Document Architecture Logicielle
Projet UML Reverse

21 mars 2016

Version :	0.1
Date :	21 mars 2016
Rédigé par :	Yohann Henry Nabil Belkhous Stephen Cauchois Anthony Godin Florian Inchingolo Saad Mrabet Nicolas Meniel

Mises à jour

Version	**Date**	**Modifications réalisées**
0.2	22/01/2016	Architecture refaite dans l'intégralité
0.1	14/01/2016	Première version

Table des matières

1 Objectif

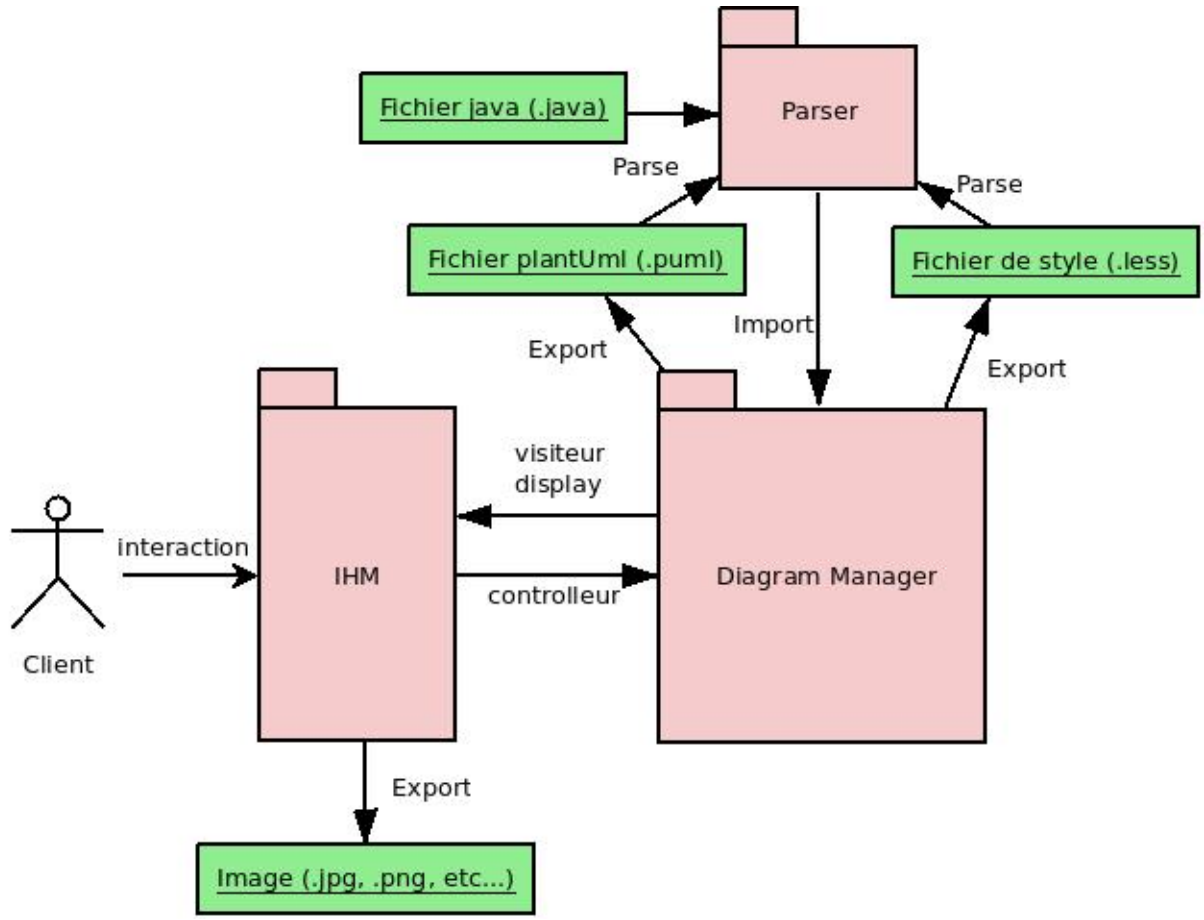
Ce document représente la structure générale du logiciel et les modèles de conception mis en oeuvre pour le réaliser. Il est destiné aux membres de l'équipe de développment, notamment aux concepteurs, ainsi qu'aux superviseurs du projet.

2 Les technologies utilisées

Nous allons utiliser différentes technologies pour la construction du projet.
Le projet est développé en java 1.8. Le projet utilise :

- *Dot* : Un outil permettant de calculer la position idéale des entités et des relations d'un graphe. Cela nous évite la partie mathématique pour le placement des relations qui aurait pu s'avérer un risque majeur.
- *JUnit* : Framework pour valider chaque classe par le biais de tests unitaires.
- *Maven* : Un outil pour la gestion des dépendances de l'application.
- *Antlr* : Un parser dans lequel nous pourrons définir les grammaires pour l'extraction des données d'un fichier java ou plantUml.
- *openJFX* : Une bibliothèque libre graphique parfaite pour ce projet. La bibliothèque permet d'associer à une entité du CSS, ce qui simplifie fortement notre travail.
- *SceneBuilder* : Logiciel permettant de créer des vus de façon ergonomique en drag and drop. Les vus sont créées en fxml.

3 Fonctionnement général



L'architecture est construite suivant le modèle MVC. Cela nous permet de séparer les responsabilités des classes.

4 Organisation des paquetages

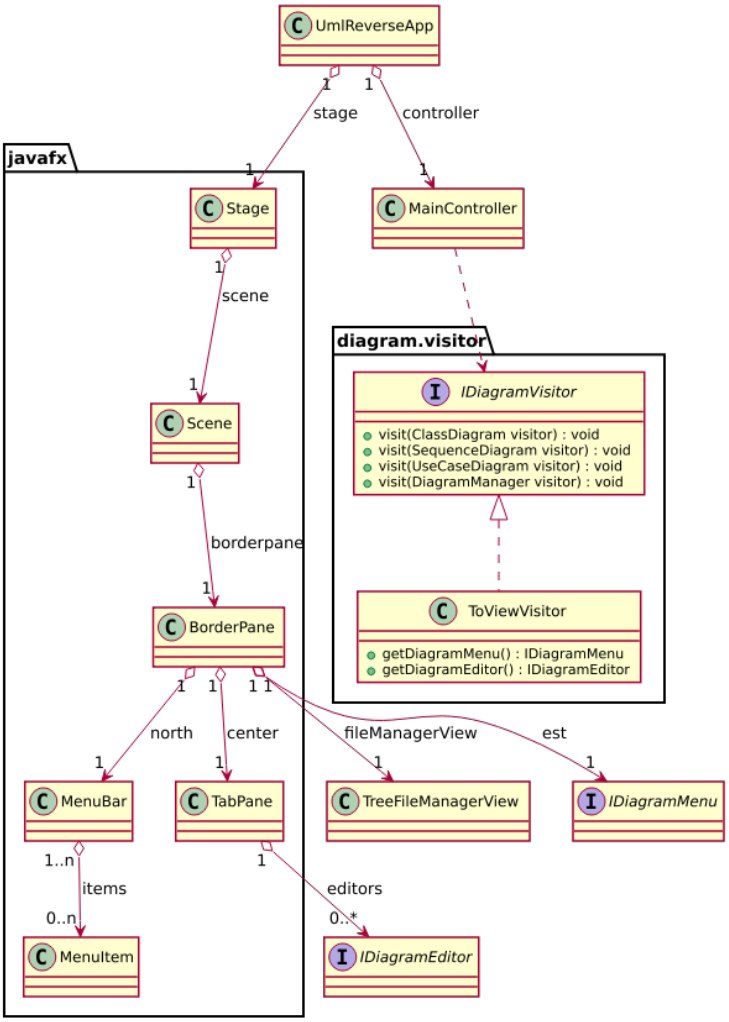
- src
  - main — Contient la classe d'entrée du logiciel
    - UmlReverseApp.java
  - model
    - diagram — Modèlise un diagramme en objet
      - clazz
        - visitor — Permet le parcourt du diagramme de classe
      - usecase
        - visitor — Permet le parcourt du diagramme de cas
      - visitor — Visiteurs globaux. Choisi d'appeller un visiteur en particulier selon le type de diagramme
    - io
      - antlr
  - ui
    - view — Contient les vues principales de l'application avec leurs contrôleurs
    - component — Contient les composants utilisés dans la vue
      - clazz
      - usecase
      - common

5 Paquetage model

Ce package contient l'intégralité du modèle. Voir le diagramme de classe en annexe (diagramClassModel).

6 Paquetage ui

L'application s'appuie sur la structure JavaFX avec la classe applicative UmlReverseApp qui est composée d'un stage qui en suivant la hiérarchie JavaFX nous ammène au BorderPane. Le BorderPane nous servira de base pour les différents éléments de notre application, tel qu'une TreeFileManagerView, une MenuBar, un IDiagramEditor et un IDiagramMenu.

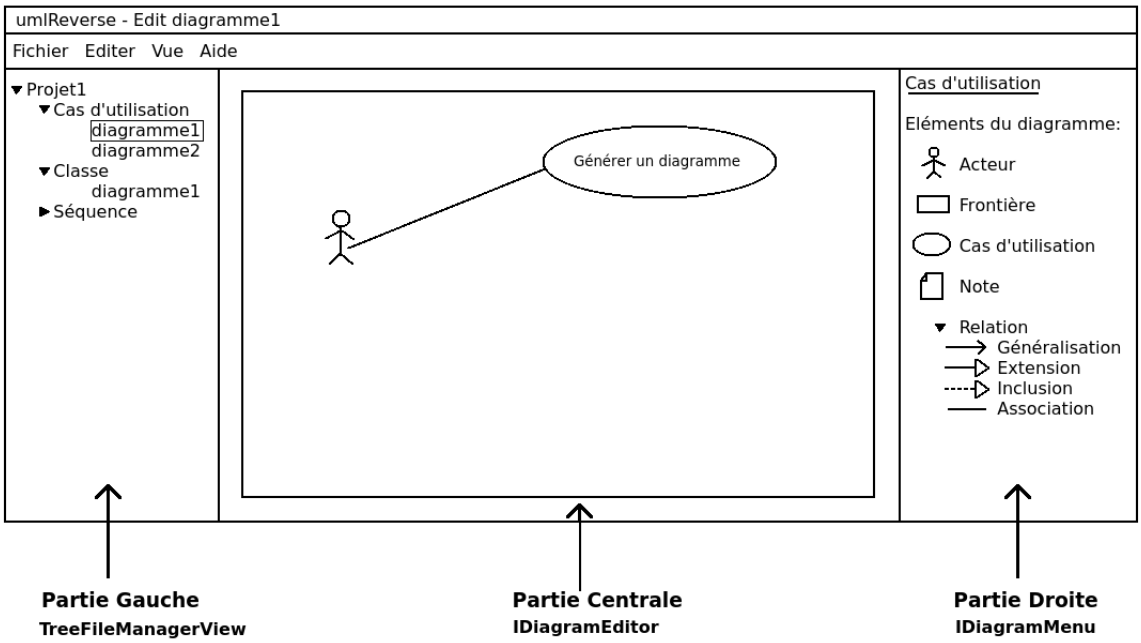


Ce paquetage contient toutes les classes utilisées pour gérer la vue. Ces classes sont toujours associées à un contrôleur. Les vues sont codées en fxml grâce à un logiciel de construction de fxml (SceneBuilder). Chaque contrôleur aura le même nom que la vue associée avec le mot "Controller" ajouté à la fin. ui contient 2 paquetages :

— view : Contient les différentes vues intégrées dans l'application. Le paquetage contient également tous les contrôleurs associés à leur vue.

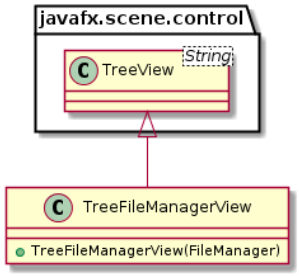
— component : Contient toutes les classes utilisées pour construire les vues. Ce sont leurs composants.

L'application graphique est l'association de plusieurs vues dans un BorderPane.



6.1 La partie gauche

La partie gauche présente la liste des projets disponibles et des diagrammes qu'ils contiennent, et permet de naviguer entre eux. Cette tâche est accomplie par un modèle qui gère les fichiers, accompagné d'une vue hiérarchique constructible à partir de ce modèle.



6.2 La partie centrale

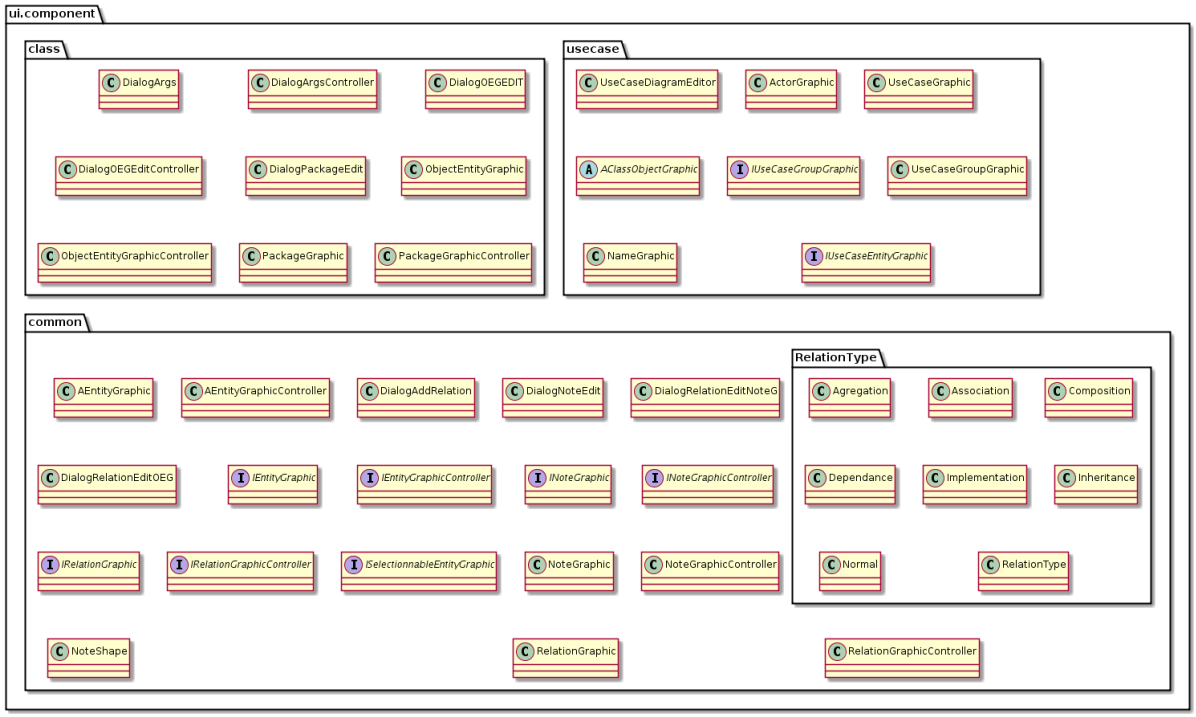
La partie centrale sert à éditer graphiquement un diagramme. La vue principale de cette partie est codée en javafx avec des classes tel que ObjectEntityGraphic.java qui dessine une classe. Et associé à son contrôleur ObjectEntityGraphicController.java par exemple.

Explication : Toutes les classes qui finissent par Graphic sont des réprésentations graphiques des élements du modèle. Elles contiennent des écouteurs de souris sur elles mêmes pour pouvoir permettre aux utilisateurs de les modifier ce qui modifiera le modèle directement. La modification du modèle modifie obligatoirement la vue du diagramme (les éléments graphiques donc) grâce à des écouteurs sur le modèle.

6.2.1 Editeur de diagramme

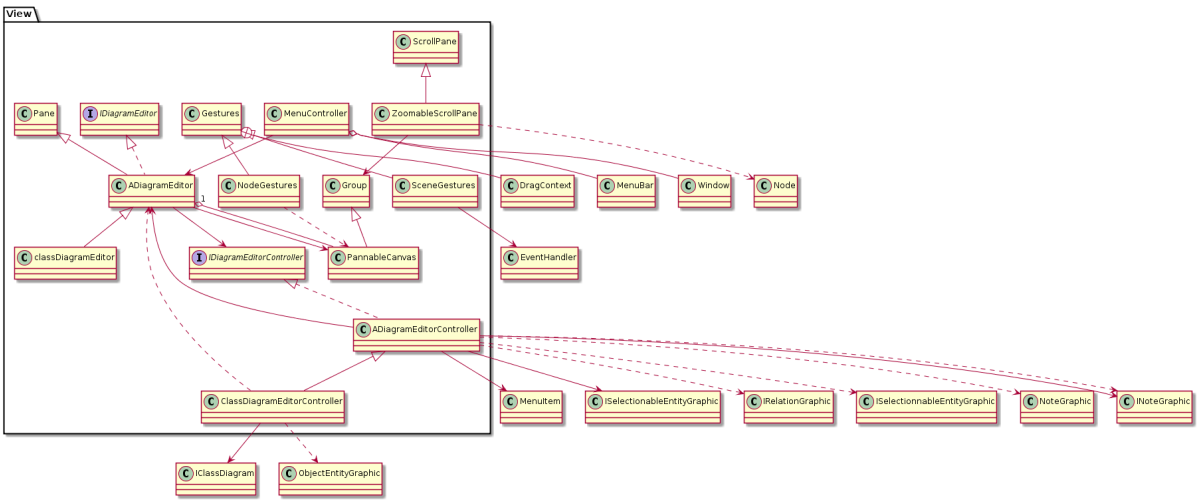
La partie MVC a été volontairement omise pour éviter de surcharger le diagramme de classe. Elle sera par contre implémenté dans les parties prévues.
Dans les diagrammes ci contre les contrôleurs controleront les actions de IDiagramEditor.
Les editeurs de seront suffixé d'Edit, il y en aura pour chaque entié que ce soit un ObjectEntityGraphic ou une Relation. Des classes pour éditer seront disponible elles seront préfixé par Dialog et suffixé par Edit.

6.2.2 Paquetages



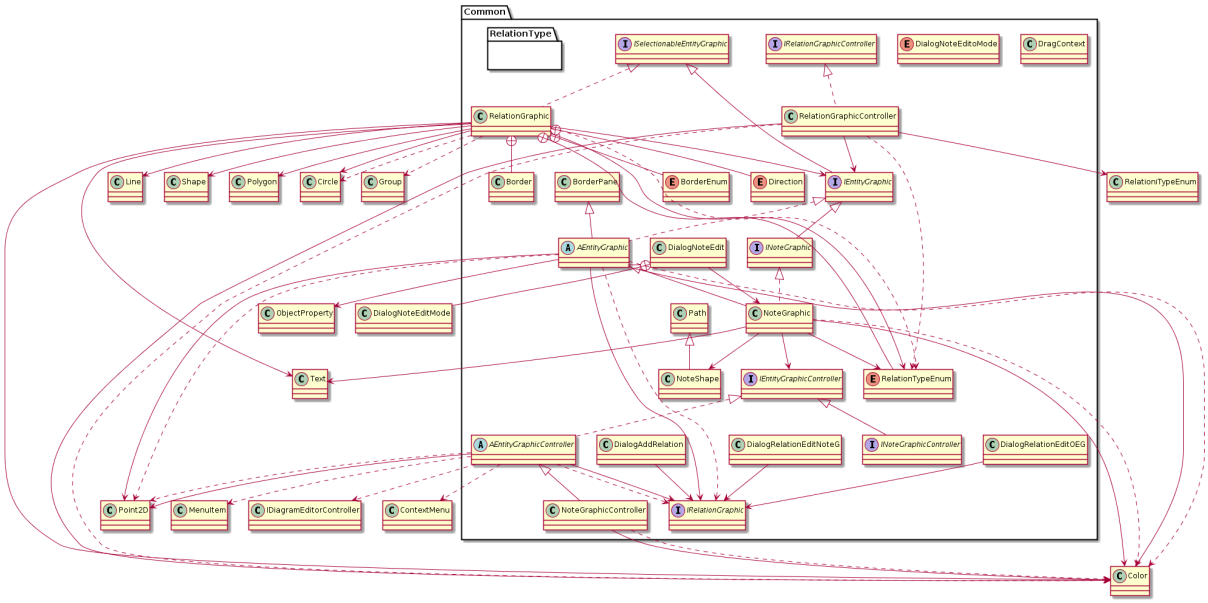
Voici toutes les entités que nous qui sont présentes dans nos packages. Elles sont ici représenté sans relation.

Package view



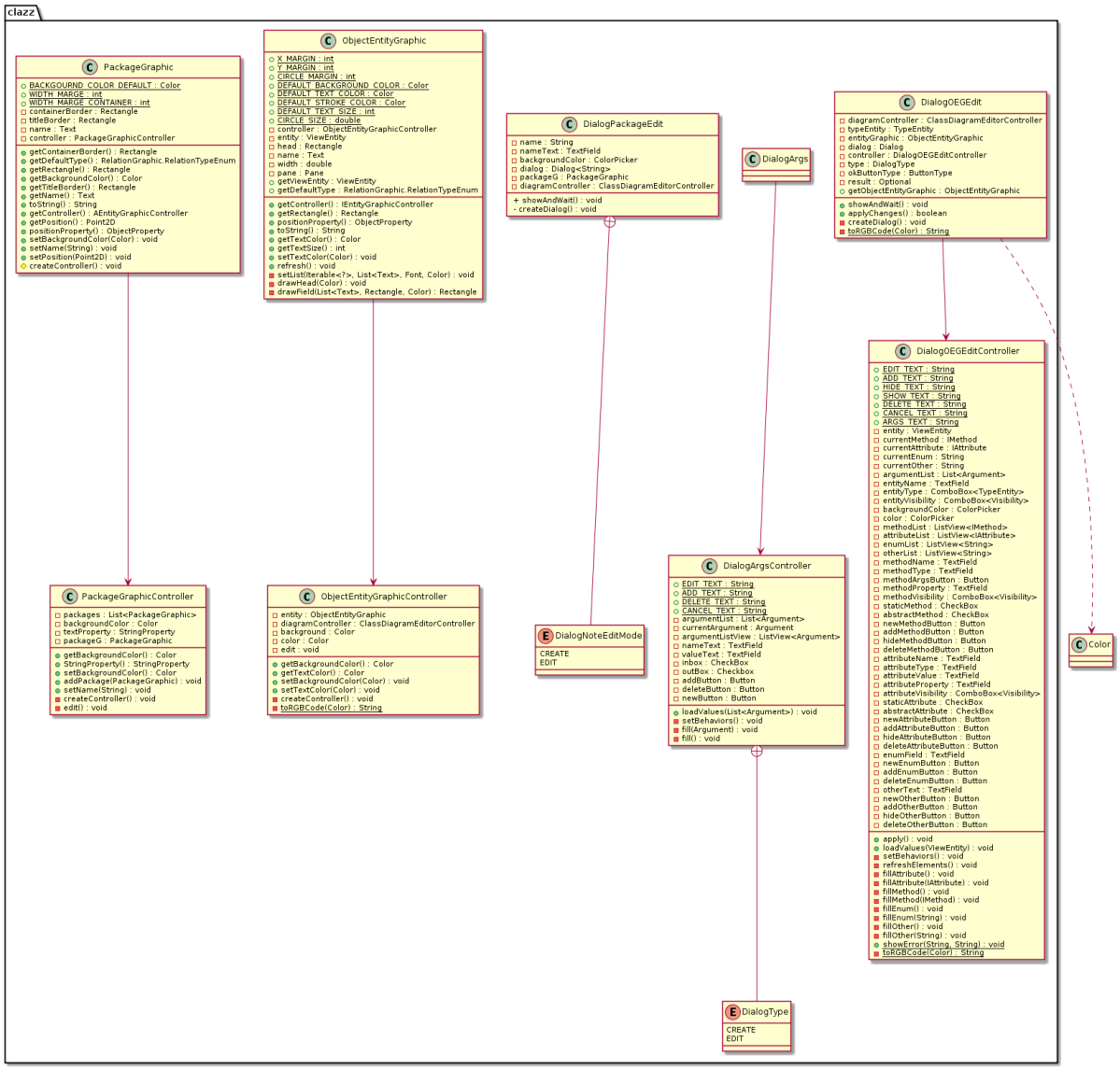
Dans ce diagramme on peut voir le package view représentant la base de notre application graphic, avec les différentes classes permettant le zoom et le drag and drop.

Package common



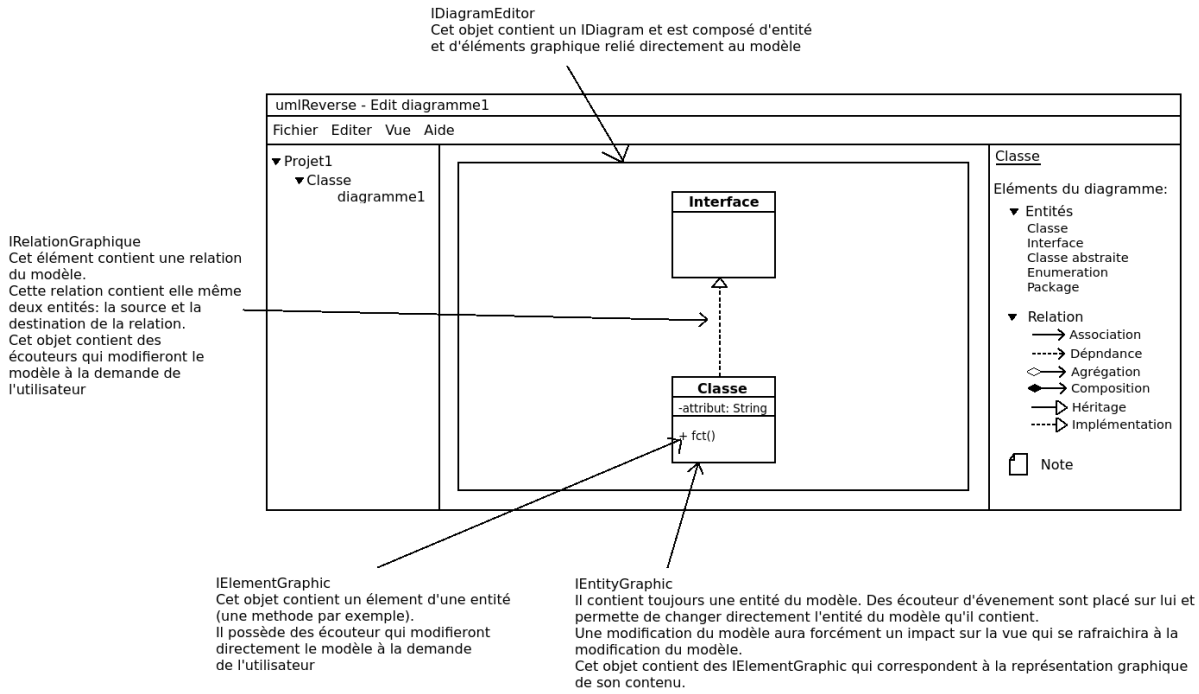
Dans ce diagramme on peut voir la représentation des entités permettant la création et l'édition de note ou de relation. La gestion du menu contextuel est aussi présente dans ce diagramme.

package clazz



Dans ce diagramme il y a toutes les entités spécifique aux diagrammes de classes

6.2.3 Explications

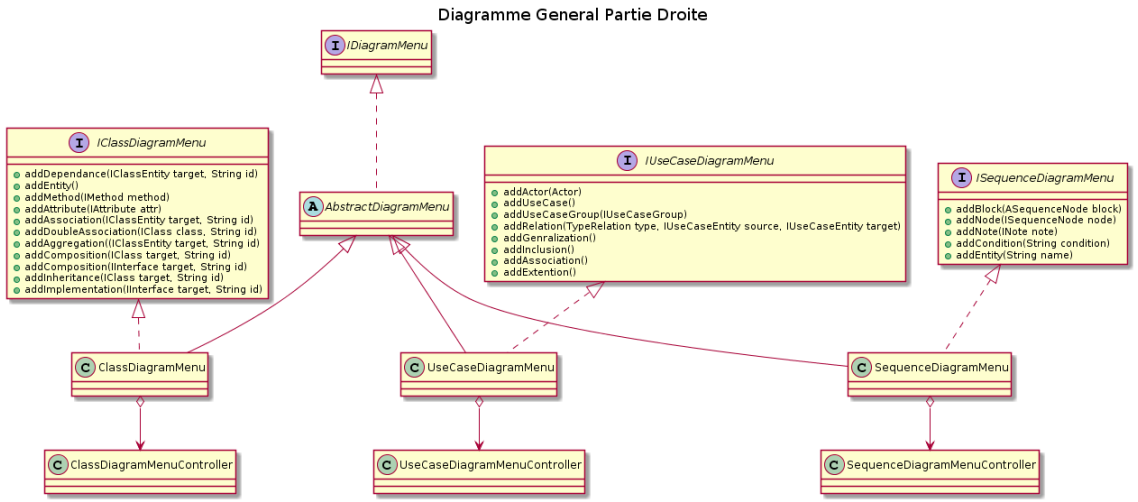


6.3 La partie droite

6.3.1 Déroulement

Quand un utilisateur séléctionne un diagramme l'action fait appel au contrôleur qui va passer par l'interface IDiagram du modèle et le visiteur permet de déterminer le type de diagramme selectionné. L'interface IDiagramMenu nous permet d'ajouter des entités et des relations dans un diagramme grâce au contrôleur qui gère les actions.

6.3.2 Diagramme de classe

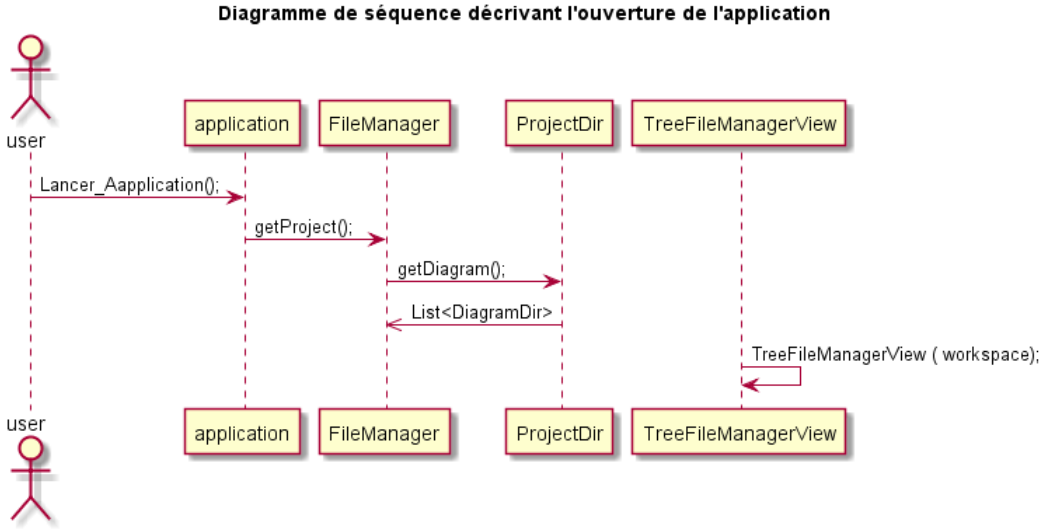


7 Paquetage main

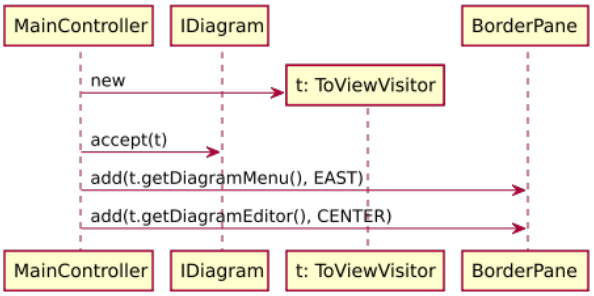
Contient la classe qui lance l'application. C'est le point d'entrée. Il charge les différentes vues du paquetage ui.view en les rassemblant toutes dans un BorderPanel.

8 Diagramme de Séquence

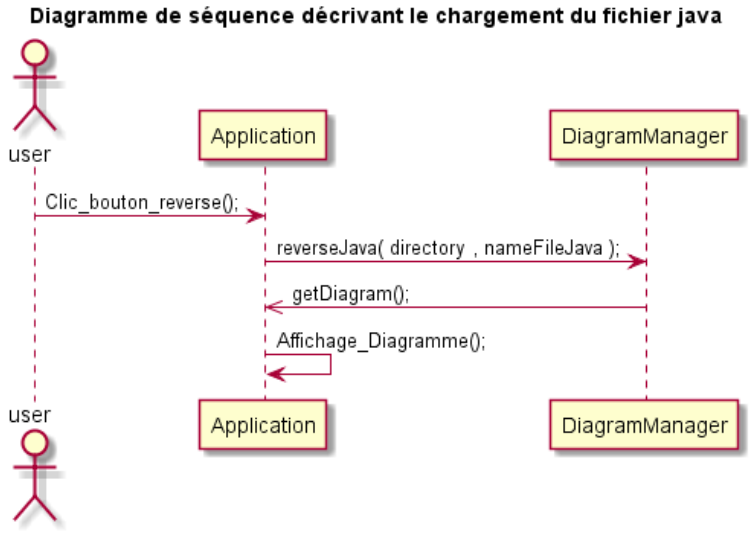
8.1 Ouverture de l'application



8.2 Chargement d'un diagramme dans la vue



8.3 Chargement d'un fichier java



9 Extensibilité

L'architecture a été pensé de façon à vérifier l'exigence EXF_70 (code modulaire, ajout de nouvelles fonctionnalités possible dans le code).

Modèle Le modèle a été pensé à facilement ajouter de nouveau type de diagramme. Pour ce faire le gestionnaire de diagramme ne se soucis pas du type de diagramme. La plupart des actions demandées sur un diagramme de manière extérieur au modèle se font par le biais des visiteurs. Pour ajouter un nouveau type de diagramme il suffit de créer une nouvelle classe qui implémente IDiagram et de mettre à jour les visiteurs globaux.

Partie gauche de l'application Les différentes classes créées peuvent être héritées afin d'étendre leurs fonctionnalitées ainsi que d'ajouter de nouveaux types de diagramme. L'architecture des dossiers d'un projet a été pensée afin que l'ajout de nouveau type de fichier pour la sauvegarde des diagrammes soit simple sans rendre les projets de cette version obsolète.

IDiagramEditor Tout d'abord, l'ajout de nouveau type de diagramme a été pensé de façon à être possible et facile à implémenter. Pour se faire, il suffit d'ajouter dans le IDiagramEditor une nouvelle classe XXXDiagramEditor. Cette nouvelle classe peut hériter ADiagramEditor si le nouveau type de diagramme peut posséder des notes. L'implémentation de celle ci sera déjà implémentée. Les classes présentes dans le paquetage view.component.common sont des classes qui peuvent être utilisées dans n'importe qu'elle type de diagramme. Ce sont généralement des classes abstraites qui font une partie du travail. Ce qui évite de tous refaire à chaque fois.

IDiagramMenu Cette partie peut aussi facilement ajouter des nouveaux types de diagrammes. Il suffit de rajouter un nouveau type de menu et d'implémenter la bonne interface.