Dans ce TD vous allez construire un diagramme de classes à partir de code existant que vous ne connaissez pas. Vous allez également écrire des commentaires pour ce code. Les objectifs d'apprentissage sont :

- manipuler le formalisme diagramme de classes UML ;
- comprendre ce qu'est un bon diagramme de classes par rapport à une utilisation donnée.

Vous avez également un *objectif de résultat* :

- augmenter du code existant pour le rendre plus facile à utiliser par un autre développeur.

1 Introduction

Un diagramme de classes peut être utilisé :

- dans la phase de conception d'une application pour décider de la décomposition en classes en vue de la résolution d'un problème ;
- pour documenter une application existante.

Le diagramme de classes doit se concentrer sur les éléments importants de l'architecture, ceux qui apportent de l'information utile.

Information utile dans un diagramme de classes. L'utilité d'une élément du diagramme dépend de ce à quoi le diagramme va servir.

Certaines informations sont presque toujours inutiles, comme par exemple en java les méthodes `equals`, `toString` et `hashCode` héritées de la classe `Object`.

La présence d'un constructeur est souvent une information inutile, comme par exemple qu'un point 2D est construit à partir de deux coordonnées :
`Point2D(double x, double y)`

Mais dans certains cas, préciser le constructeur est utile. Par exemple, soit la classe `Personne` qui a comme attributs `String nom; int age; String email;`. La présence du constructeur `Personne (String nom)` indique que seul le nom est un attribut obligatoire, on va donc l'inclure dans le diagramme de classe si cette information est utile.

De même, les *getters* et *setters* sont souvent inutiles dans un diagramme de classes, sauf si on veut insister sur le fait que certains des attributs ne peuvent être modifiés.

Un bon diagramme de classes est un diagramme le plus concis possible, mais qui reprend les éléments essentiels par rapport aux objectifs poursuivis.

Utilité des diagrammes de classes à construire aujourd'hui. Aujourd'hui vous allez construire des diagrammes de classes d'applications existantes avec l'objectif de **faciliter la modification de ces applications** en vue de leur ajouter de nouvelles fonctionnalités ou de corriger des erreurs.

C'est à dire, votre diagramme de classes doit aider un programmeur à effectuer ces tâches avec le moins d'effort possible :

- comprendre l'architecture globale de l'application ;
- comprendre la responsabilité de chacune des classes ;
- identifier la ou les classes qui sont responsables pour une fonctionnalité donnée ;
- localiser les endroits du code à modifier si on veut corriger une fonctionnalité existante ou en ajouter une nouvelle.

Faites ce travail avec sérieux : dans une prochaine séance votre diagramme de classes sera utilisé par un autre étudiant qui devra effectuer des modifications.

2 Travail préparatoire

Vous allez travailler sur trois applications différentes, toutes issues des projets agiles du début d'année :

application numéro 0 Monopoly tradition

application numéro 1 Drag'genda

application numéro 2 Luxury casino

Déterminer sur quelle application je dois travailler. Soit n le numéro de la machine de TP que je suis en train d'utiliser. Soit r le reste de la division entière de n par 3. Je prends l'application numéro r. Si je suis parmi les auteurs l'application numéro r, alors j'échange avec un de mes voisins. Je garde l'application choisie jusque la fin de la séance de TP.

Télécharger les fichiers nécessaires au TP. Vous devez cloner les dépôts git. Voici les liens :

application numéro 0 Monopoly tradition `https://github.com/ptitguigui/Projet-agile-.git`

application numéro 1 Drag'genda `https://github.com/Azzerood/ProjetAgile11.git`

application numéro 2 Luxury casino `https://github.com/RachelDiaz/Projet_Agile.git`

3 Construire le diagramme de classes

Vous allez d'abord extraire un diagramme de classes automatiquement depuis le code. Vous devez ensuite le simplifier pour ne garder que les informations qui vous semblent utiles par rapport à l'objectif poursuivi (voir ci-dessus).

Tout au long du TP vous devez utiliser le logiciel Umbrello, qui est un outil relativement simple permettant de construire et manipuler des diagrammes de classes (entre autres diagrammes UML).

Configuration de Umbrello. Commencez par indiquer à Umbrello de montrer par défaut les attributs et opérations non publiques : `Settings` → `Configure Umbrello UML Modeller` → `Class` puis décocher `Public only`.

Extraction automatique d'un diagramme de classes. Dans Umbrello, sélectionner `Code` → `Code Importing Wizard`. Dans le menu, sélectionner *tous les fichiers* source de l'application, puis `Next` → `Start Import` → `Finish`.

Les classes sont maintenant importées et visibles dans le panneau de gauche `Tree view`. Glisser déposer toutes les classes vers le grand panneau central `class diagram`.

Modification du diagramme de classes. Vous pouvez maintenant modifier le diagramme. Vous consacrerez à cette tâche entre 1h et 1h30.

Ne vous fiez pas aux associations déjà présentes après l'importation : l'outil Umbrello utilise l'agrégation comme relation par défaut entre les classes, et ce n'est pas celle qui est forcément la plus fréquente. Vous devrez peut-être modifier toutes ces agrégations.

Pour pouvoir simplifier le diagramme de classes, vous devez d'abord comprendre l'architecture de l'application, deviner les choix de conception des auteurs, et identifier la responsabilité de chacune des classes et les concepts les plus importants dans l'application.

Vous aurez sans doute besoin d'utiliser le code source.

4 Confronter sa solution à celle des autres

Une fois que vous êtes confiants dans le résultat obtenu (au maximum 1h30 après le début de la séance de TP), vous allez vous mettre par groupe de 2-4 étudiants ayant travaillé sur le même diagramme de classes

pour **comparer vos solutions et arriver à une solution commune**.

Nous vous suggérons de procéder de la manière suivante :

1. Prendre une capture d'écran (assez grande) de votre solution, et la placer dans votre répertoire public.
2. Changer d'ordinateur pour vous regrouper avec les personnes avec qui vous allez travailler, éventuellement sur le même ordinateur.
3. Comparer vos solutions respectives, discuter des différences, pour arriver à une solution commune.

Une fois que vous vous êtes mis d'accord et construit une solution commune, l'un de vous doit la déposer sur Moodle pour une utilisation ultérieure. Donner à votre fichier un nom qui mentionne le nom de l'application sur laquelle vous avez travaillé.

5 Rédiger des commentaires

On peut supposer maintenant que vous comprenez assez bien l'architecture de l'application sur laquelle vous avez travaillé. Votre prochaine tâche consiste à ajouter des commentaires dans le code source qui pourraient faciliter le travail du prochain étudiant qui devra travailler sur la même application.

Pour ce faire, mettez vous d'accord qui va travailler sur quelle classe, et ajouter des commentaires qui vous semblent utiles.

Vingt minutes avant la fin de la séance vous devez :

1. mettre ensemble toutes les classes source modifiées,
2. compiler, puis exécuter les tests (s'ils existent), ou à défaut exécuter le programme, pour vous assurer que vous n'avez pas modifié le code source par inadvertance,
3. créer une archive avec le code source, et la déposer sur Moodle dans le dépôt prévu à cet effet.