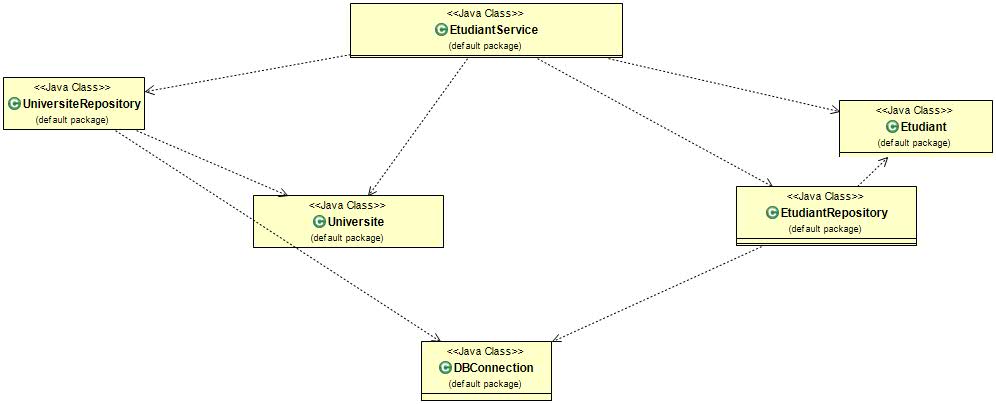
**Projet Tp Architecture des Logiciels**

Racha Bekhouche

Master1 ISII

1.Diagramme de dépendance



2. Le principe d’inversion de contrôle

On peut voir que la classe Etudiant Service dépend fortement des classes Universite, UniversiteRepository, Etudiant et EtudiantRepository.

L’objectif étant de stabiliser l’application, nous avons implémenté une solution où nous avons utilisé des interfaces : InterfaceEtudiant, InterfaceEtudiantRepository, InterfaceEtudiantService, InterfaceUniversite, InterfaceUniversiteRepository et InterfaceDBConnecion.

Ainsi la classe EtudiantService utilise des Interfaces au lieu d’utiliser des classes concrètes, ce qui permet de stabiliser notre application.

10. La méthode « inscription » de la classe « EtudiantService » ajoute un étudiant, vérifie le format de l’email, vérifie l’existence de l’email et du matricule et initialise aussi le nombre de livre mensuel autorisé.

Cependant nous avons pensé que cette solution n’est pas optimale car elle ne respecte pas le Principe du Single Responsability.

Pour optimiser notre application nous avons créé deux nouvelles méthodes :

Une qui vérifie l’existence, le format de l’email et l’existence du matricule « EtudiantVerification » et une autre qui initialise le nombre de livre «SetNbrLivreMensuelAutorisé ».

La methode Inscription prendre donc plusieurs paramètres ( Etudiant, EtudiantRepository, Universite et UniversiteRepository) et appellera les méthode afin d’effectuer l’inscription.

15 Ici, plusieurs conditions sont présentent et dictent la stratégie de cette dernière. Ainsi classe ne respecte pas le principe O

23. Afin que le contrôleur « ControleurInscription » dépende de l’abstraction de la présentation et non pas de son implémentation nous avons créé une classe Abstraite AbstractViewInscription et qui applique l’injection de dépendance dans la classeControleurInscription.

27.

