ANNEE 2021/2022

RAPPORT FINAL
PROJET TUTORE

EZUML - LOGICIEL POUR CREER UN DIAGRAMME DE CLASSES

Membres de l'équipe :
ARNOUT Fabrice
RENARD Guillaume
RICHIER Marcus
WEISS Lucas

Professeur tuteur : THOMAS Vincent

REMERCIEMENTS

Nous tenons tout d'abord à remercier monsieur Vincent Thomas, notre tuteur, pour le sujet qu'il a proposé, pour son suivi et ses conseils tout au long de ce projet. Nous avons apprécié la confiance ainsi que la liberté qu'il nous a accordée. Ses conseils utiles et avisés qui nous ont permis d'amener ce projet le plus loin possible, mais également de nous améliorer en tant que développeur.

Merci à M. NARBEY pour ses enseignements en conception UML, mais aussi pour son aide ainsi que ses conseils concernant l'analyse du sujet, ce qui a permis de poser les fondations nécessaires à la bonne réalisation de ce projet.

Pour finir, nous tenons à remercier l'ensemble de l'équipe enseignante de l'IUT de ses deux dernières années pour leurs enseignements.

TABLE DES MATIERES

INTRODUCTION

Présentation de l'équipe

Notre équipe est composée de 4 personnes qui étaient dans la même classe lors du troisième semestre. Le groupe est composé d'ARNOUT Fabrice, RENARD Guillaume, RICHIER Marcus et WEISS Lucas.

Qu'est-ce que les projets tutorés ?

Le projet tutoré en deuxième année DUT Informatique est un projet dans le domaine informatique réalisé en groupe réparti sur toute l'année. Basé sur un sujet, guidé par un professeur tuteur, cela nous permet de nous plonger dans la vie préprofessionnelle en nous permettant de découvrir ce qu'est le travail de groupe, l'implication d'autrui sur notre travail que ce soit sur l'analyse ou la conception, mais également devoir respecter le sujet du commanditaire, ses attentes et livrer un produit final dans un temps imparti.

Présentation du sujet

Parmi les 23 sujets proposés cette année, il nous a été assigné le sujet de M. THOMAS qui est le suivant : « Logiciel pour créer/manipuler un diagramme de classe ».

Pour tous les membres de l'équipe, ce sujet était l'occasion de mettre en pratique les deux années de connaissances acquises au sein de la formation DUT Informatique de l'IUT Nancy-Charlemagne.

Le sujet qui nous a été donné est celui de créer un diagramme de classe à partir de fichier .class que l'utilisateur nous donne. Le but est de donner notre logiciel aux étudiants des années suivantes pour les aider à aborder leurs diagrammes de classe. Tout d'abord, un diagramme de classes est un schéma utilisé pour représenter les classes, les interfaces du système ainsi que leurs relations. Le fait qu'il soit généré de façon automatique signifie simplement que l'application va reconnaître une classe, l'exploiter afin de connaître ses attributs, ses constructeurs et ses méthodes, mais aussi définir ses dépendances, c'est-à-dire, les relations qu'elle a avec les autres classes présentes.

Planning du déroulement du projet

Nous avons articulé notre projet de manière itérative. Notre projet a été séparé en 4 parties. On a commencé par l'étude préalable de notre projet qui a débuté lorsque l'on nous a donné le sujet jusqu'au 9 décembre. Cette partie a permis de nous préparer pour les itérations qui allaient commencer. Nous avons donc une étude sur l'existant, une étude sur les solutions techniques de notre projet.

Durant cette période, nous avons fait des diagrammes pour schématiser les besoins et l'utilisation de notre logiciel. Grâce à ces diagrammes (diagramme de cas d'utilisation, scénarios, diagramme de séquence, diagramme de classe, ...) nous avons pu obtenir l'organisation de nos différentes itérations.

La première itération s'est déroulée du 10 décembre au 10 janvier, durant cette partie nous avions débuté la partie graphique, mais nous avions principalement développé la partie backend de notre application.

La deuxième itération, qui s'est déroulée du 17 janvier au 4 mars, cette partie nous a permis de faire la création automatique d'un diagramme, ainsi que le déplacement de classe.

La dernière itération, qui s'est déroulée du 5 mars au 8 avril, cette partie finale nous a permis d'ajouter de nombreuses fonctionnalités.

I. ANALYSE DU SUJET

➢ A quel besoin doit répondre le produit ?

Le projet a pour but d'être utilisé par les étudiants des années suivantes. Le but est de créer un logiciel qui permet de créer simplement et rapidement un diagramme de classe à partir de fichier .class (java compilé) dont ils pourront déplacer les classes pour réorganiser le diagramme à leur souhait. Les professeurs de l'IUT montraient jusque-là des diagrammes venant de objectAID aux élèves, l'avantage du programme est de laisser les étudiants comprendre par eux même avec leurs propres programmes par exemple.

➢ A quelles contraintes doit répondre le produit ?

De notre besoin, découle des contraintes. En effet, notre logiciel va être utilisé à l'IUT par des étudiants. Déjà comme on compte utiliser l'introspection, notre logiciel doit être en Java ou faire appel à la JVM. Il faut donc que le logiciel fonctionne avec la version de Java installée sur les machines de l'IUT (Java 1.8). On va aussi vouloir porter notre logiciel sur tous le plus de système d'exploitation possible. Finalement, le logiciel doit avoir un temps d'apprentissage inférieur à 10 minutes.

➢ Analyse de l'existant

Les applications de créations automatiques de diagramme de classe ne sont pas nouvelles. Il en existe plusieurs qui permettent plus ou moins la réalisation automatique de diagramme de classes.

Par exemple, il existe ObjectAid. Contrairement à d'autres, comme plantUML où c'est à l'utilisateur de coder le diagramme, ObjectAid prend des classes et les génère dans la zone visuelle.

De plus, un point important à souligner est que dans ObjectAid, nous avons la possibilité de déplacer chaque rectangle, où un rectangle représente une classe. Le déplacement permet de manipuler chaque classe, tandis que l'application déplace en même temps les dépendances, voire l'image 1.

Grâce à ses deux fonctionnalités, nous avons pris la décision de prendre exemple sur ObjectAid pour les bases de notre projet. Ensuite, lorsque nous entrons dans le détail, certaines autres fonctionnalités sont assez intéressantes.

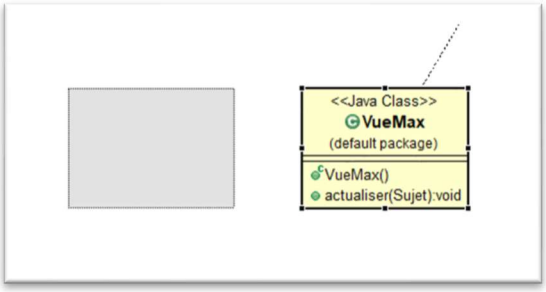


Image 1 : déplacement d'une classe

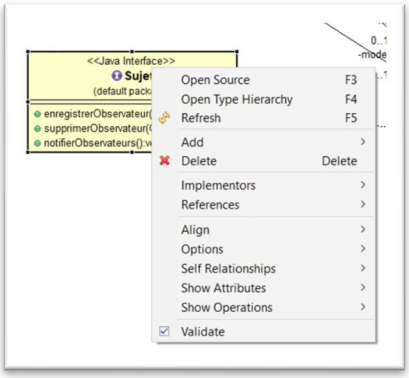


Image 2 : clic gauche sur une classe.

Après quelques manipulations, nous pouvons nous apercevoir que les fonctionnalités intéressantes sont celles lorsque l'on manipule les classes par rapport à leurs contenus, ou bien en affichant les classes existantes dont elles héritent. De plus, la facilité du logiciel fait qu'il est très simple à prendre en main avec le menu simple d'utilisation qu'on peut voir dans l'image 2.

Mais il y a également quelques points négatifs comme le fait que le logiciel ne réalise que de l'affichage. En effet, à part la fonctionnalité de déplacement et celles concernant l'affichage du contenu des classes, on ne peut pas modifier le diagramme de classes, c'est-à-dire qu'on ne peut pas à partir d'un diagramme de classes généré, d'en réaliser un manuellement afin de pouvoir représenter l'itération suivante du projet en cours.

Pour finir avec ObjectAid, on peut noter quelques autres fonctionnalités intéressantes telles que le fait de modifier le code dans une classe le mettra à jour dans le diagramme en temps réel.

Enfin, si nous devons tirer une leçon des échecs d'ObjectAid, c'est principalement parce qu'il est lié à Éclipse, où l'application doit régulièrement être mise à jour pour toujours être le plus fonctionnel possible avec Eclipse qui effectue ses mises à jour. On peut donc en déduire que le mieux serait une application bien distincte qui ne devra pas être mise à jour dans le seul objectif de rester fonctionnel.

D'autres solutions sont également présentes sur le marché, en voici une liste non-exhaustive :

- Le logiciel Bouml : BOUML - une suite UML gratuite. On y retrouve des fonctionnalités comme le glisser-déposer de code pour créer un diagramme modifiable dynamiquement et l'inverse également. Le programme gère différents types de diagramme. Mais le logiciel demande un temps d'apprentissage trop élevé par rapport à ce qui est demandé.

- uml-parser, dont une version mise en place sur un site web qui permet de générer des diagrammes de classes statiques aisément. Malheureusement, le programme est trop limité et les diagrammes statiques ne correspondent pas aux attentes.

- Umodel est un logiciel payant, très complet (donc complexe à maîtriser). Même s'il ne convient pas à ce que l'on cherche pour les deux raisons citées précédemment, ses fonctionnalités pourront nous servir d'inspiration.

- ArgoUML est un logiciel open-source qui peut créer de nombreux diagrammes UML à partir de code source de différents langages et de créer la structure de code java à partir de diagramme. Mais le logiciel reste complexe et ne gère pas les gros projets.

- StarUML est la version "complète" d'ArgoUML, mais qui est plus dure à prendre en main. StarUML n'est plus mise à jour depuis un moment.

Plein d'autres logiciels existent dans ce but, mais avec notre sélection nous seront sûrement capables de mettre en avant les différentes fonctionnalités dont nous pourrions nous inspirer, et des problèmes à ne pas reproduire.

Avec l'étude que nous avons faite sur les applications concurrentes, on peut voir quelques problèmes liés à ces applications, que notre application devra éviter de reproduire. Un exemple serait de ne pas être lié à un IDE, car lors de sa mise à jour, il faudra que notre application marche encore et faire des modifications si nécessaires.

On a pu voir que dans certaines applications de nombreuses fonctionnalités étaient bien, mais que parfois, on pouvait avoir de grosses pertes de performance lors de la création de l'UML, on doit éviter cela. Un autre problème récurrent est un temps d'apprentissage requis trop élevé ou des fonctionnalités manquantes.

Une des fonctionnalités avec laquelle on va pouvoir se démarquer de nos concurrents est la possibilité de "sortir" une classe de java pour l'afficher, non plus comme type d'attribut mais comme classe à part entière.

➢ Les fonctionnalités principales

– *Réaliser l'introspection des classes*

L'introspection est la capacité d'un programme à examiner les propriétés d'un objet. En java, cela nous permet d'avoir accès aux informations de la classe pour les éléments qu'ils soient publics ou privés. Cela nous permettra de savoir comment est composé un fichier et de faire notre classe du diagramme en conséquence.

– *Stocker les informations pour les classes du diagramme*

La manière d'organiser notre architecture des classes s'est révélé être un vrai défi, car nous voulions faire une arborescence des packages chargés. Cela nous a grandement complexifié la tâche, car les deux vues ont besoin de format de stockage différent (liste et arbre).

– *Afficher les différentes classes*

Une fois les informations en place, il a fallu créer une vue qui corresponde à nos attentes, on a utilisé une police monospace pour pouvoir aisément calculer la largeur et la hauteur nécessaire pour le texte et ainsi faire un cadre adéquat. Cette fonctionnalité était critique à notre programme, car elle a permis de voir le résultat de nos efforts passé et a permis de voir le résultat des fonctionnalités suivantes.

– *Afficher l'arborescence*

Lorsque que l'on charge un diagramme, une arborescence de fichier est créée directement à gauche de notre interface graphique. Cette arborescence est faite par rapport aux noms des packages des fichiers déjà chargés.

– *Afficher les flèches (associations, héritage, implémentation)*

L'affichage des flèches est un élément essentiel pour un générateur de diagramme de classe. Les flèches sont séparées en trois types : associations, héritage, implémentation. Nous avons fait cette fonctionnalité en plusieurs étapes cumulative : héritage, implémentation, association, association qui s'auto-pointent, ne se superpose pas, et avec cardinalités.

– *Exportation*

On veut pouvoir exporter nos diagrammes dans différents formats, car les élèves vont vouloir rendre, montrer, réutiliser les diagrammes de classes créer avec notre logiciel pour leurs cours. La priorité a été mise sur l'exportation en image, car c'est à la fois la plus simple à créées et la plus importante, car elle sera probablement la plus utilisée.

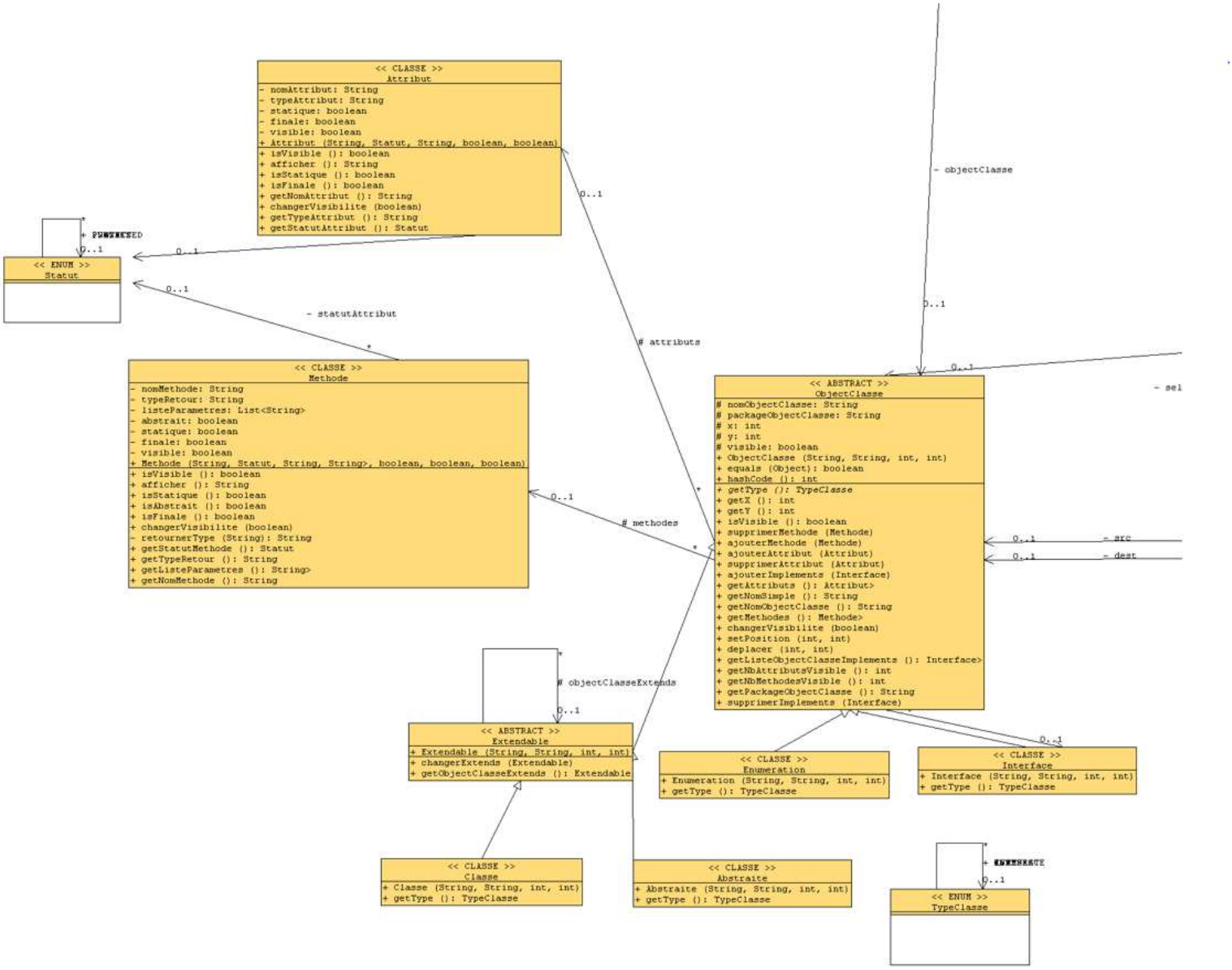
- *Sauvegarde et chargement*

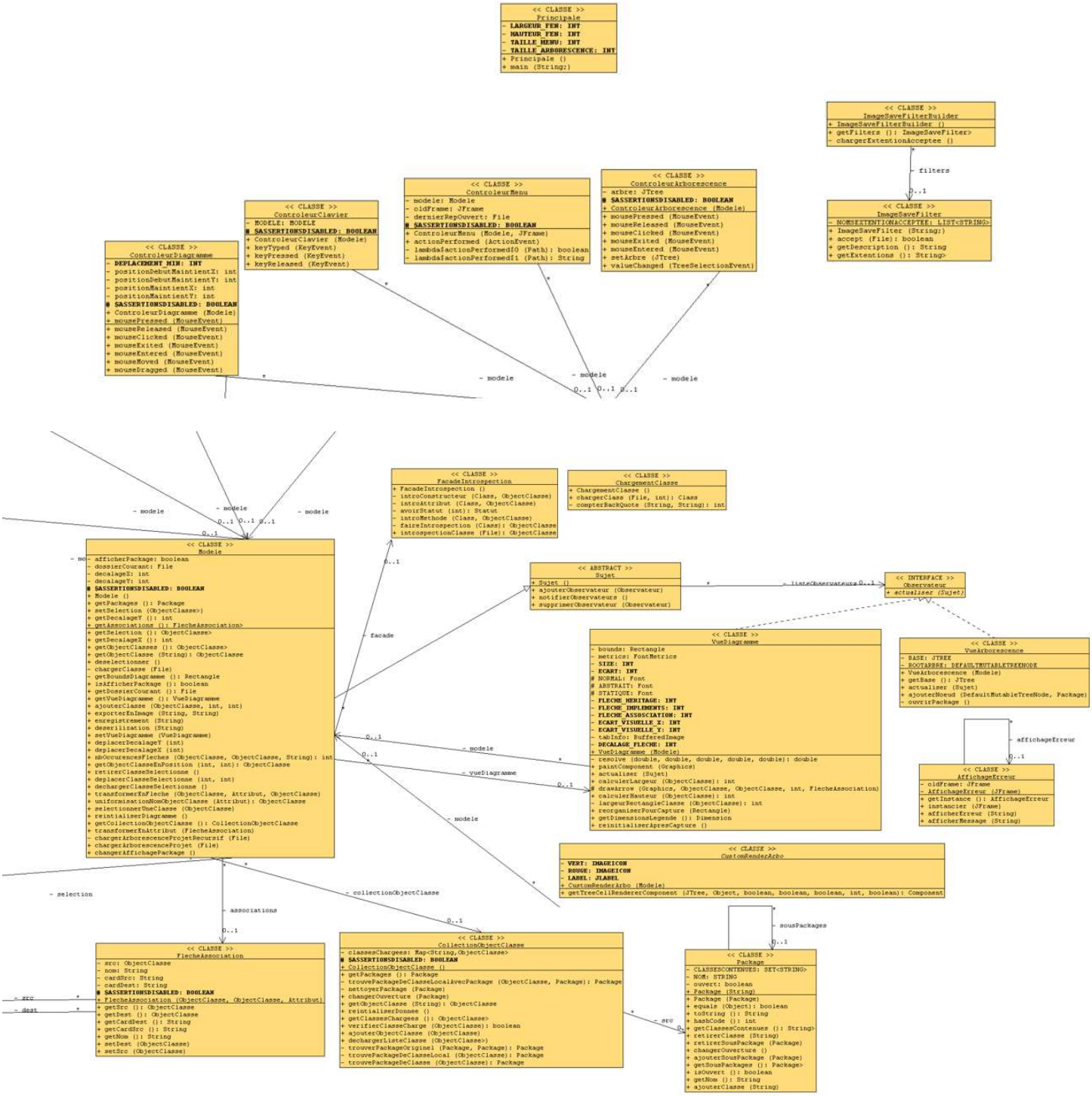
La fonctionnalité de sauvegarde va nous permettre d'enregistrer le diagramme créé, cela sera effectué. Le chargement d'un diagramme permet de retrouver un diagramme anciennement sauvegardé. Cette fonctionnalité est possible, grâce à la sérialisation qui est déjà implémenté dans Java.

Digrammes UML

Le diagramme de classe

Partie de notre diagramme de classe du stockage des données du diagramme :

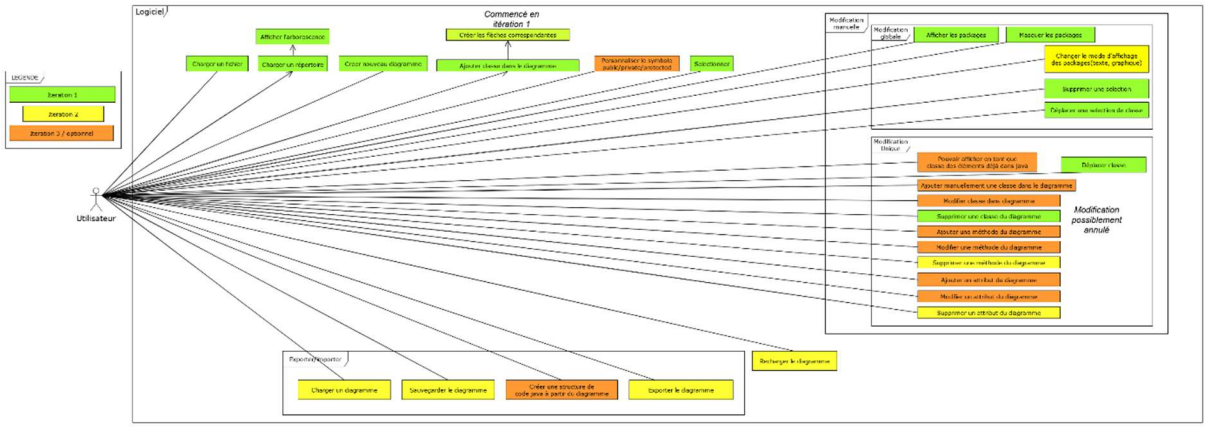




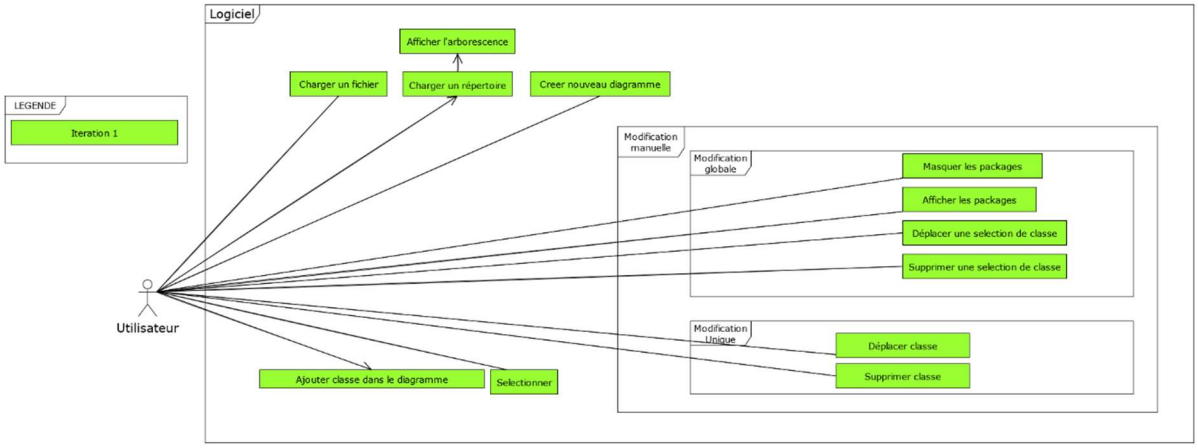
- *Le diagramme de cas d'utilisation*

Le diagramme de cas d'utilisation est le premier diagramme que l'on a fait. Il nous a permis de synthétiser comment l'utilisateur allait utiliser notre logiciel et quelle fonctionnalité majeure allais le composer.

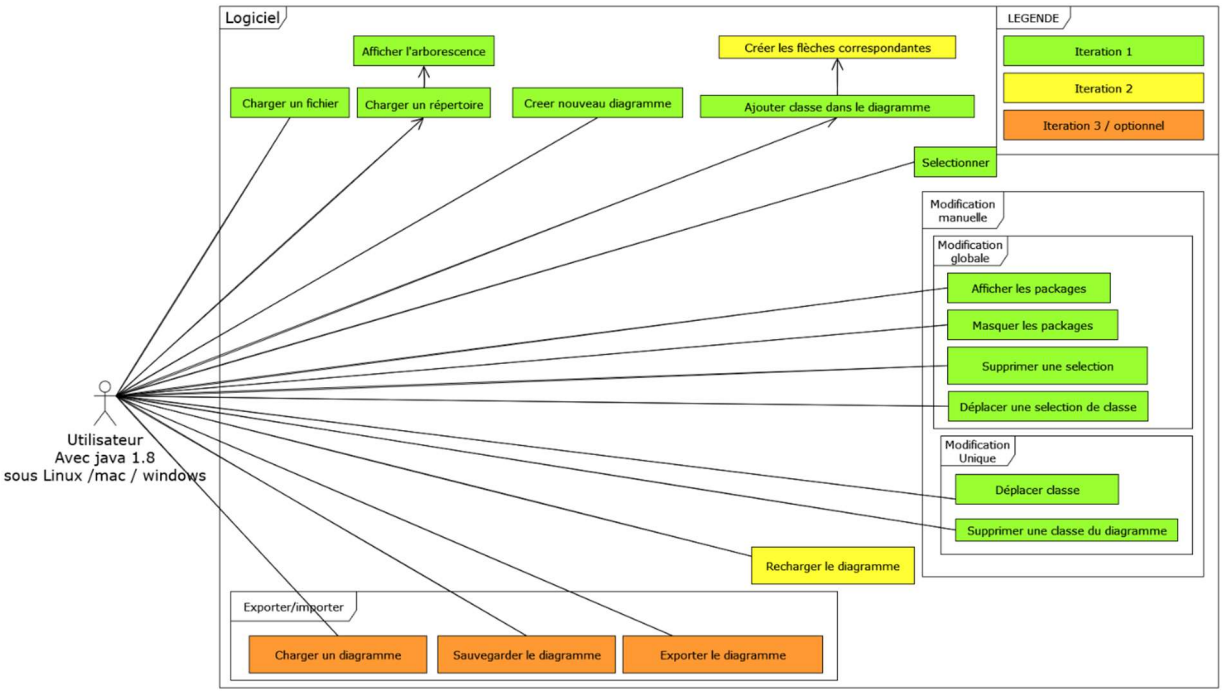
Lors de notre étude préalable, nous avons réalisé deux diagrammes de cas d'utilisation. L'un, complet répertorié l'entièreté des cas d'utilisation prévu dont certains facultatif, car l'itération 3 était difficilement prévisible.



L'autre diagramme, avec uniquement les cas d'utilisation liée à l'itération 1.



Finalement, les fonctionnalités prévues initialement pour l'itération 3 de manière facultative n'ont pas été faites soit par manque de temps à cause de divers retards accumulés, soit parce que les fonctionnalités n'étaient finalement pas indispensables au projet contrairement à d'autres fonctionnalités que l'on a découvert au fur et à mesure des itérations.



On peut rapidement voir que nos espérances ont été revues à la baisse notamment à cause de retards inopinés.

- *Les diagrammes de séquences*

Chargement d'un diagramme

Scénarios :

L'utilisateur ouvre notre application. Il appuie sur le bouton "charger un ".class". Un explorateur de fichier s'ouvre, l'utilisateur peut donc choisir le ou les fichiers qu'il veut charger. Lorsqu'il fait ouvrir ou appuie sur la touche entrée. Notre application met à jour l'arborescence avec le nom du package et celui du/des fichiers.

Diagramme de séquence :

Le diagramme de séquence ci-dessus, représente les méthodes qui permettent le chargement d'une ou plusieurs classes qu'un utilisateur donne à notre application. Dans un premier temps, on fait l'introspection du ou des fichiers donnés, c'est-à-dire, on regarde ce que contiennent le ou les fichiers et on l'enregistre dans la classe objectClasse que l'on a créé. Après cela, on met à jour l'arborescence par rapport au package du ou des fichiers.

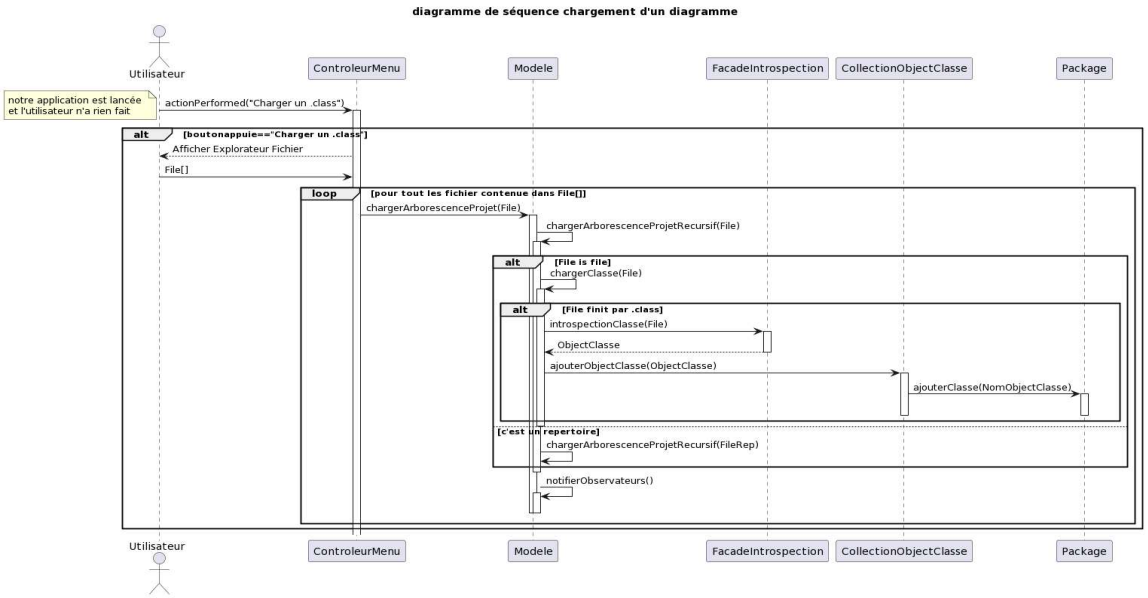


Diagramme de séquence sauvegarde d'un diagramme déjà chargé.

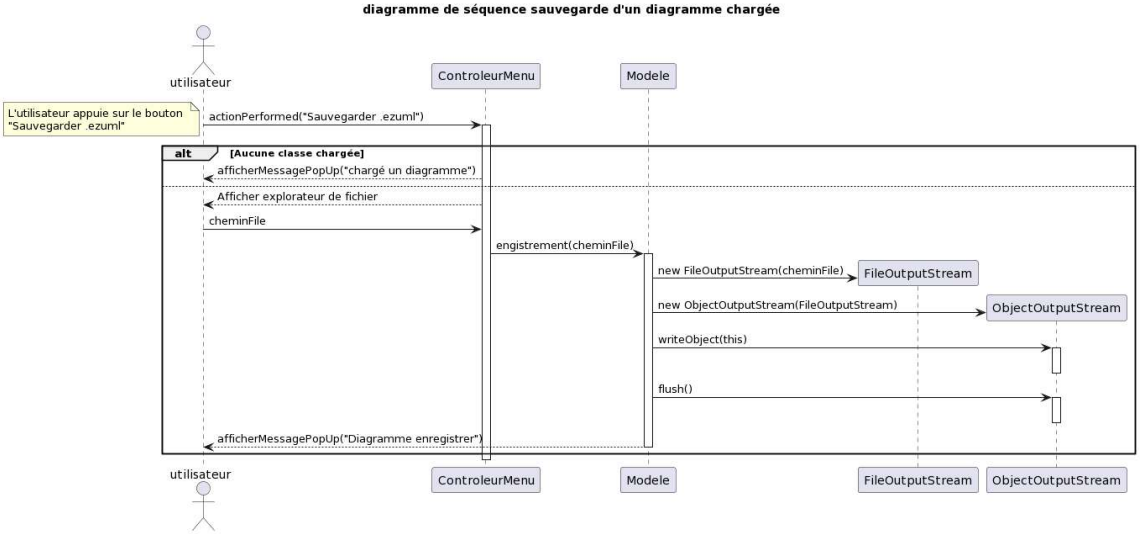
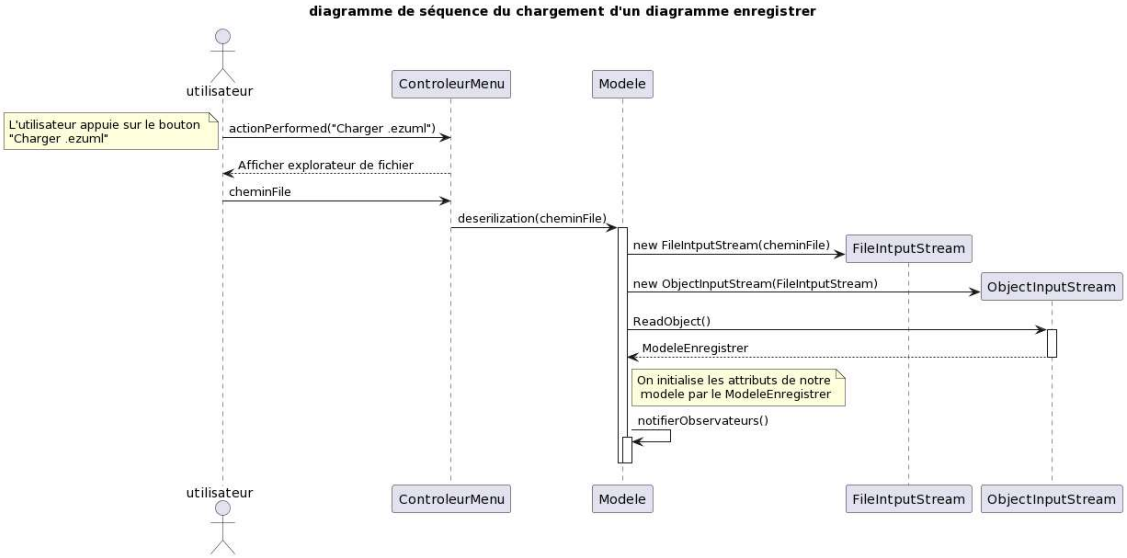


Diagramme de séquence chargement d'un diagramme déjà enregistré.

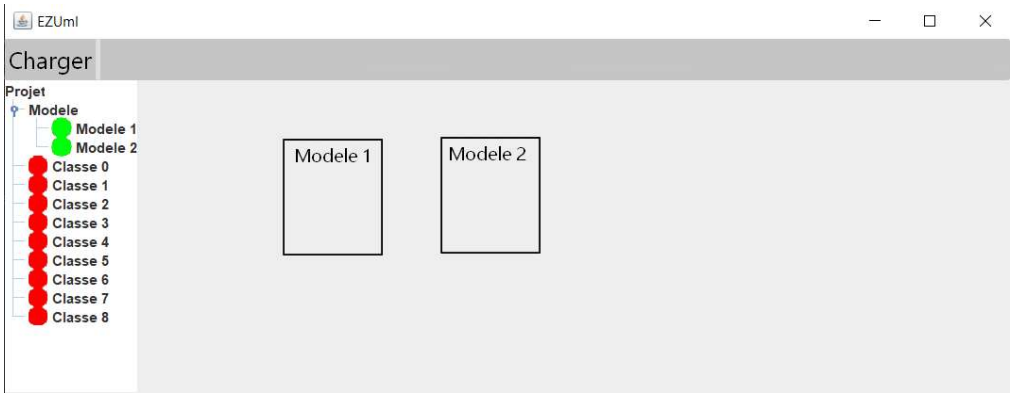


II. REALISATION DU PROJET

➢ Première itération

Durant la première itération qui s'est passé entre le 10 décembre et le 10 janvier, nous avons commencé la partie back-end de la création du diagramme. Ce qui a fait que nous nous sommes trop attardés sur cette partie, au détriment de la partie visuel de notre projet, qui à la fin de l'itération n'avait qu'une partie de l'affichage de l'arborescence, alors qu'au début, il était prévu de faire un diagramme sans flèche à partir d'un fichier .class ou d'un répertoire et d'autre fonctionnalité que nous n'avons pas réussi à faire. Ces fonctionnalités ont été déplacées pour les prochaines itérations.

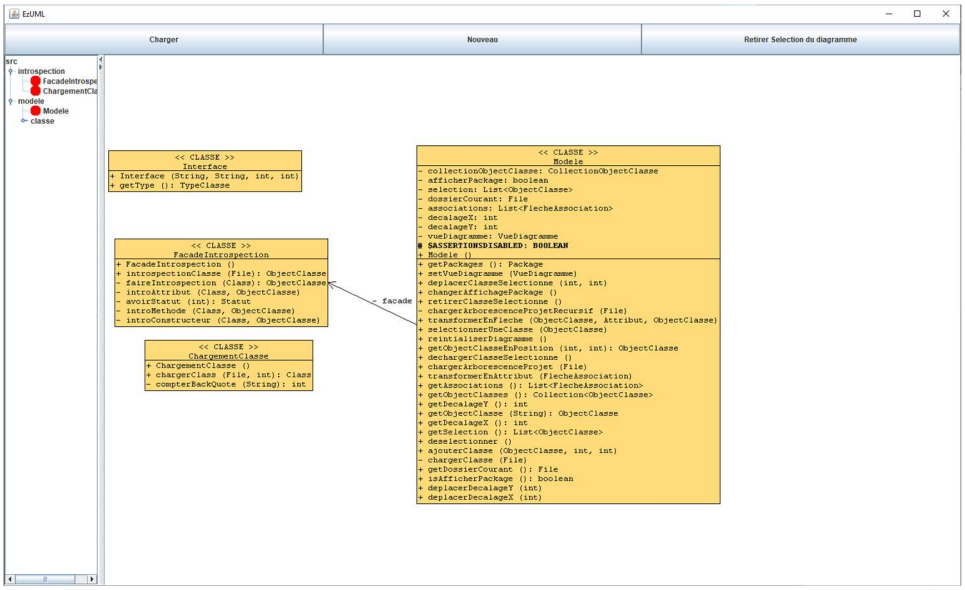
Image de notre application à la fin de l'itération 1 :



➢ Seconde itération

La deuxième itération s'est déroulée du 17 janvier au 4 mars. Durant cette seconde itération, nous avons essayé rattraper le retard que nous avions sur la partie visuelle. A la fin de cette itération, nous pouvions charger un diagramme de classe et le sélectionner pour le déplacer. De plus, l'arborescence se mettait à jour lors du chargement d'un diagramme. Les flèches d'association, d'héritage, d'implémentation ont aussi été faites. Ces fonctionnalités implémentées ont fait que l'on avait une version assez stable de notre application, avec la possibilité de faire un diagramme. Malheureusement, une fonctionnalité prévue n'a pas pu être faite, car nous avions remis en cause notre choix pour la sauvegarde des données, ce qui a affecté la fonctionnalité.

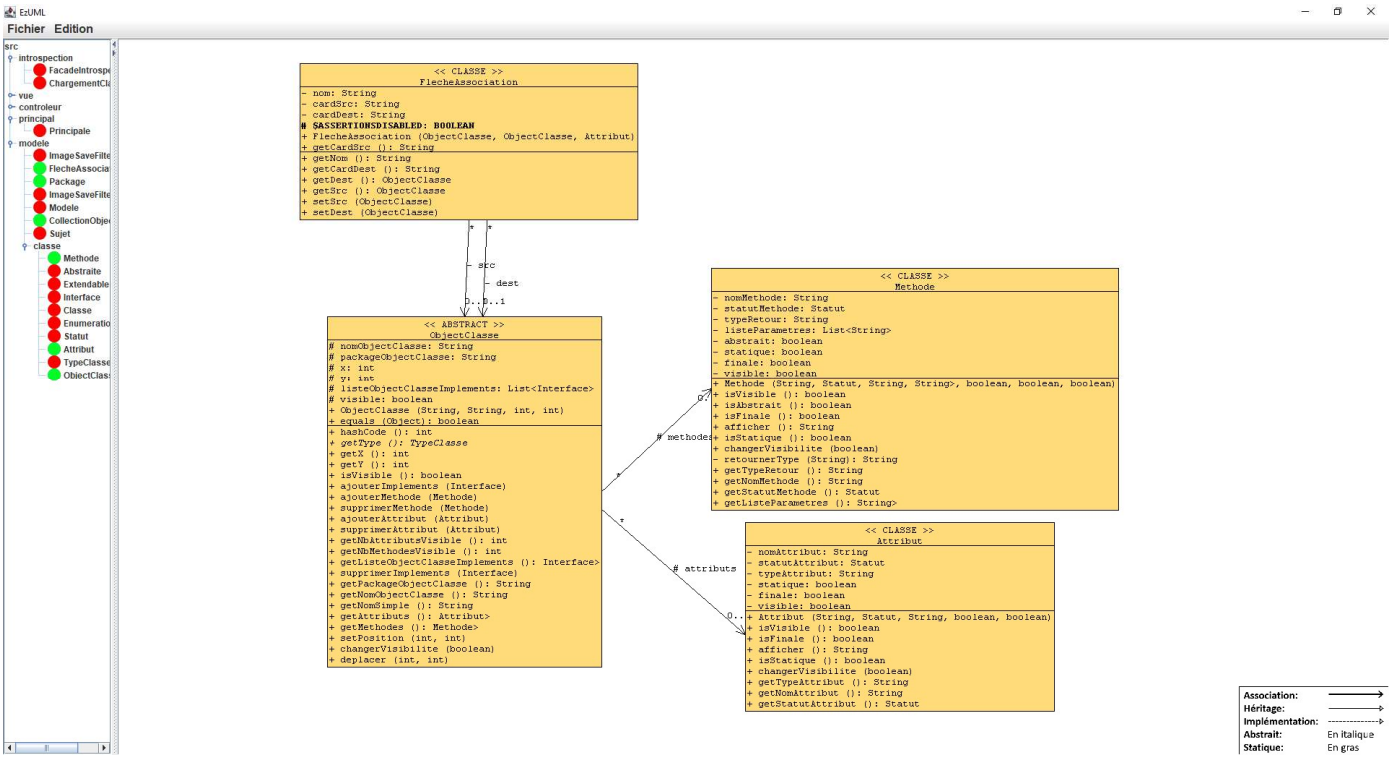
Image de notre application à la fin de l'itération 2 :



➢ Troisième itération

La dernière itération s'est déroulée du 5 mars au 8 avril. Cette dernière itération, nous a permis de terminer les fonctionnalités principales que l'on avait prévues. Nous avons donc fait l'enregistrement d'un diagramme, l'ouverture d'un diagramme sauvegardé, l'exportation du diagramme au format « image », la création des flèches avec des cas particuliers, les cardinalités lors des liens d'association, l'affichage des erreurs avec des pop-ups et l'amélioration graphique de notre barre.

Image de notre application à la fin de l'itération 3 :



CONCLUSION

➢ Un produit fini

En conclusion, nous avons réalisé le sujet de telle manière à ce que les étudiants en première année puissent réaliser des diagrammes simples.

Notre application permet aux utilisateurs de créer rapidement un diagramme de classe à partir de plusieurs fichiers ou répertoire. Notre application comporte aussi de nombreuses fonctionnalités qui sont la sauvegarde d'un diagramme, le rechargement d'un diagramme sauvegardé, l'exportation du diagramme en image.

Notre application est certes entièrement utilisable, mais il reste de nombreuses pistes pour l'améliorer : les diagrammes de séquences, la reconnaissance de patron de conception utilisé, masquer des éléments d'une classe ou des flèche (présent en backend, mais pas encore utilisable). C'est pour cela que nous pensons que notre application pourrait servir de base à un futur projet tutoré. De plus, nous avons mis l'accent principalement sur le backend ce qui permettrais à une équipe suivante de remplacer notre frontend par une technologie plus récente (JavaFX par exemple).

➢ Apports personnels

Ce projet nous a apportés de nombreuses satisfactions, aussi bien à titre personnel qu'à titre collectif. En effet, ce projet a parmi à chacun de connaître d'avantages les éléments qui ont constitué le groupe, et ainsi en apprendre des autres et approfondir nos connaissances dans le domaine de l'informatique. Il s'agit d'une expérience enrichissante où chacun a dû s'adapter aux autres pour mener à bien le travail à réaliser.

Par exemple, le travail a dû être divisé afin de pouvoir tous travailler sur différents axes, ce qui a pu être possible grâces aux outils que l'on a utilisés. Nous avons dû également se mettre d'accord sur nos idées, échanger afin de confronter nos points de vue et d'aboutir sur un choix commun.