RAPPORT DE TP : MINI-EDITEUR DE TEXTE

BINOME : ATTIOUMOU KONAN & KITOYI ABDOUL-HALIK

M1 MIAGE - GROUPE A

ENCADRANT DE TP : LE ROCH ADRIEN

Table des matières

Le but de ce TP est de développer un mini éditeur de texte en appliquant techniques de conception vues au cours du module d'ACO.

Le développement du mini-éditeur de texte est fondé sur les concepts et les fonctionnalités suivantes :

- Le texte à éditer est contenu dans un buffer.
- À l'intérieur de ce texte, l'utilisateur peut déterminer une sélection (avec un début et une fin).
- Tout texte saisi par l'utilisateur vient remplacer le contenu de la sélection dans le buffer.
- L'utilisateur peut copier le contenu de la sélection dans le buffer dans un presse-papier clipboard.
- Le contenu de la sélection peut aussi être copié dans le presse-papier puis supprimé (cut)
- L'utilisateur peut coller le contenu du presse-papier à la place du contenu de la sélection dans le buffer.
- L'interface homme-machine peut être de type textuel (console) ou graphique.

Une technique de développement en spirale est utilisée pour permettre de gérer trois versions successives du mini éditeur :

• La version 1 fournira les fonctionnalités de base indiquées ci-dessus.

• La version 2 permettra d'enregistrer et de rejouer les actions de l'utilisateur.

• La version 3 permettra de défaire et refaire les actions de l'utilisateur, sans limitation sur la longueur de l'historique (l'utilisateur peut ramener le contenu de l'éditeur dans son état initial).

Version 1.0

Fonctionalités de bases

L'architecture de base de l'application est donnée par le diagramme de classe ci-dessous :

Une Classe EngineImpl qui implémente les fonctionnalités de bases de l'application définies par l'interface Engine : insérer du texte, supprimer du texte, copier…

Le buffer est de type StringBuilder. Cette classe fournit des méthodes facilitant l'implémentation des fonctionnalités de bases de l'application.

La sélection du texte est gérée par un objet Selection de telle sorte que le buffer de la sélection et le buffer de l'engine sont les mêmes.

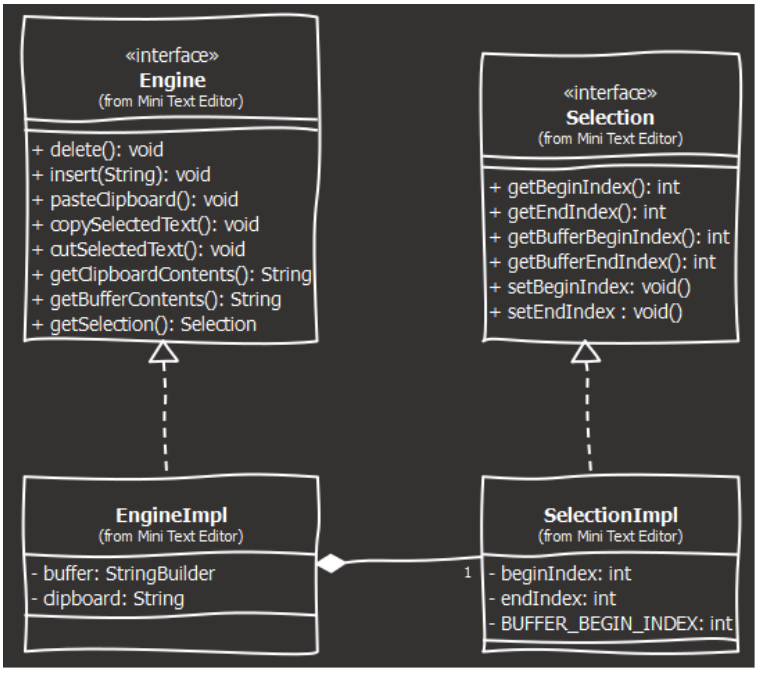


Figure 1 Fonctionnalités de base

Interface utilisateur

Nous avons choisi une interface utilisateur de type textuel (console). Pour faire le lien entre l'interface utilisateur et les fonctionnalités de bases, nous avons utilisé le patron de conception Command.

Le diagramme de classe obtenu avec les différents rôles :

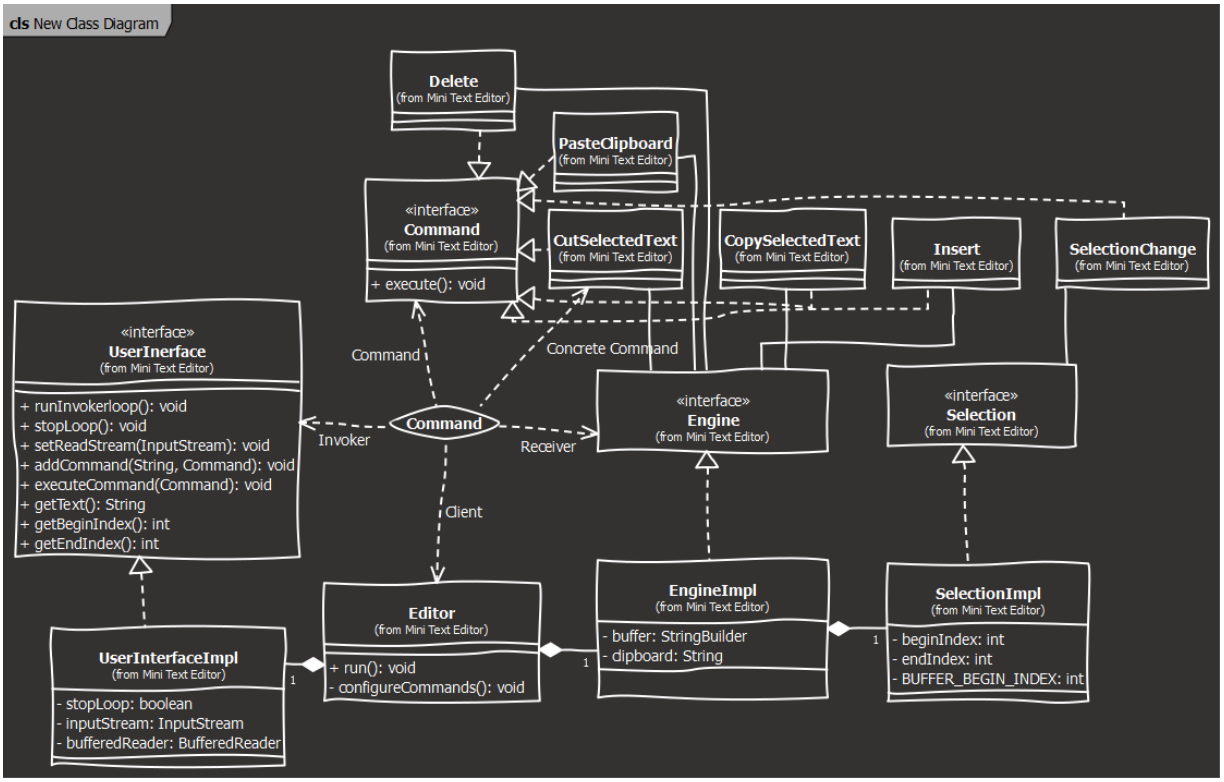


Figure 2 Diagramme de classe v1

L'application se trouve dans le package fr.istic.aco.editor.userinterface. Il faut compiler puis exécuter le fichier Editor.java.

La liste des commandes disponibles est la suivante :

- Insert : Insérer le texte juste après la commande.
- Select : Insérer dans un premier temps l'index de début ensuite l'index de fin.
- Copy.
- Cut.
- Paste.
- Delete.
- Undo : Remettre l'éditeur dans son état précédant
- Redo : Remettre l'éditeur dans un état ultérieur (possible si on a utilisé la commande précédente avant).
- Start : Démarrer l'enregistrement du recorder.
- Stop : Arrêter l'enregistrement du recorder.
- Replay : Rejouer toutes les commandes enregistrées par le recorder.
- Show : afficher le contenu du buffer.
- Index : afficher la position courante des indexes.
- Quit : Arrêter et quitter le programme.

Version 2.0

Pour l'enregistrement des commandes, nous avons utilisé le design pattern Memento.

Ci-dessous le digramme de classe obtenu ainsi que les différents rôles.

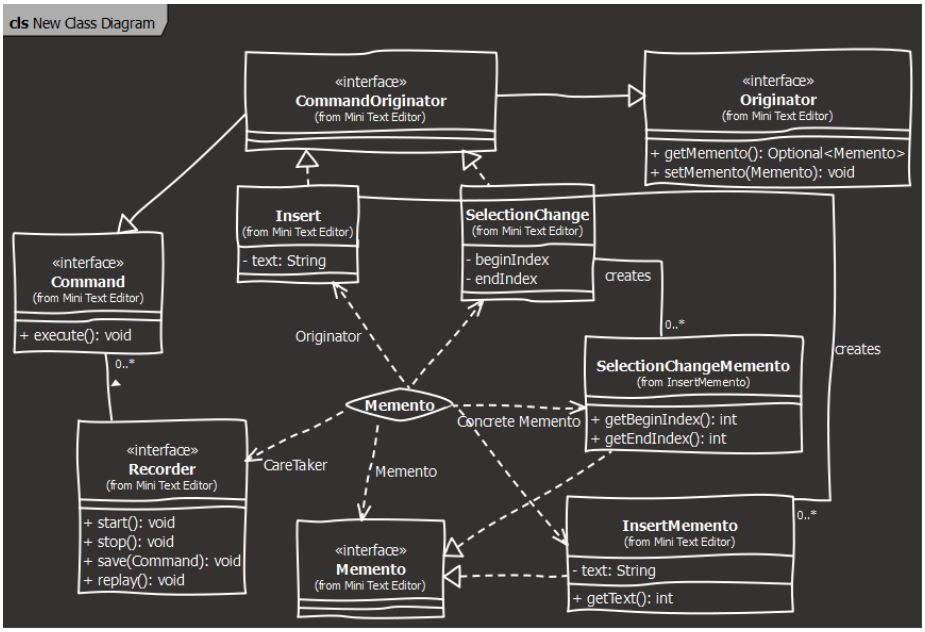
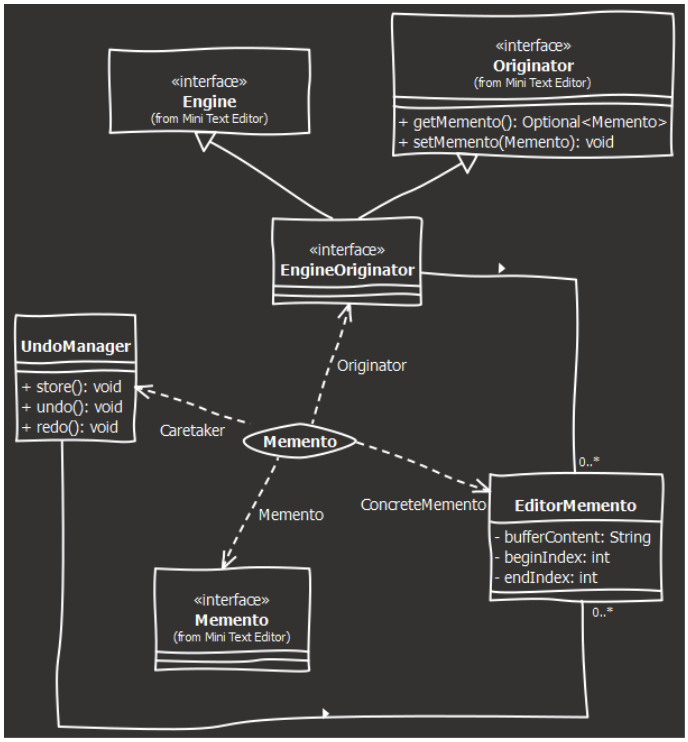


Figure 3 Diagramme de classe v2

Version 3.0

Pour cette version, nous avons choisi la solution qui consiste à sauvegarder les états de l'éditeur de texte. Cette solution a l'avantage d'être rapide. L'inconvénient est qu'elle nécessite beaucoup d'espace mémoire.



Ci-dessous le diagramme de classe final (sans les éléments de v3, Le diagramme devenait illisible) :

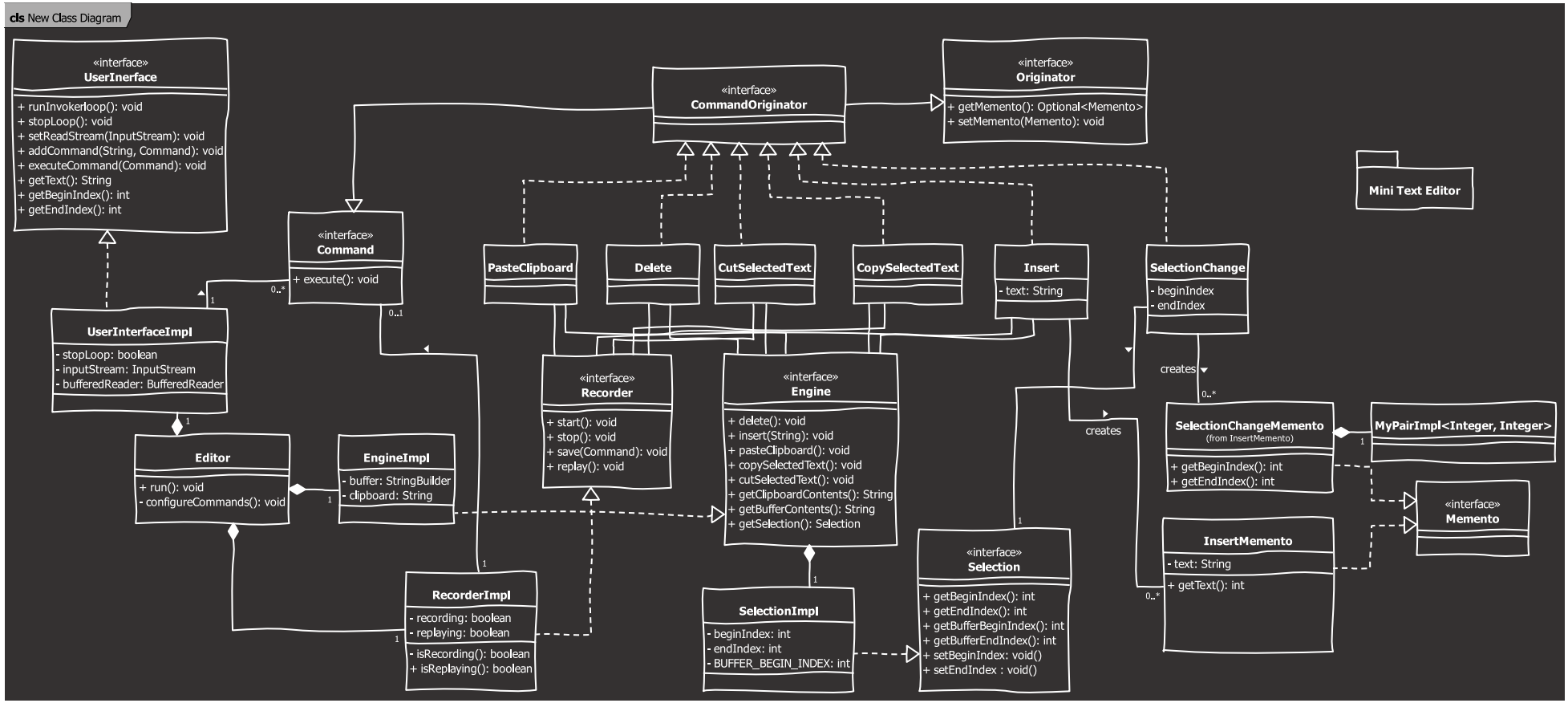


Figure 4 Diagramme de classe final