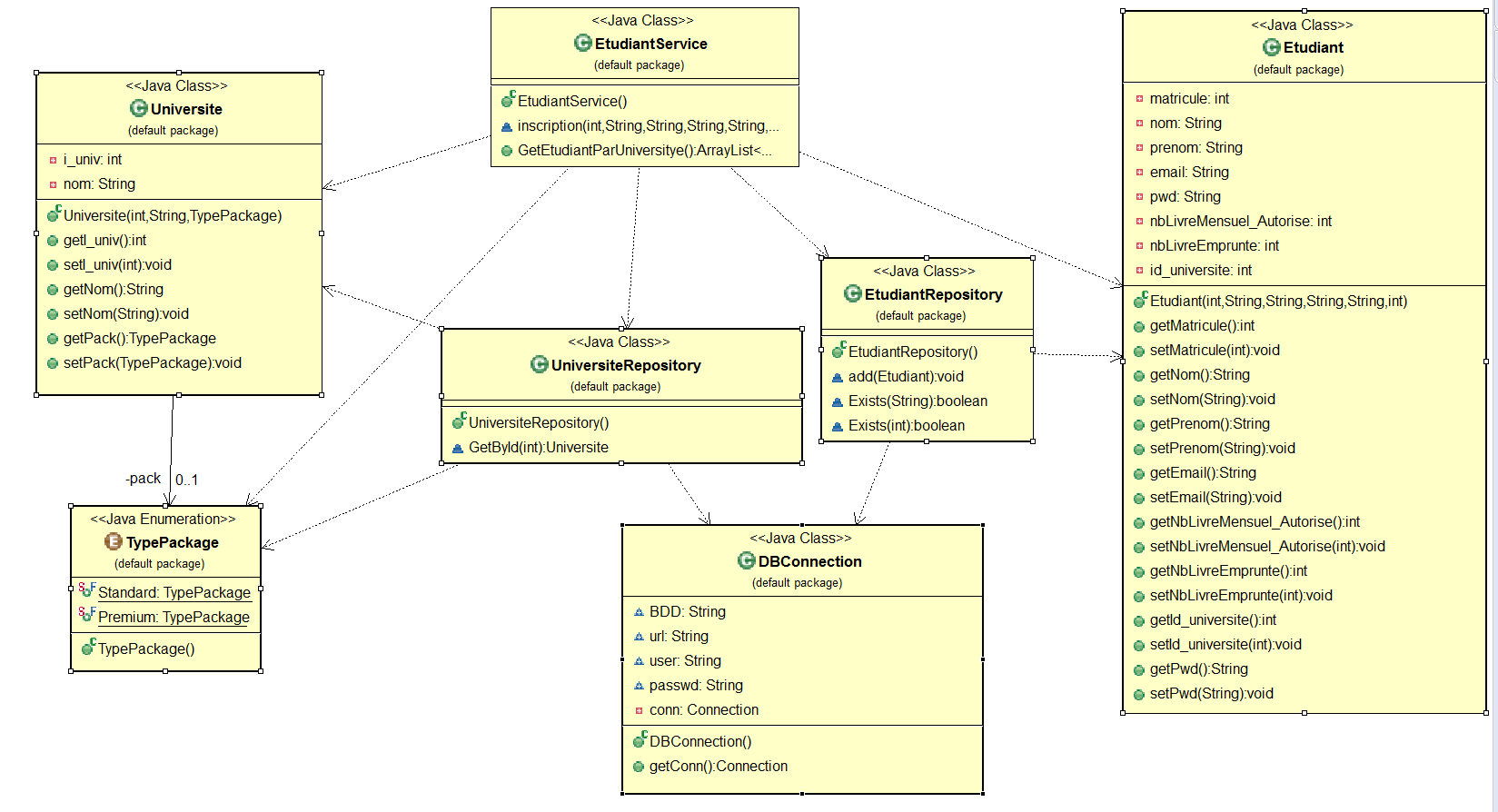
**Spécialité :** 1 ère année master ISII

**Chargé du projet :** Dr.GUENDOUZI Wassila

**Question 3 : Donnez le diagramme de dépendance entre ces classes**

Dans ce projet, nous avons 8 classe qui sont : DBconnection, Etudiant, EtudiantRepository, EtudiantService, Universite, UniversiteRepository, TypePackage et enfin la classe MainApp.

Nous allons à présent, illustrer le diagramme de dépendances entre ces classe et en se basant sur l’utilisation d’un logiciel de schématisation **Eclipse ObjectAid.**



**Remarque** : on a enlevé la classe MainApp qui est dans notre cas dépendante de la classe EtudiantService, car la classe MainApp peut utiliser toutes les entités de notre application et pas seulement la classe EtudiantService.

On dit qu’une Classe A dépend d’une classe B , lorsque A utilise B comme attribut , possède une opération dont le type de l’un de ces paramètre est B….

La classe la plus stable est : EtudiantService, et la classe la moins stable est celle de type Repository(EtudiantRepository) et qui a un accès à la base de donnée.

**Question 6 : Objectif du principe de l’inversion de contrôle**

L’objectif du principe de l’inversion de contrôle (inversion des dépendances) qui est présenté comme l’un des principes SOLID le D plus précisément est : appliquer la stabilité dans l’application, diminuer le couplage entre deux modules , s’assurer que les modules doivent dépendre de modules plus stable qu’eux et que les modules de haut niveau doivent interagir avec les abstractions comme les interfaces.

-En ce qui concerne **la question 12** au sujet des responsabilités dans la méthode inscription, j’ai cité auparavant que c’était momentanément acceptable mais il pourrait avoir un non-respect de l’un des principes SOLID plus précisément le S qui indique qu’une classe ne doit comporter qu’une seule responsabilité. La classe qui a plusieurs responsabilités est une classe qui contient beaucoup d’attributs, beaucoup de méthodes publiques...

Comme on peut le remarquer notre fonction inscription contient non seulement beaucoup d’attributs et on lui a associé plus qu’une responsabilité : elle est chargée d’ajouter un étudiant mais aussi elle vérifie le format de l’email, vérifie l’existence de l’email et du matricule et aussi elle initialise le nombre de livre mensuel autorisé. Par conséquent, au lieu que les paramètres de la méthode inscription soient des attributs d’une classe qui est Etudiant dans notre cas, nous les remplaceront par un objet de cette classe et ajouter d’autre types si nécessaire.

**Question 17 :**

A l’initialisation du nombre de livre mensuel autorisé ou à l’ajout du bonus, les traitements dans les deux cas dépendent du forfait de l’université à laquelle appartient l’étudiant. Cependant, la méthode n’a pas respecté le principe O (ouvert aux extensions et fermés aux modifications) surtout avec l’utilisation des if else .

Comme solution, nous avons adopté l’utilisation du patron abstract factory.

**Question 25 :**

Afin de réaliser ça, nous allons ajouter une classe abstract AbstractViewInscription qui va appliquer la dépendances dans la classe ControleurInscription.

**Question 29 : Donner le diagramme de dépendances entre les packages**

A l’aide du logiciel StarUml, le diagramme de dépendances entre packages est définit comme suit :

