SAE S3.01 - Développement d’une application

Partie analyse

ALLART Noah

BELALIA BENDJAFAR Amin

DE WASCH Clément

MANGIN Adrien

Groupe S3B

Sommaire

1. Liste des fonctionnalités

1) Charger un package afin d'afficher les fichiers contenus dans le package, pour ensuite pouvoir glisser les différents classes/interfaces… dans la zone.
2) Afficher un fichier (une classe) Java en format diagramme de classe :

- Afficher le nom, le type de la classe, ses attributs / méthodes / constructeurs (avec leur type d'accès (sous forme de petit logo / souligné / italique), type de retour, type des paramètres).

3) Afficher un fichier (une classe) Java en choisissant le chemin d'accès du fichier
4) Afficher un fichier (une classe) Java en glissant le fichier directement sur l'application
5) Afficher les dépendances entre les classes (liens d'héritages, implémentations, etc).

Interagir avec les "boîtes" :

6) Pouvoir afficher ou masquer les informations (par exemple ne pas afficher les méthodes)
7) Pour modifier la boîte (ajouter / supprimer / modifier un attribut ou une méthode et ses particularités)
8) Pouvoir déplacer le boîtes partout dans la zone
9) Pouvoir modifier la longueur et la largeur des boîtes
10) L'affichage d'une boîte doit être responsive

Menu divers :

11) Créer une classe from scratch (Edition)
12) Format système d'exploitation (Fichier Édition etc…) + bouton d'export
13) Ajout des objets graphiques PlantUML via un menu de création (menu fixe dans l'application ou via un clic droit)
14) Afficher un menu qui représente l'arborescence d'un dossier pour pouvoir y glisser les fichiers (classes) directement sur l'application
15) Ajouter des raccourcis clavier (CTRL+C, CTRL+V, SUPPR, …)
16) Pouvoir exporter le tout en image, code plantUML, résultat d'une compilation PlantUML
17) Actualiser (rafraîchir) l'arborescence

2. Liste des cas d'utilisation de l'application et les diagrammes de cas d'utilisation

- Charger un fichier
- Charger un package

- Exporter en format code plantUML
- Exporter en format image plantUML
- Exporter en format squelette Java
- Exporter en format PNG

- Créer une classe from scratch

- Afficher classe
- Afficher méthodes
- Afficher attributs

- Afficher flèche
- Afficher cardinalités
- Afficher attribut sur la flèche

- Masquer méthodes pour une classe
- Masquer méthodes pour toutes les classes
- Masquer attributs pour une classe
- Masquer attributs pour toutes les classes

- Ajouter méthode
- Ajouter constructeur
- Ajouter attribut
- Ajouter classe
- Ajouter dépendance
- Ajouter zone package

- Modifier méthode
- Modifier constructeur
- Modifier attribut
- Modifier classe
- Modifier dépendance

- Supprimer méthode
- Supprimer constructeur
- Supprimer attribut
- Supprimer classe
- Supprimer dépendance
- Supprimer zone package

Diagramme de cas d'utilisation

Ce diagramme rassemble les différents systèmes de cas d'utilisations liées à la gestion du chargement et exports des fichiers.

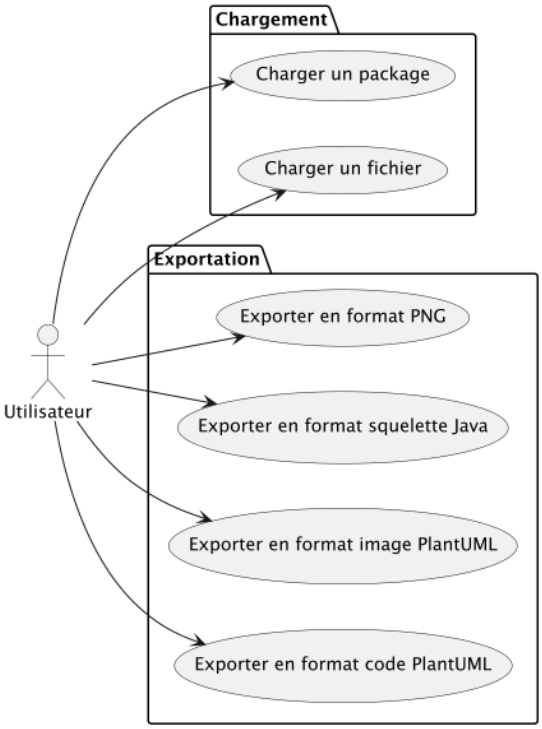


Diagramme de cas d'utilisation

Ce diagramme rassemble les différents systèmes de cas d'utilisations liées à la gestion des éléments graphiques.

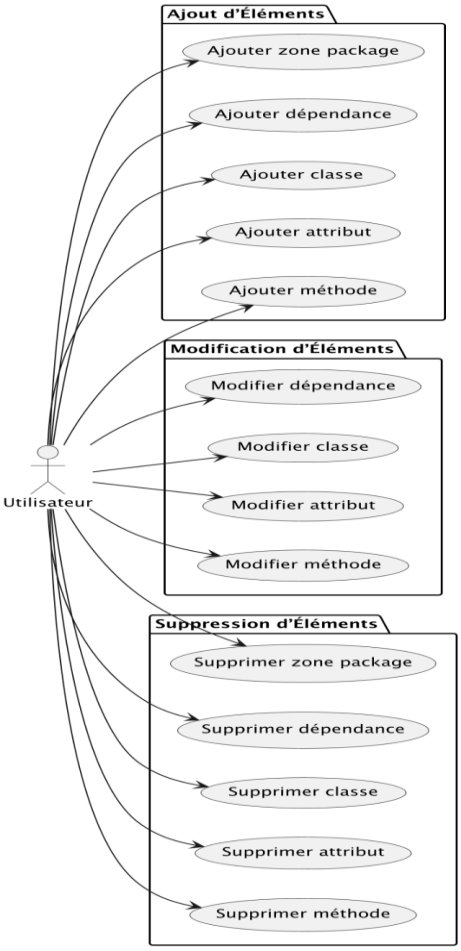
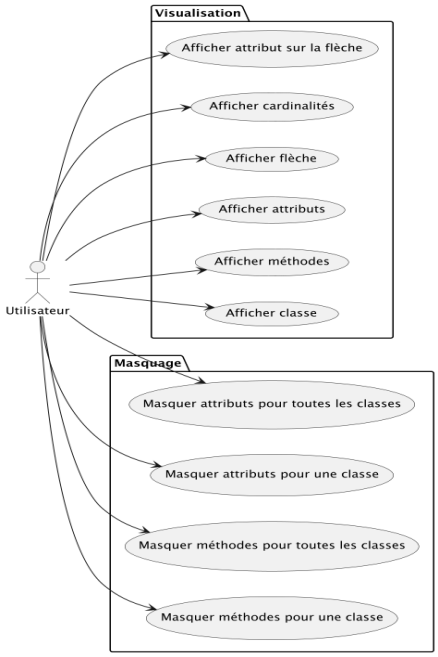


Diagramme de cas d'utilisation



Ce diagramme rassemble les différents systèmes de cas d'utilisations liées à la gestion de l'affichage des éléments graphiques.

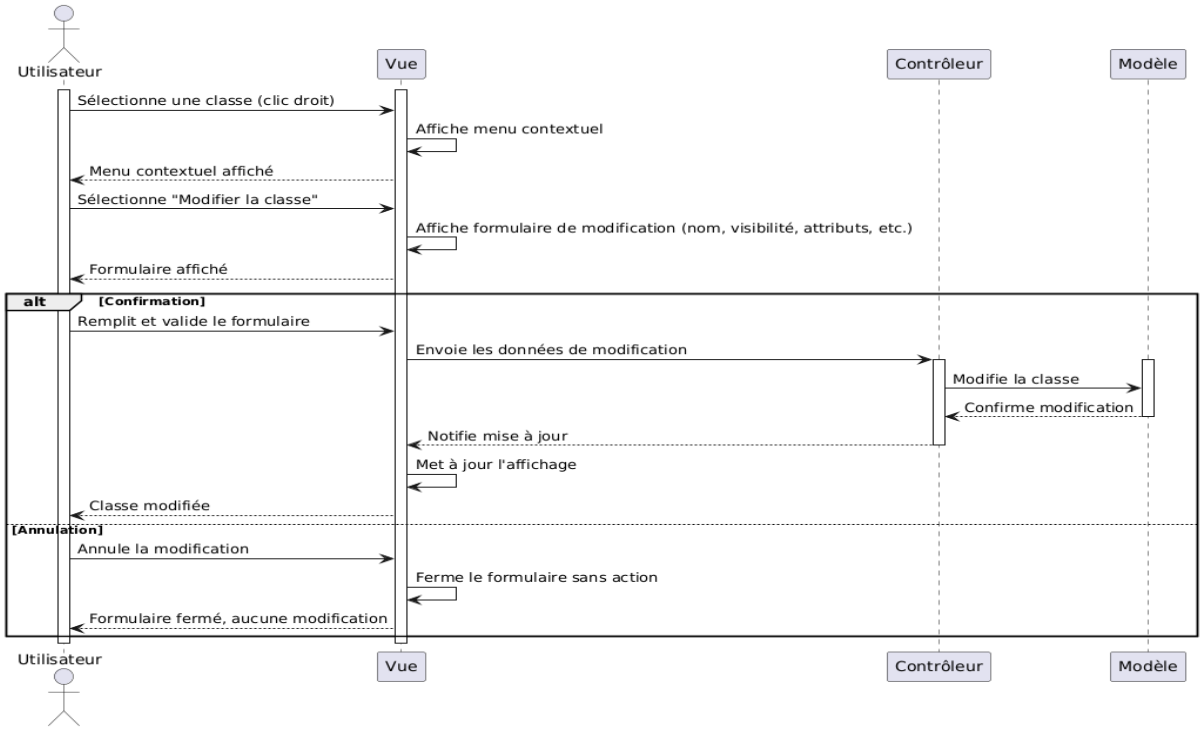
3. Descriptions textuelles, des DSS et/ou scénarios

Acteur :

- Utilisateur

DSS :

Modifier une classe



Description textuelle du cas "Modifier une classe".

Précondition :

- Il existe une classe dans l'interface.

Postcondition :

- La classe est modifiée, et ses changements sont visibles dans l'affichage, si l'affichage des classes est activé.

Déroulement normal :

1) L'utilisateur fait apparaître le menu contextuel de la classe sélectionnée.
2) Il sélectionne l'option "Modifier la classe".
3) Une fenêtre ou un formulaire de modification de la classe apparaît, permettant à l'utilisateur de changer le nom, la visibilité, les attributs ou autres propriétés.
4) L'utilisateur remplit et valide le formulaire.

5) La classe est modifiée dans le modèle.
6) La mise à jour est notifiée à la vue, et l'affichage est mis à jour en conséquence.

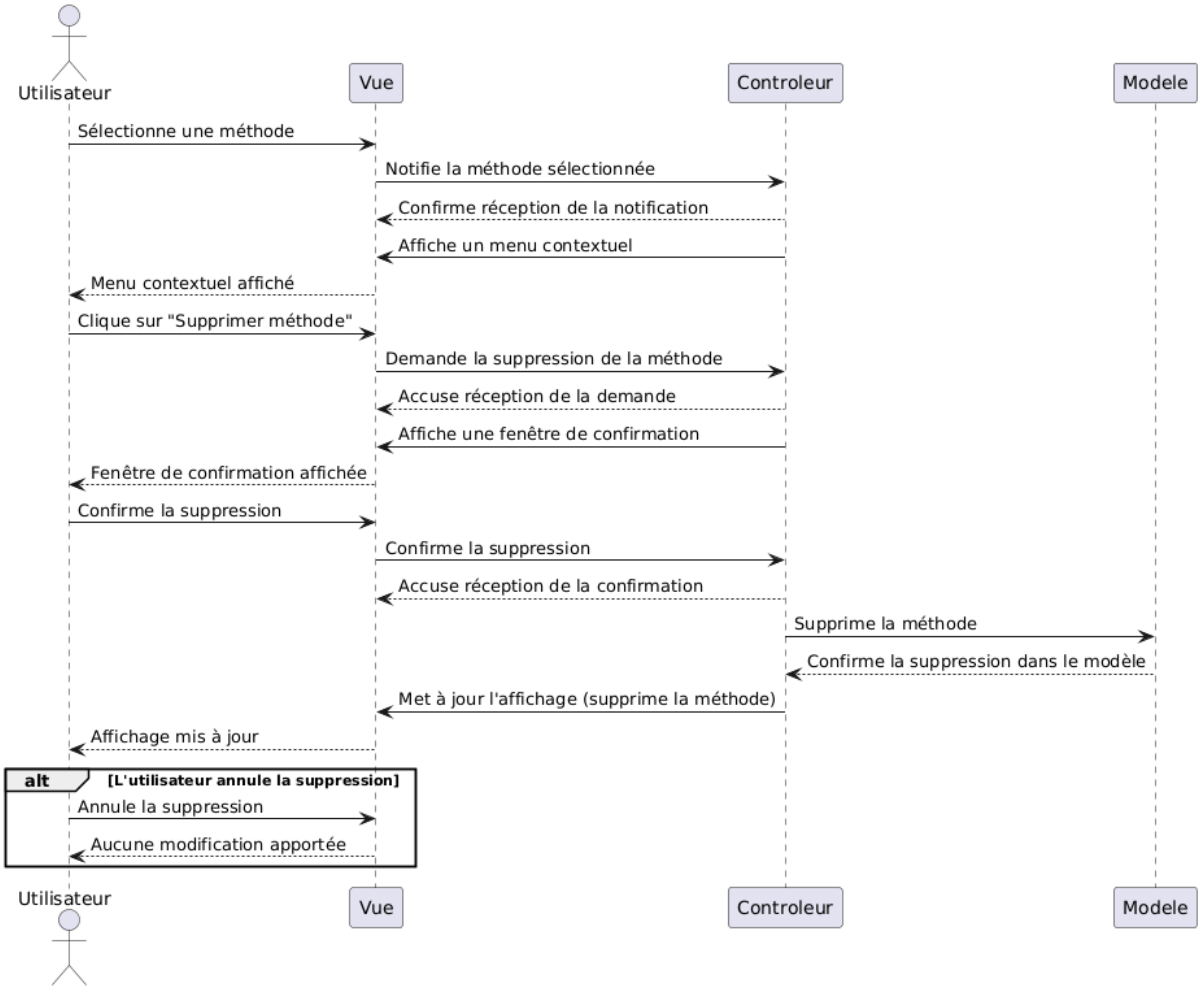
Variante(s) :

- Si l'utilisateur annule la modification, aucun changement n'est effectué et l'affichage reste inchangé.
- Si l'utilisateur ferme le formulaire sans valider, aucune modification n'est appliquée.

Contraintes non fonctionnelles :

- Le processus de modification doit être rapide et l'interface réactive

Supprimer une méthode



Description textuelle du cas "Supprimer une méthode".

Précondition :

- Des classes ayant des méthodes doivent être disponibles sur l'interface.

Postcondition :

- La méthode sélectionnée par l'utilisateur est supprimée donc elle ne s'affiche plus.

Déroulement normal :

1) L'utilisateur sélectionne la méthode concernée et fait un clic droit.
2) Un menu contextuel avec les actions possibles sur la méthode apparaît.
3) L'utilisateur clique sur "Supprimer méthode" pour la méthode qu'il voudra supprimer.
4) Une fenêtre de confirmation apparaîtra
5) L'utilisateur pour supprimer devra confirmer la suppression.
6) Le gestionnaire du modèle supprime la méthode et notifie à l'élément graphique.
7) L'affichage n'affiche plus la méthode supprimée.

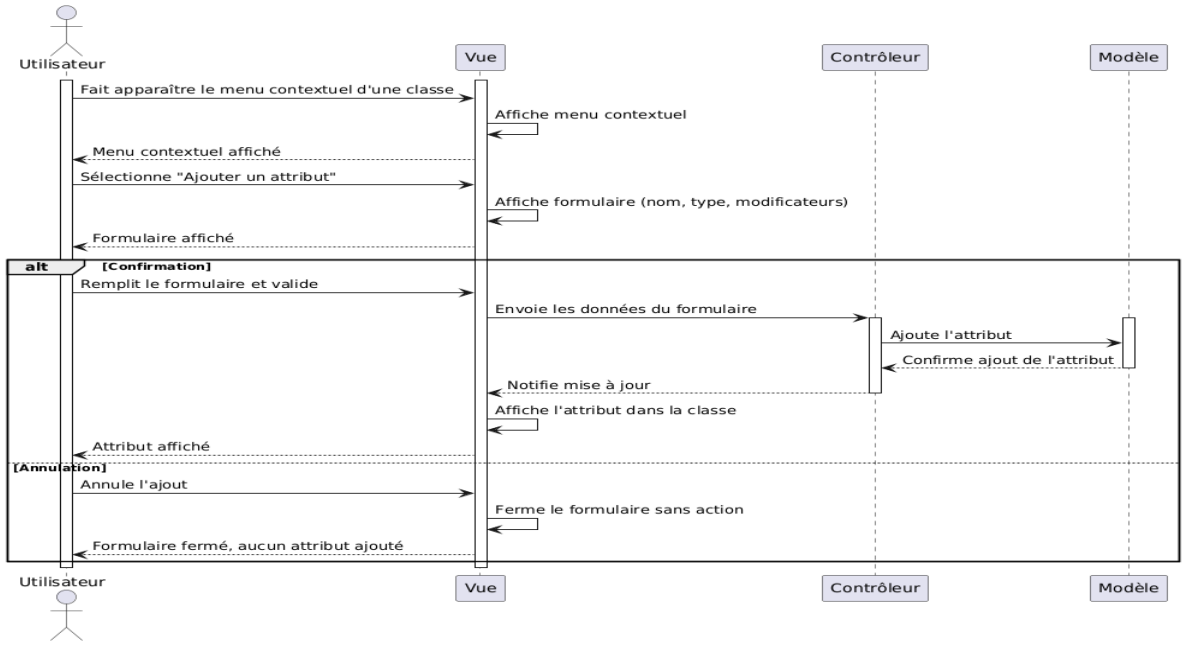
Variante(s):

- Au moment où la fenêtre de confirmation apparaît, si l'utilisateur clique sur annuler la méthode ne sera pas supprimé.

Contraintes non-fonctionnelles :

- Le gestionnaire du modèle doit supprimer la méthode sélectionnée au plus vite.

Ajouter un attribut



Description textuelle du cas "Ajouter un attribut".

Précondition :

- L'utilisateur a chargé un fichier Java qui contient des méthodes.

Postcondition :

- L'attribut ajouté est affiché si l'affichage des attributs n'est pas désactivé

Déroulement normal :

1) L'utilisateur fait apparaître le menu contextuel d'une classe
2) Il sélectionne l'option ajouter un attribut
3) Il remplit le formulaire contenant le nom de la variable, le type, et sélectionne parmi les modificateurs.
4) Il valide le formulaire
5) La méthode inscrite est affichée dans la classe sélectionnée.

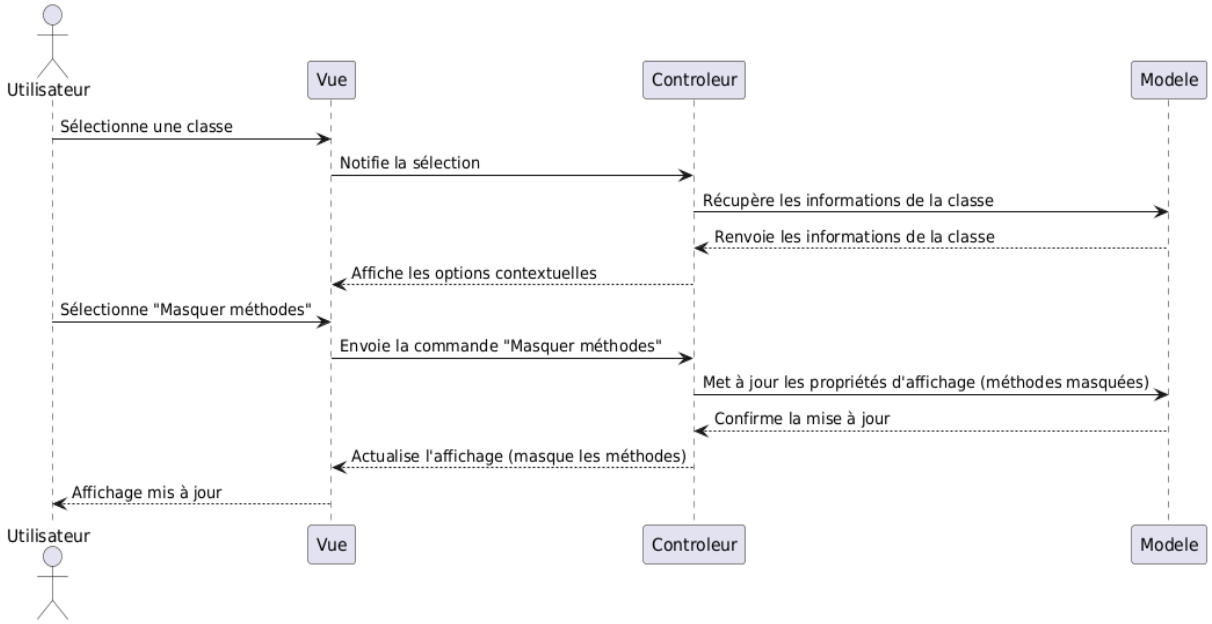
Variante(s):

- Au moment où le formulaire apparaît pour ajouter un attribut, si l'utilisateur clique sur annuler aucune action sera faite.

Contraintes non-fonctionnelles :

- Le système doit être réactif, moins de 5s pour mettre à jour l'affichage

Masquer les méthodes



Description textuelle du cas "Masquer les méthodes"

Précondition :

- Une ou plusieurs classes ayant des méthodes sont affichées.

- L'utilisateur souhaite masquer les méthodes pour simplifier l'affichage.

Postcondition :

- Les méthodes sélectionnées ne sont plus visibles sur le diagramme.

Déroulement normal :

1) L'utilisateur fait un clic droit sur une classe.
2) Il accède au menu contextuel et choisir l'option "Masquer les méthodes"
3) L'affichage est mis à jour pour ne plus afficher les méthodes de la classe sélectionnée.

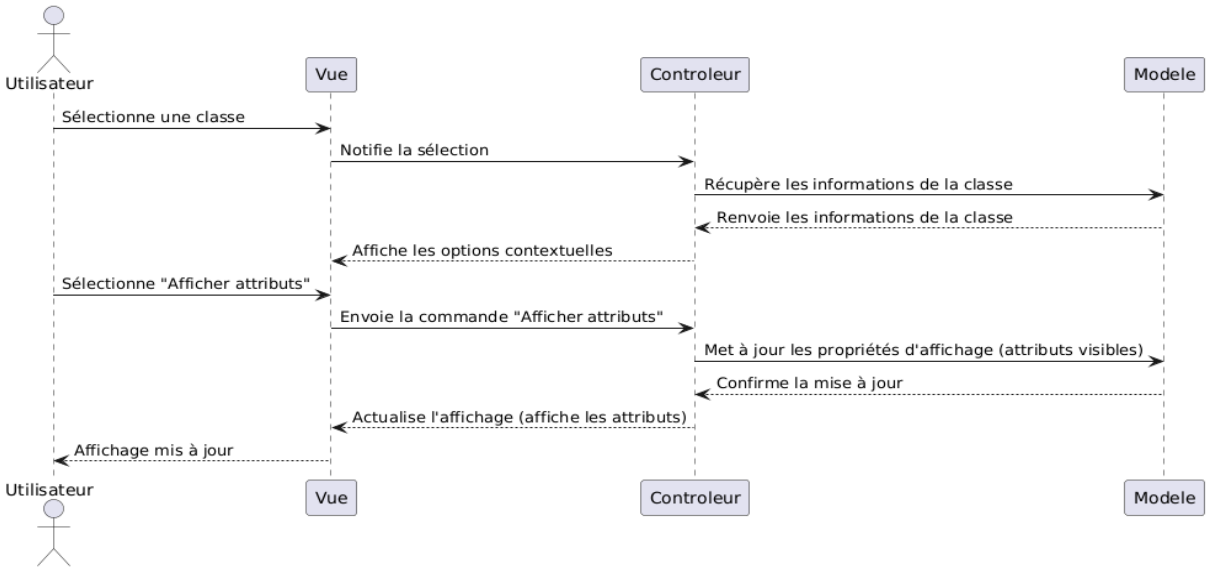
Variante(s):

- Si la classe n'a pas de méthode, aucune action ne sera faite.

Contraintes non-fonctionnelles :

- Le système doit être réactif et réaliser l'action au plus vite.

Afficher les attributs



Description textuelle du cas "Afficher les attributs"

Précondition :

- Une ou plusieurs classes sont disponibles dans l'interface.
- L'utilisateur a précédemment masqué les attributs ou veut voir les attributs de nouvelles classes.

Postcondition :

- Les attributs sont visibles pour les classes sélectionnées.

Déroulement normal :

1) L'utilisateur fait un clic droit sur une classe.
2) Il accède au menu contextuel et sélectionnez "Afficher les attributs".
3) L'affichage est actualisé pour inclure les attributs.

Variante(s) :

- Si les attributs sont déjà visibles, l'action ne pourra pas se faire.

Contraintes non-fonctionnelles :

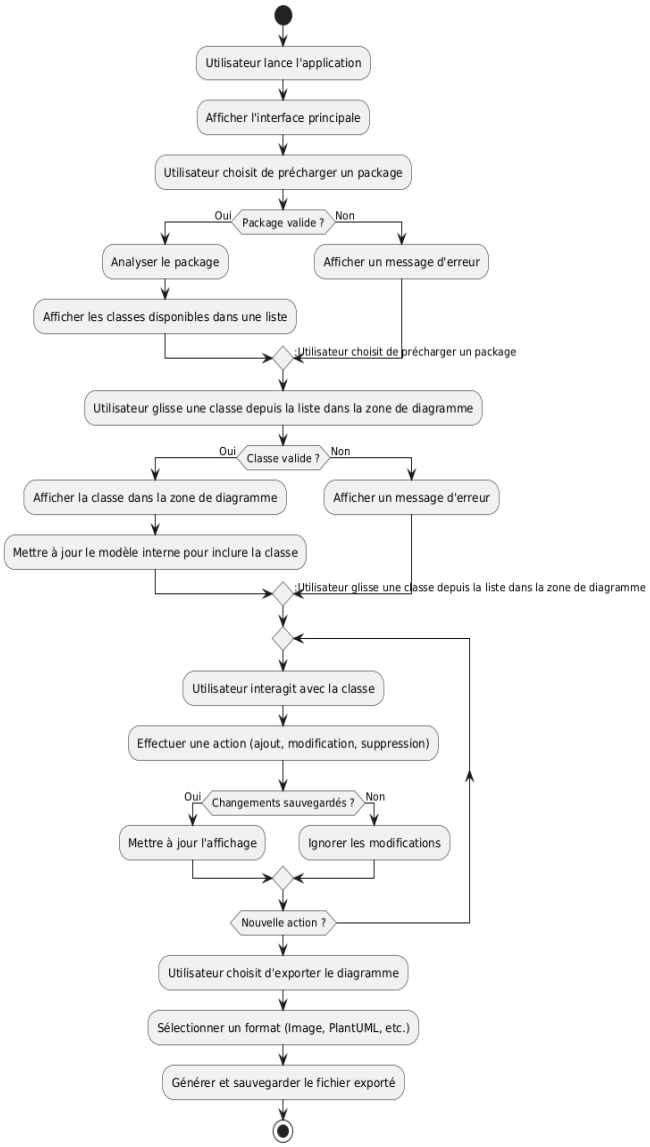
- Le système doit rafraîchir l'affichage au plus vite.

4. Diagramme d'activités de l'application.

Ce diagramme d'activités illustre donc le fonctionnement de l'application, il détaille dans ce cas les étapes principales de l'interaction utilisateur et traitement. Ici on a donc une représentation des flux d'actions possibles pour l'utilisation de l'application.

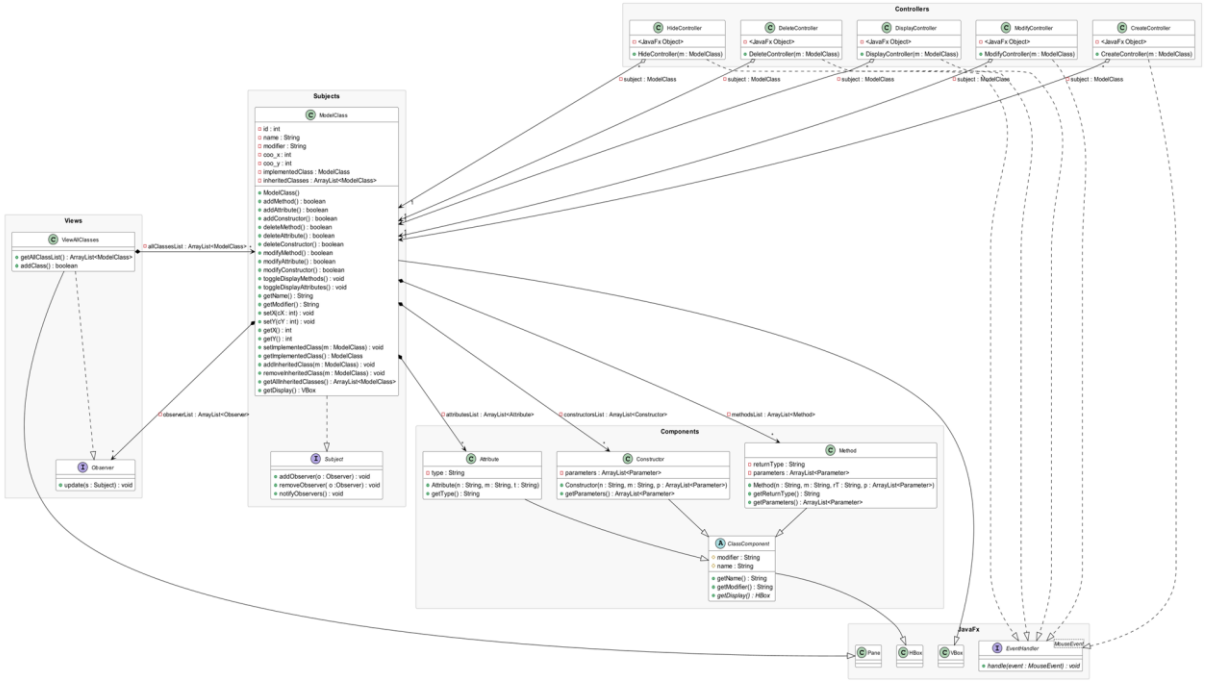
Pour ce diagramme, nous sommes dans le cas où l'utilisateur charge un package, effectue des actions comme glisser une classe dans la zone du diagramme, interagit avec celui-ci et sauvegarde son travail.

On peut donc identifier 4 "étapes" majeures sur ce diagramme, ces étapes vont être importantes lors de la conception de l'application car c'est à partir de cela qu'on se basera. Donc on modélise de cette manière on pourra utiliser le développement et test de l'application.



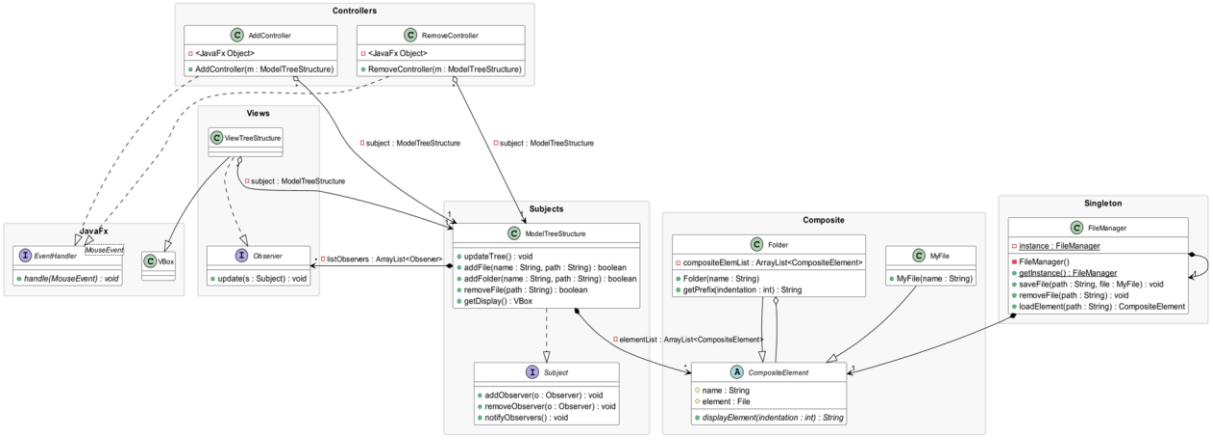
5. Conception, diagrammes de classe et patrons de conception prévus.

Observateur (pour MVC) :



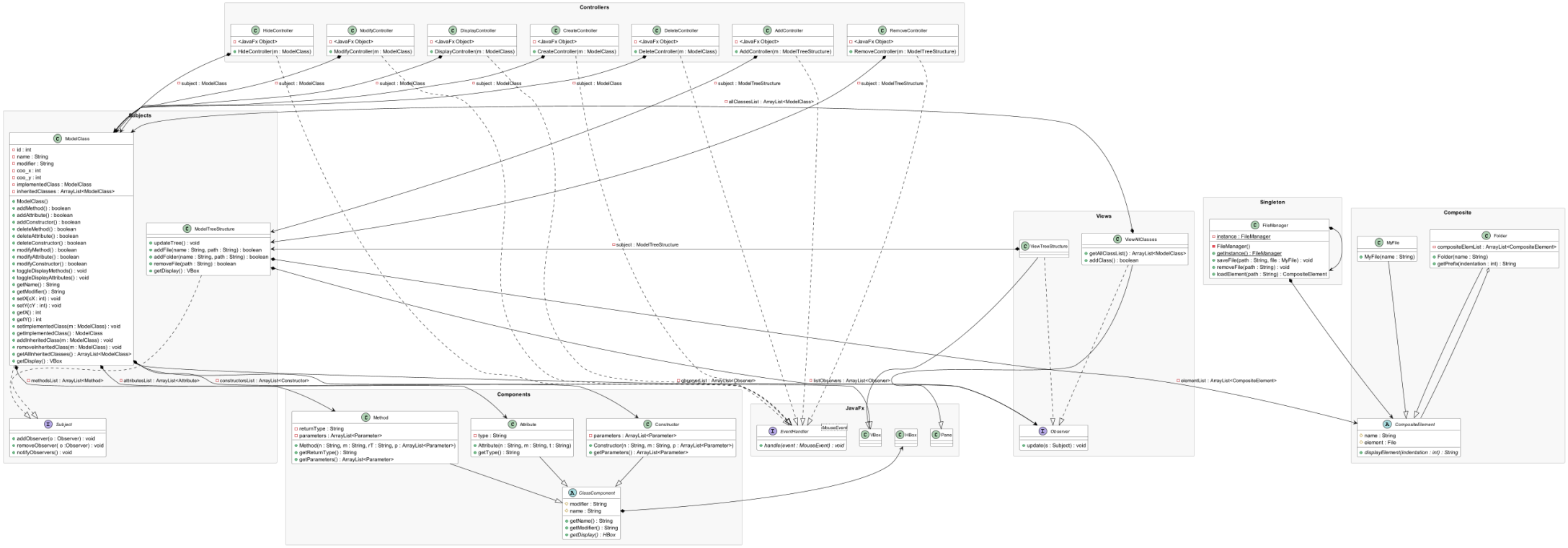
Le modèle d'architecture MVC est évidemment le cœur du projet, en effet, c'est le moyen idéal de gérer une application graphique. Ce modèle est présenté à travers deux patrons. Les vues sont responsables de l'affichage des données et les interactions utilisateur sont gérées par des contrôleurs sur les différents boutons.

MVC + Singleton + Composite :



Nous avons fait le choix d'utiliser le patron Singleton pour les cas d'utilisation lié aux chargements et exports des fichiers afin de s'assurer que ces actions ne soit gérée que par une instance. Ce patron est également lié au patron composite qui permet une gestion efficace des fichiers et dossiers chargés.

Diagramme complet



6. Maquette balsamiq de l'application.

Voir fichier Balsamiq adjoint.

7. Planning des itérations prévues et les objectifs de chacune (en termes de cas d'utilisations) avec identification des risques.

Lien du Trello: https://trello.com/b/9wCCWM52/sae-s301-java

Itération 1 - Mise en place des bases : (Patron MVC)

- Créer une classe from scratch
- Afficher classe
- Afficher méthodes
- Afficher attributs
- Afficher constructeur
- Ajouter méthode
- Ajouter constructeur
- Ajouter attribut

Risques : Mauvaise conception des vues, potentiel oublié d'ajouter la vue au modèle.

Itération 2 - Affichage dépendances :

- Ajouter une dépendance entre deux classes
- Afficher flèche
- Afficher cardinalités
- Afficher attribut sur la flèche

Risques : Interaction entre l'affichage et le modèle.

Itération 3 - Affichage : suppression, début des exportations :

- Supprimer méthode
- Supprimer constructeur
- Supprimer attribut
- Supprimer classe
- Supprimer dépendance
- Supprimer zone package
- Exporter en format squelette Java
- Exporter en format PNG

Risques : Une fois masqués, les éléments n'apparaissent plus.

Itération 4 - Modification direct des classes et dépendances ainsi que masquage :

- Masquer méthodes pour une classe
- Masquer méthodes pour toutes les classes
- Masquer attributs pour une classe
- Masquer attributs pour toutes les classes
- Modifier méthode
- Modifier constructeur
- Modifier attribut
- Modifier classe
- Modifier dépendance

Risques : Mauvaise conception des modèles.

Itération 5 - Arborescence des fichiers et répertoires (MVC + Composite + Singleton) :

- Afficher l'arborescence des classes dans le menu à gauche
- Charger un fichier
- Charger un package

Risques : Difficulté à charger les fichiers correctement depuis leur source, difficulté indentation de l'arborescence.

Itération 6 - Exportation :

- Exporter en format code plantUML
- Exporter en format image plantUML

Risques : Difficulté d'export en image : Résolution, format, allez générer l'image plantUML depuis plantUML…