ISTIC UFR Informatique et Électronique
Université de Rennes 1, Campus de Beaulieu
263 , avenue du Général Leclerc
CS 74205, 35042 Rennes CEDEX, France
02 23 23 39 00

MASTER 1 INFORMATIQUE

PARCOURS MIAGE

Juvénal ATTOUMBRE

Nadège YEO

Sous la responsabilité de
Adrien LEROCH, adrien.leroch@univ-rennes1.fr

Mini éditeur de texte

Réalisé en décembre 2021

SOMMAIRE

1 Table des matières

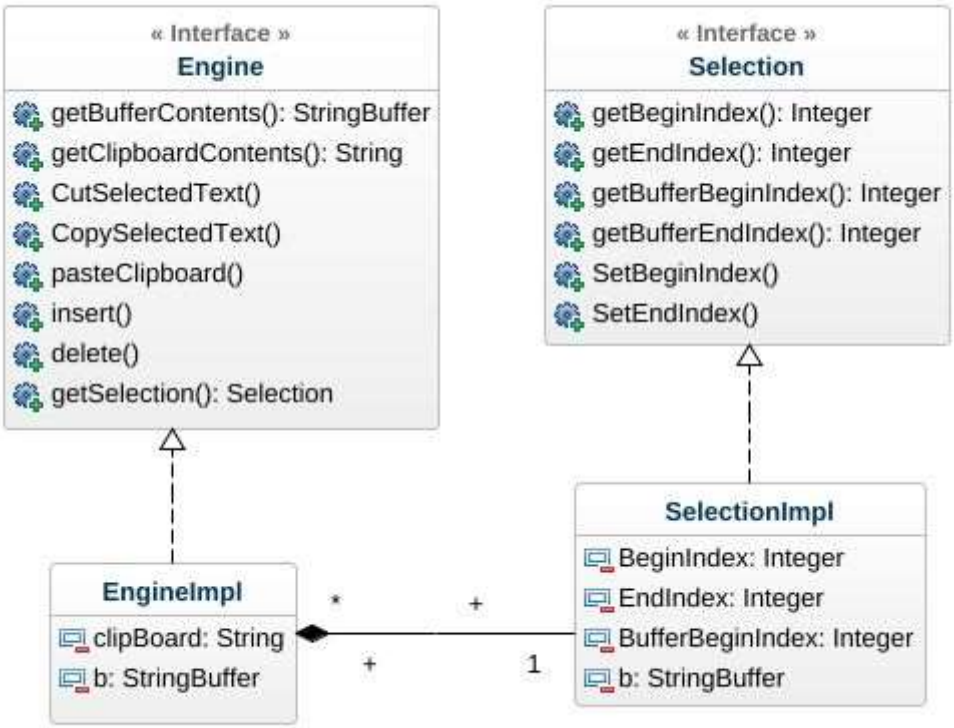
Introduction

L'Analyse et Conception d'Objet (ACO) est un module qui pour objectif de former l'étudiant à concevoir des logiciels robustes et professionnels. Ce module aborde les designs patterns et la représentation de ceux-ci par la méthode d'analyse et de conception de systèmes d'information Unified Modeling Language (UML).
Ce rapport présente une modélisation de d'un mini éditeur de texte en UML avec les designs pattern Command et Memento ainsi une implémentation en java du mini éditeur de texte.
L'implémentation de notre mini éditeur a été faite en 3 versions.

1. Première version

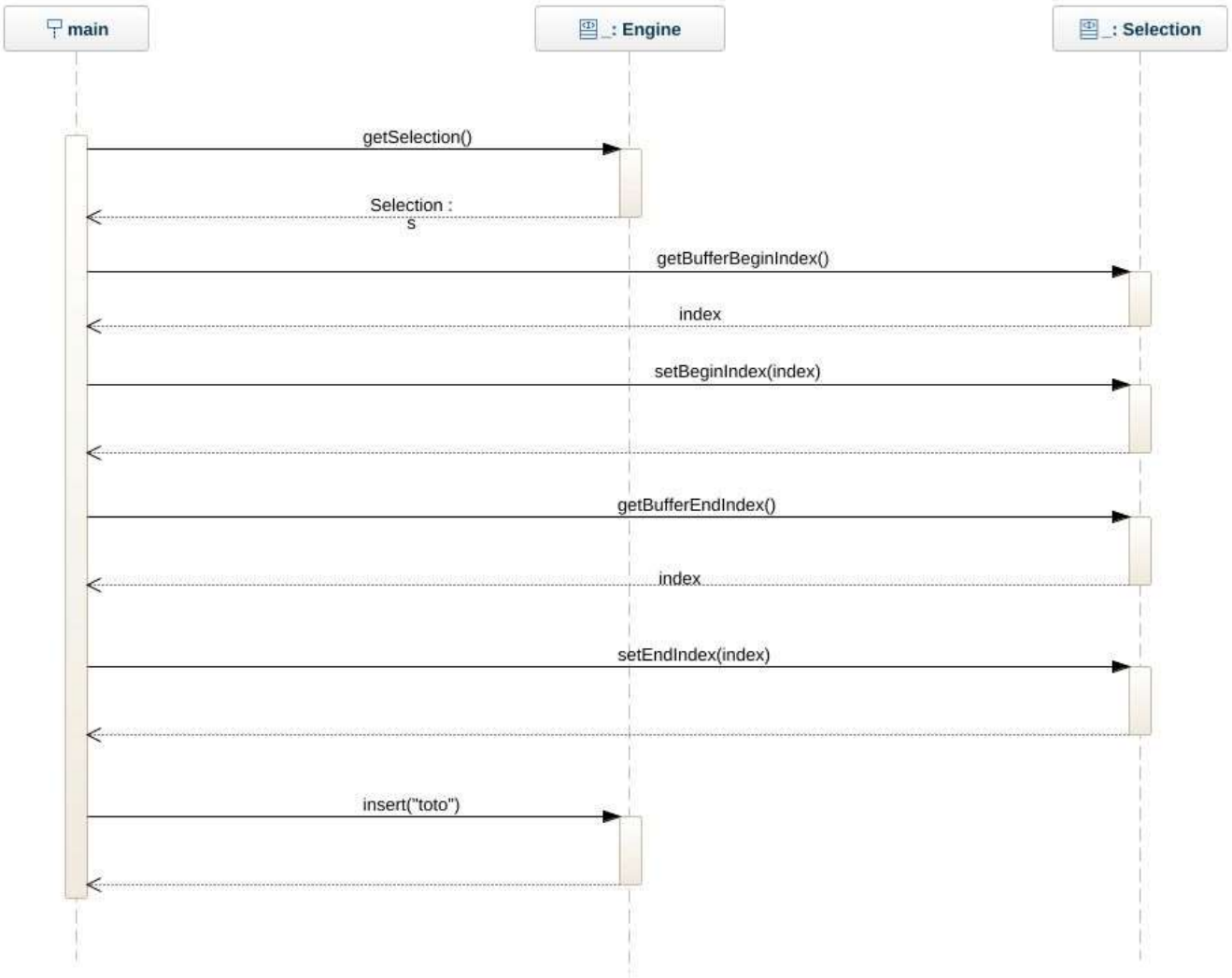
a) Diagramme de classe

La première version est le moteur de notre projet, le receiver au sens du pattern command.



b) Diagramme de séquence

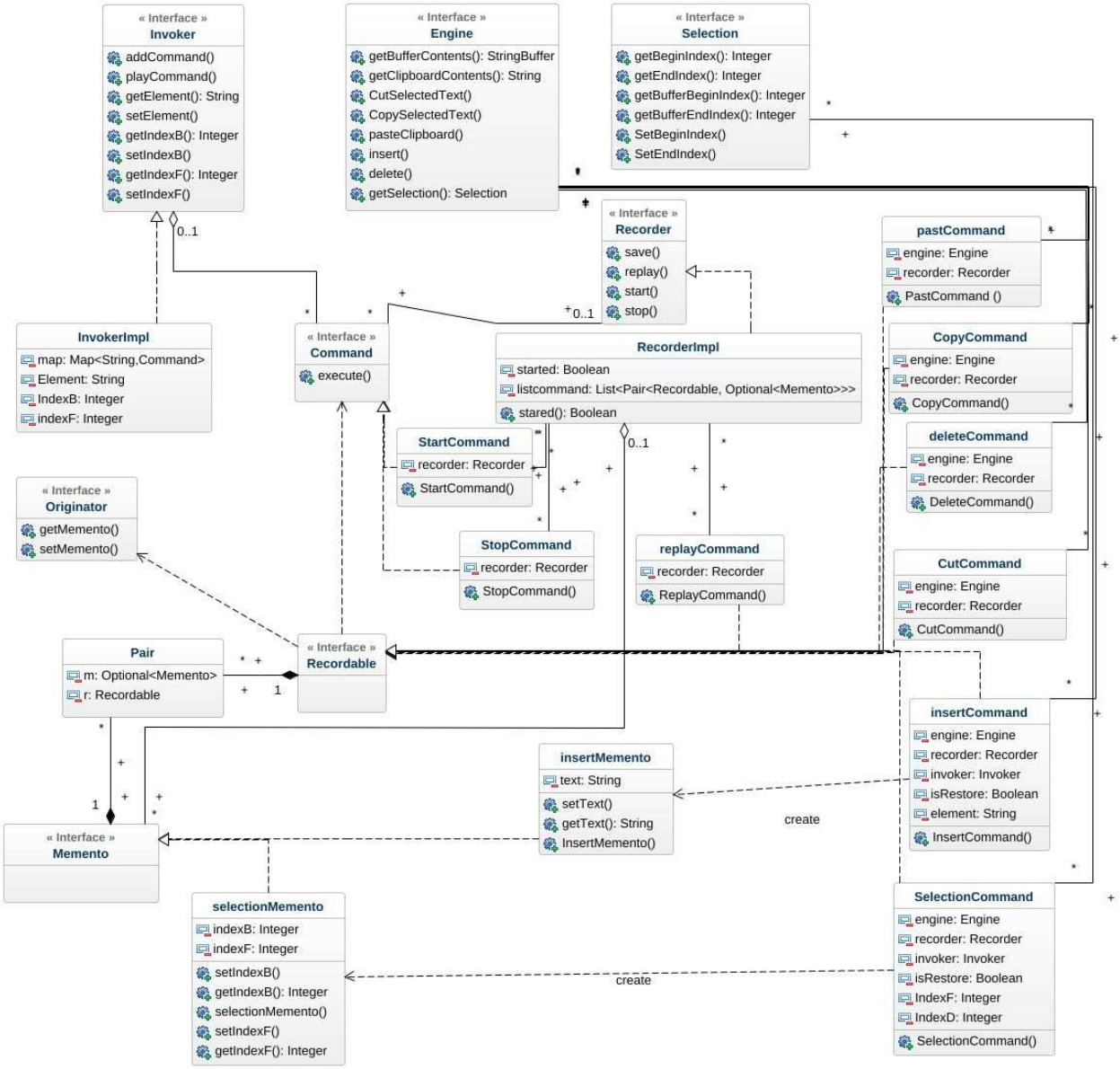
Nous vous présentons la selection et l'insertion d'un mot dans ce diagramme.



2) Deuxième version

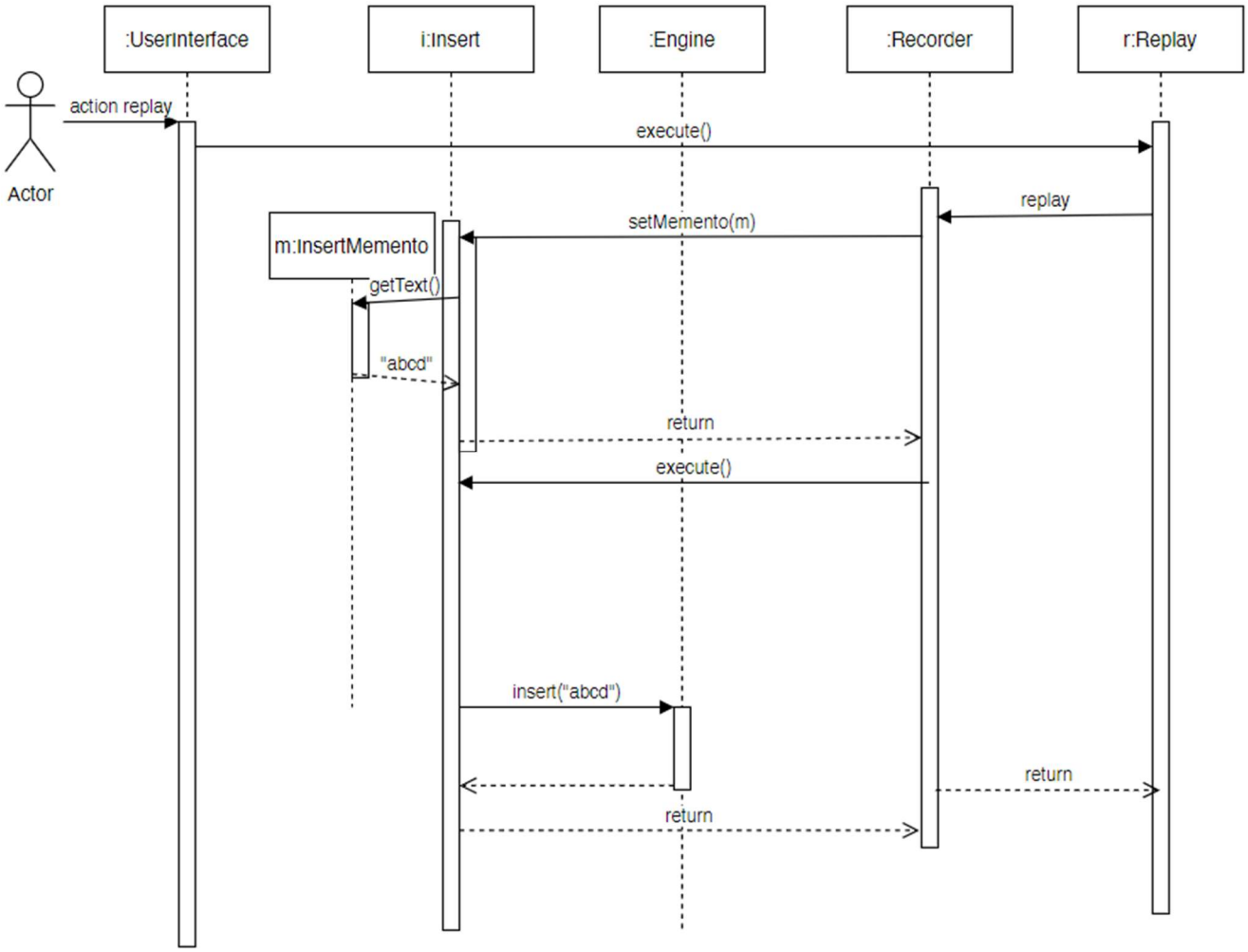
a) Diagramme de classe

Dans cette partie nous modélisons le fait qu'un utilisateur puisse rejouer une commande. Nous le faisons en combinant deux patrons de conceptions : Pattern Command et pattern Memento.
Le recorder nous permet de sauvegarder une commande avec son memento associé et de rejouer toutes les commandes jouées. Les memento permettent de stocker les elements de la selection ou de l'insertion.



b) Diagramme de séquence

Les sauvegarde d'une paire commande et memento se fait automatiquement à chaque fois qu'un utilisateur joue une commande. Lors du replay, le recorder se comporte comme un receiver au sens du pattern command.

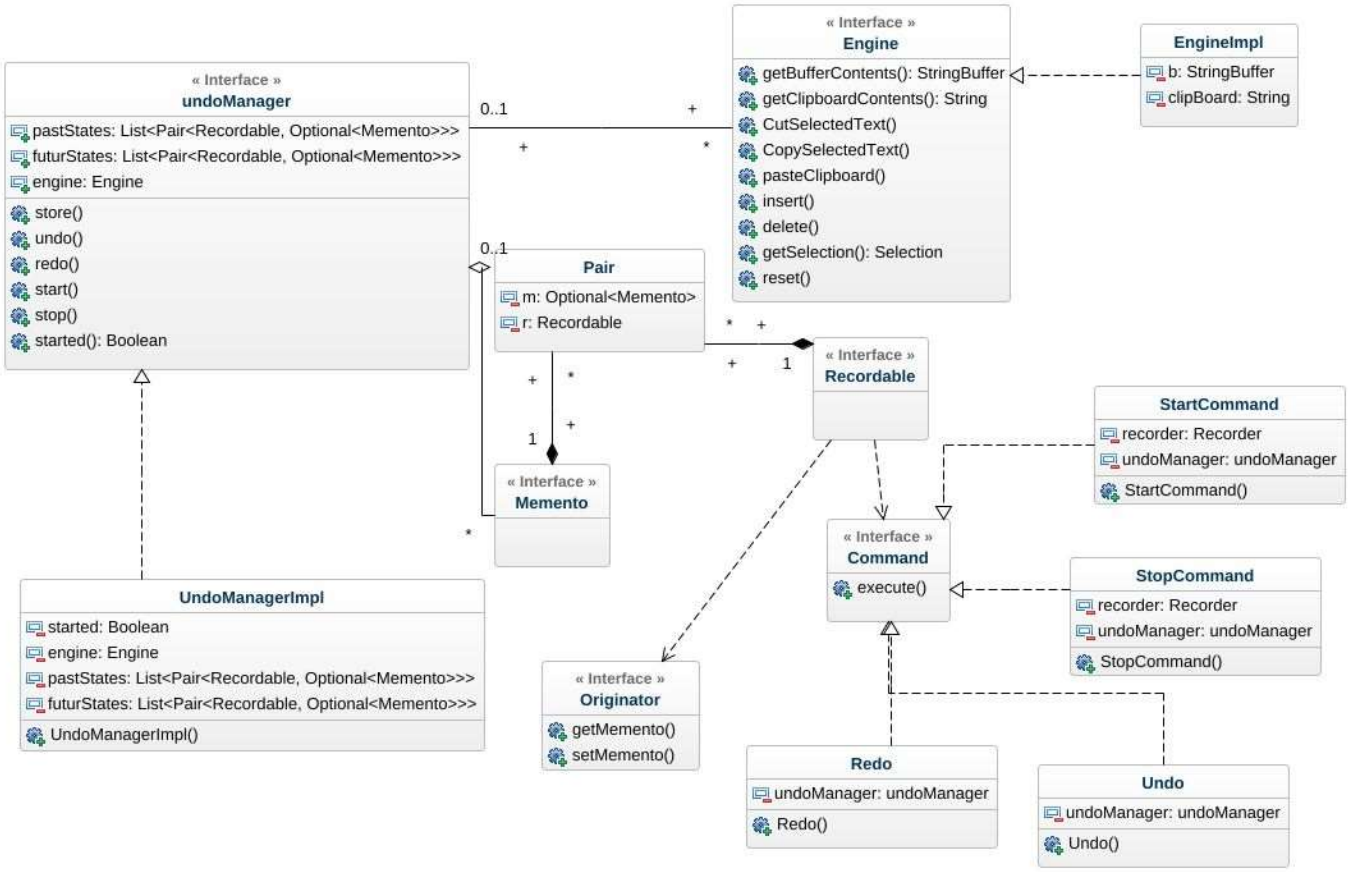


3) Troisième version

a) Diagramme de classe

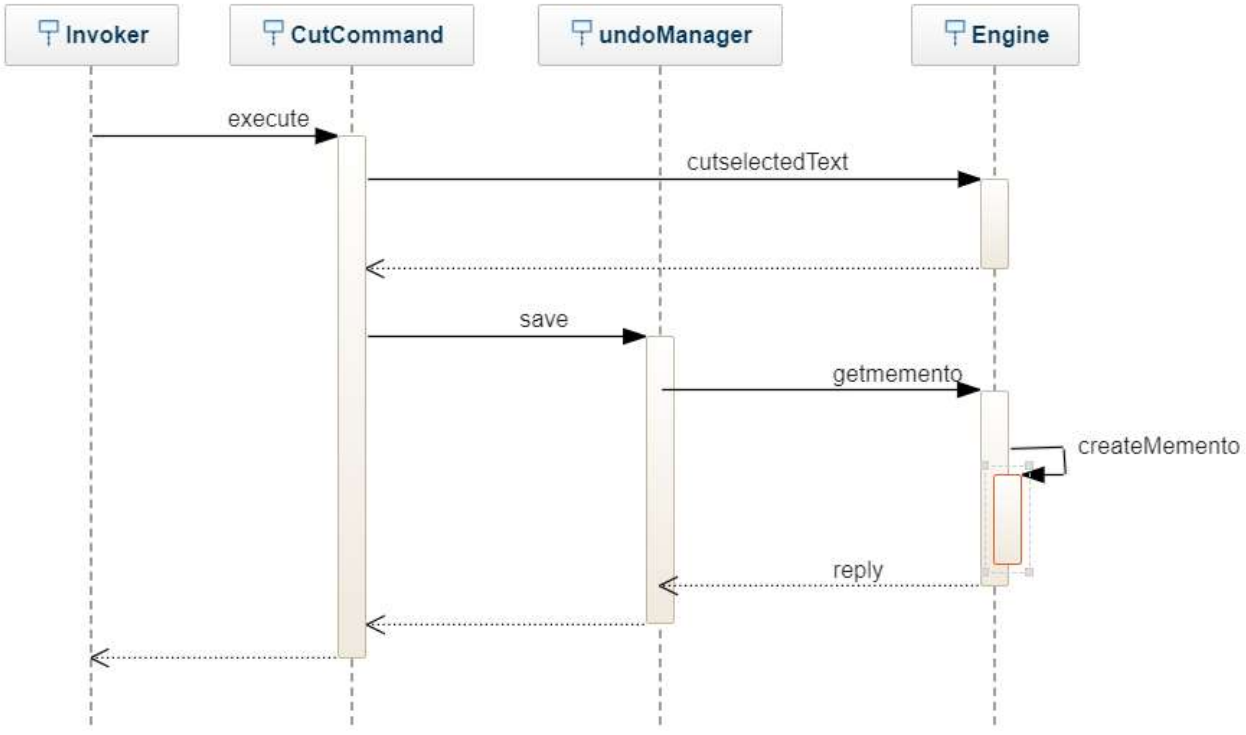
Pour cette version, nous avons choisi la solution 2 : sauvegarder toutes les commandes. L'idée est de supprimer l'engine et de réappliquer toutes les commandes enregistrées sauf la dernière. Son avantage est qu'elle est moins gourmande en mémoire par contre, elle est lente.

Dans cette partie, nous modélisons le fait que l'utilisateur puisse revenir à l'état précédent du buffer et de la selection. Après cela, il pourra aussi retrouver les états futurs qu'il avait déjà atteint.
Pour ce faire, on utilise le UndoManager qui se charge de sauvegarder les paires commande, memento à différents états par lesquelles passent ces composantes.



b) Diagramme de séquence

L'utilisateur fait un cutCommand par son execute. Celui-ci lance le cutselectedText de engine et renvoie une réponse au CutCommand. Celui-ci lance le save qui à sont tour génère un memento par getMemento.



Conclusion

Le mini-éditeur que nous avons réalisé est basé sur deux principaux Design Patterns à savoir Memento et Command. Nous avons appris à les utiliser ainsi qu'à appliquer certaines bonnes pratiques de programmations et d'éviter les mauvaises pratiques.

Les différentes versions du projet sont sur notre hébergeur :
https://gitlab.istic.univ-rennes1.fr/kattoumbre/tpaco
La version 1 : https://gitlab.istic.univ-rennes1.fr/kattoumbre/tpaco/-/tree/v1
La version 2 : https://gitlab.istic.univ-rennes1.fr/kattoumbre/tpaco/-/tree/v2
La version 3 : https://gitlab.istic.univ-rennes1.fr/kattoumbre/tpaco/-/tree/V3