# SAE S3.01 - Développement d'applications

Rapport final.

Allart, Noah

Belalia Bendjafar, Amin

De Wasch, Clément

Mangin, Adrien

**Groupe S3B – 10 janvier 2025**

# Sommaire



# Introduction

Le projet SAE 3.01 avait pour objectif principal de développer une application permettant de générer des diagrammes de classes UML à partir de fichiers .class Java en utilisant Java et Java FX. Ce besoin répond à un travail d'analyse réalisé au préalable identifiant les besoins, listant les fonctionnalités à réaliser ainsi que l'organisation du projet.

Notre application permet :

- De représenter graphiquement les classes et leurs relations (héritage, association, etc.) grâce à l'introspection.

- D'interagir avec ces classes à travers une interface graphique intuitive pour les déplacer, masquer des éléments ou en ajouter.

- D'exporter les diagrammes sous différents formats (PNG, PlantUML, squelette Java).

Donc dans ce rapport on va détailler les fonctionnalités réalisées, les problèmes rencontrés ainsi que les évolutions apportées par rapport au rapport d'analyse initial.

# 1. Liste de fonctionnalités réalises.

## 1.1. Itération 1.

**Afficher une classe** :

- Mise en place de la division entre l'arborescence et la zone de dessin.
- Ajout de classes avec des positions initiales fixes codées en dur.

**Exporter au format PNG** :

- Capture d'écran de l'application avec sauvegarde personnalisée.

**Exporter au format squelette Java** :

- Génération d'un squelette Java à partir des classes affichées.

**Exporter en format code PlantUML** :

- Création d'un fichier PlantUML contenant les noms de classes.

**Charger un fichier Java** :

- Chargement dynamique de fichiers via un ClassLoader.

## 1.2. Itération 2.

**Afficher les attributs** :

- Présentation des attributs avec leur modificateur et leur type.

**Afficher les constructeurs et méthodes** :

- Affichage des constructeurs et méthodes avec leurs signatures complètes.
- Mise à jour des formats d'export pour inclure ces éléments.

## 1.3. Itération 3.

**Affichage des dépendances (flèches)** :

- Représentation des relations entre classes sous forme de flèches connectées.

**Cardinalités** :

- Inclusion des cardinalités dans les dépendances.

**Exporter les dépendances** :

- Export des dépendances en squelette Java et en code PlantUML.

### 1.4. Itération 4.

**Masquer / Afficher** :

- Options pour cacher ou afficher les attributs, méthodes ou constructeurs.

**TreeView et Drag and Drop :**

- Implémentation d'un panneau arborescent interactif avec glisser-déposer pour gérer les éléments.

### 1.5. Itération 5.

**Ajout des cardinalités** :

- Gestion des cardinalités spécifiques pour les collections.

**Masquage des classes** :

- Fonctionnalité permettant de masquer graphiquement les classes.

### 1.6. Itération 6.

**Corrections de bugs** :

- Résolution des problèmes d'affichage des flèches de dépendances.

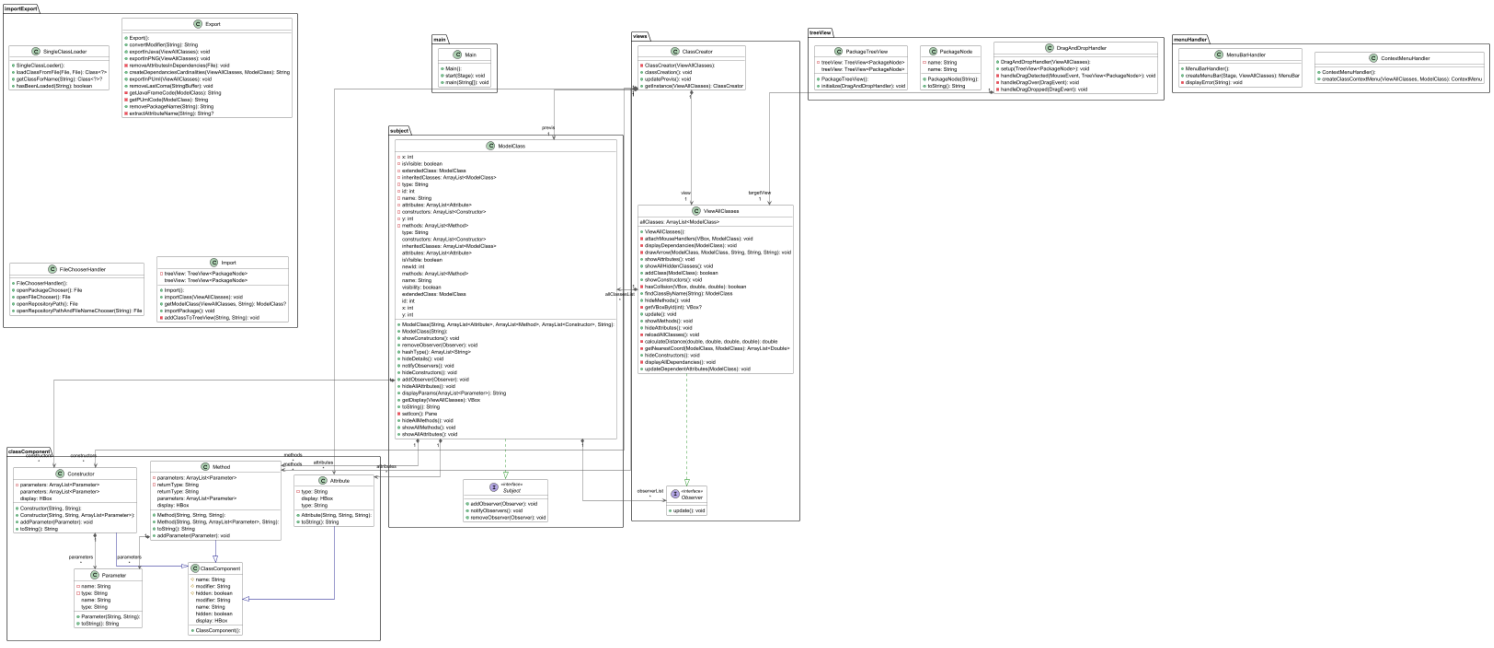
**Création de classe depuis l'application** :

- Ajout d'une interface pour créer des classes directement dans l'application.

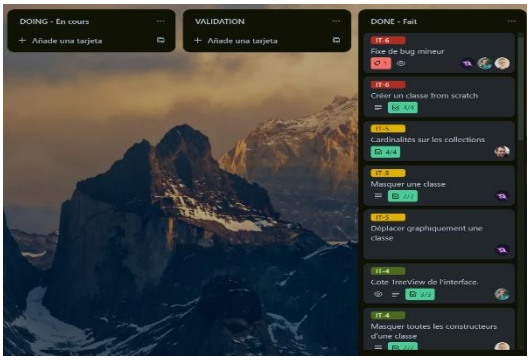
**Export amélioré en PlantUML** :

- Prise en charge des dépendances et des cardinalités dans le code PlantUML.

## 2. Diagramme de classe final.



# 3. Répartition du travail.




Pour ce qui est la répartition du travail, on a utilisé des outils comme Trello qui nous a permis par la suite d'affecter chaque membre du groupe à des fonctionnalités à implémenter, de voir l'avancée du projet ainsi que pour communiquer aux autres membres des taches à réaliser tels que la validation d'autres fonctionnalités...

Ainsi pour ce qui est la répartition du travail on s'est organisé de la manière suivante :

**Noah Allart**

- Affichage graphique d'une classe
- Exporter en PNG
- Affichage des dépendances
- Mise en place du ClassLoader
- Déplacer graphiquement une classe

**Amin Belalia**

- Exporter en PNG.
- Afficher attributs.
- Afficher méthodes.
- TreeView et Drag and Drop.
- Optimisation affichages, bugs et expérience utilisateur.

**Clément De Wasch**

- Exporter en squelette Java.
- Exporter en format code PlantUML.
- Afficher constructeur.
- Exporter les dépendances.
- Masquer/Afficher attributs méthodes et constructeurs d'une classe.
- Création d'une classe de zéro.

**Adrien Mangin**

- Export en code PUML
- Affichage des Attributs

- Affichage des attributs sur les flèches
- Affichage des cardinalités (Collections)
- Mise en place du ClassLoader

# 4. Problèmes rencontrés et adaptations.

**Itération 2 : Problème avec le ClassLoader**

- **Défi :** Le ClassLoader ne parvenait pas à charger les classes dans des packages.
- **Solution :** Tester tous les chemins de dossier possibles pour trouver le début du package.

**Itération 4 : Difficultés avec le Drag-and-Drop**

- **Défi :** Synchronisation entre la gestion des fichiers et la création des modèles graphiques.
- **Solution :** Mise en place d'évènements pour les interactions utilisateur.

**Itération 5 : Gestion des cardinalités**

- **Défi :** Identifier les attributs de type collection dans les dépendances.
- **Solution :** Création d'algorithmes pour analyser et afficher correctement les cardinalités.

**Itération 6 : Gestion des flèches de dépendances**

- **Défi :** Superposition des flèches dans le diagramme.
- **Nous n'avons pas réussi résoudre ce problème mais il est possible de le resoudre avec un itération supplémentaire.**

Ainsi il y a eu quelques problèmes de conception qui a rendu l'implémentation de certaines fonctionnalités compliqué. Les problèmes majeurs de notre conception sont la bonne modularisation de ModelClass donc de MVC.

# 5. Changements par rapport à l'analyse préalable.

Par rapport au rapport d'analyse initial, plusieurs changements ont été effectués pour adapter les étapes du projet aux contraintes rencontrées :

**Priorisation des fonctionnalités essentielles** :

- Certaines fonctionnalités comme la création d'une classe « from scratch » ont été retardées pour être réalisées à la fin.

**Simplification de la conception initiale** :

- Suppression d'éléments jugés secondaires pour se concentrer sur les aspects graphiques et l'export.

**Ajout du Drag-and-Drop** :

- Fonctionnalité non prévue initialement mais devenue essentielle pour améliorer l'expérience utilisateur.

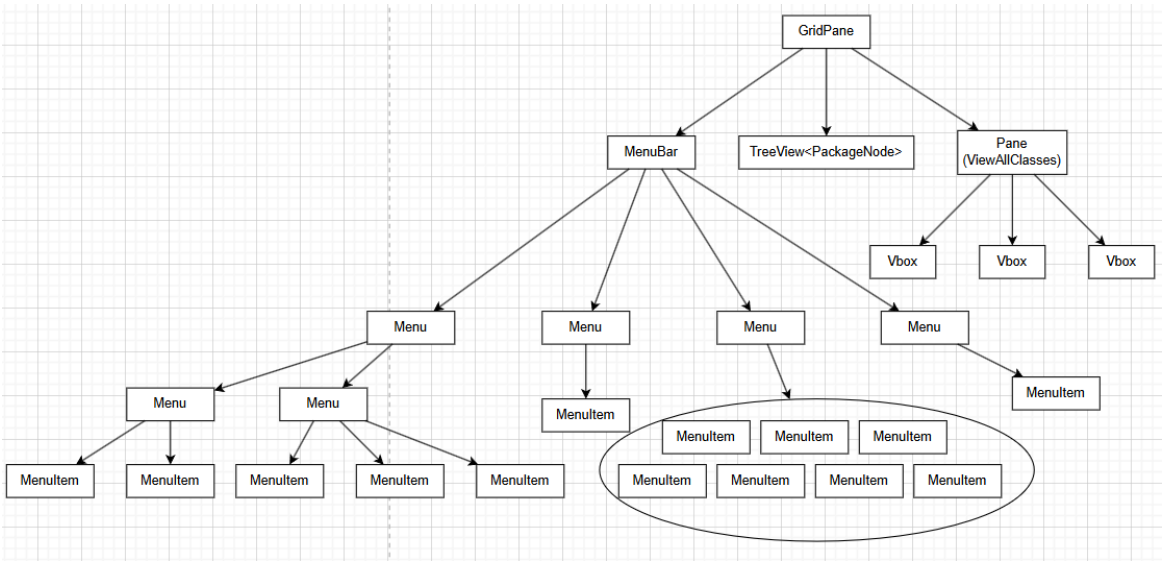
**Gestion des flèches** :

- Les problèmes liés à la superposition ont demandé une refonte partielle de l'algorithme d'affichage.

**Optimisation des exports** :

- Inclusion des dépendances et cardinalités dans les fichiers exportés en PlantUML.

# 6. Graphe de scène et organisation graphique de l'application.

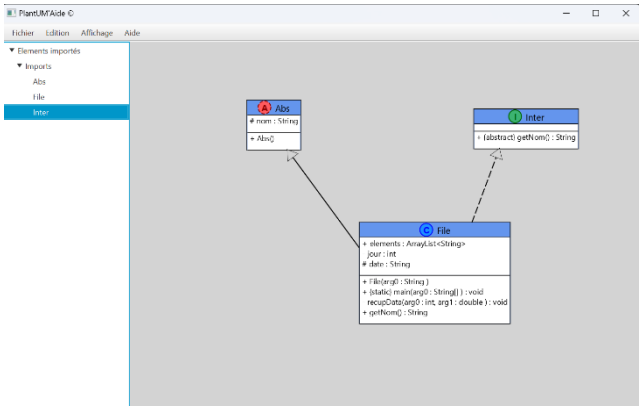



Notre application est constituée de cette manière :

- Un GridPane contenant toutes les zones de notre application
- Une MenuBar représentant la barre en haut de notre application
- Des Menu représentant le groupement des options
- Des MenuItem représentant les boutons pour exécuter les options
- Un TreeView représentant l'arborescence des classes importées
- Un pane représentant la zone graphique du diagramme
- Des VBox représentant les classes du diagramme

# 7. Fonctionnement de l'application.

Cette application a été créé sur Java 21 et Java Fx 5.8.1, pour ce qui est l'utilisation de l'application, on a l'interface suivante :




Dans fichier vous pourrez d'abord importer les fichiers java un par un, celui-ci s'importera directement sur la zone graphique, si vous désirez importer un package, tous les fichiers .class de ce package seront dans la partie gauche, à la suite de cela vous pourrez glisser les classes ver l'interface graphique.

De plus vous pourrez dans le menu Edition vous pourrez trouver le bouton créer une classe qui vous permettra donc de créer une classe de zéro avec les attributs, constructeurs et méthodes désires.

Pour ce qui est la zone graphique en faisant un clic droit vous aurez plusieurs fonctionnalités tels que masquer la classe, masquer les attributs, méthodes et constructeurs ainsi que réinitialiser l'affichage de celui-ci.

Finalement dans le menu Affichage il est possible de masquer toutes les méthodes, attributs et constructeurs ainsi qu'afficher les classes masquées.

## Conclusion.

Ce premier projet complexe nous a permis de comprendre la nécessité de la phase de conception d'un projet. Bien qu'imparfaite celle-ci nous a permis de mener à bien le projet en implémentant la plupart des fonctionnalités souhaitées même si nous avons eu quelques difficultés. Il y a effectivement en pratique, une différence entre la conception initiale et la mise en place concrète, l'expérience devrait à terme nous permettre de minimiser ce problème.