Projet d’architecture logicielle :

Assessment Manager

# Descriptif du projet

Ce projet réalisé en java est une application utilisable en invite de commande. Cette application du nom « Assessment Manager » a pour objectif d’aider un professeur à gérer l’enregistrement des évaluations de ses élèves. En plus de cette fonctionnalité, nous proposons également la visualisation de statistiques sur les notes et les élèves d’un cours telles que l’évolution moyenne des notes (en fonction des années). Il est également possible de visualiser un graphe comportant les noms des élèves les points qu’ils ont obtenus ou un histogramme qui comportera les points obtenus et le nombre d’occurrences de ces mêmes points.

# Diagramme de classe

# 

* Une école possède un ensemble de professeurs.

Si l’école ferme, le professeur existe toujours.

* Chaque professeur donne une liste de cours.

Si le professeur arrête de donner cours, son cours peut être repris par quelqu’un d’autre.

* Un cours contient un ensemble d’évaluations, si le cours disparait, les évaluations ne servent plus à rien donc elles disparaissent aussi.
* Un cours contient un ensemble d’élèves. Si le cours est supprimé, l’élève existe toujours.

# Design pattern

2 designs pattern nous ont principalement aidé à réaliser l’implémentation de nos classes, il s’agit du design pattern abstract factory, ainsi que le composite design pattern. Le premier nous permettant de répondre à la problématique de création des objets ainsi que le deuxième qui nous a aidé à structurer notre projet.

## Abstract factory design pattern

Parmi les designs pattern existant, nous nous sommes inspirés du factory design pattern qui nous permet de créer une classe abstraite « Personne » qui sera implémentée par « Etudiant » et « Professeur », cela nous permet donc d’éviter de la duplication de code, notamment en nous offrant la possibilité de rajouter des attributs et méthodes dans la classe personne (comme par exemple, la date de naissance d’une personne).

## Composite design pattern

Ce design pattern nous a inspiré dans la conception de notre application car nous avons globalement une structure arborescente. En effet, un objet de la classe « Professeur » possède une liste de « Cours », cette même liste de cours possède des « Evaluations » ainsi que des « Etudiants ». On peut également constater qu’en remontant l’arborescence de professeur, on constate que celui-ci fait partie de l’objet « Ecole »

# Critères de qualité

## Maintenabilité

Nous avons décidé que le code devait être maintenable notamment pour la prochaine équipe. Pour réaliser cela, nous avons fait en sorte que les classes soient séparées dans des fichiers bien définis. On peut mesurer ce critère de qualité en mettant en évidence le nombre de composants de notre programme. Nous avons également tenu à rendre la documentation (ce fichier ci donc) la plus claire possible. De plus, dans la mesure du possible nous avons tentés de limiter la duplication de code grâce à la création de nos classes tel que la classe « Personne » qui est héritée par « Professeur » et « Elève ».

## Reliability/Fiabilité

Nous avons opté pour le critère de qualité Reliability car il est important que l’application fonctionne bien dans tous les cas de figures qui puissent se présenter, ou du moins dans un maximum. Pour atteindre cet objectif, nous avons mis en place un ensemble de tests unitaire qui vérifient si l’application fait bien ce qu’elle est censée faire. Encore une fois, ce critère de qualité est mesurable. En effet, via les tests unitaires, on obtient directement le code coverage. Attention toutefois, le nombres de tests est actuellement limité et toutes les fonctions créées n’ont pas encore été testées. Ceux-ci sont exécuté dans la « testmainclass ». Il serait également intéressant dans un futur d’automatiser ces tests.

## Extensibility

Notre application est ouverte à l’extension. En effet, notre programme étant assez simpliste, il est fort probable que dans l’avenir, les développeurs soient tentés de rajouter de nouvelles fonctionnalités, c’est pourquoi nous avons facilité cela autant que possible en choisissant notamment des relations entre les différentes classes adaptées.

## Usability

Notre application étant destiné à des utilisateurs pas nécessairement spécialisé en informatique, nous avons opté pour une interface assez simpliste et proche de l’utilisateur qui lui met à disposition une application user-friendly grâce à l’invite de commande créé par Yannis et testée par les 2 juliens qui après confirment qu’elle est bien ergonomique. Attention toutefois, par manque de temps nous n’avons pas pu avoir autant de feedback qu’en le faisant tester à un groupe d’utilisateur externe.

# Spécifications

## Enregistrement des objets

En raison du manque de temps, nous n’avons pas pu réaliser l’enregistrement de nos objets dans un dictionnaire Json. Toutefois, nous avons décidé de faire hériter toutes nos classes de la classe « Serializable », ainsi vous pouvez constater un exemple en début de notre programme de création d’un objet « School » qui est sérialisé et enregistré dans un fichier texte. Nous récupérons ensuite ce même fichier pour le désérialiser et retrouver l’objet d’origine avec toutes ces informations. Cette technique nous offre donc une perspective pour l’enregistrement des données.