

## **OMD-S7: TP2**

A rendre pour le 28/10/2024

*Responsable: FEUILLATRE Hélène*

# Sommaire

1. Introduction .....	3
2. Version 1 .....	4
2.1. Diagramme de cas d'utilisation .....	4
2.2. Description des cas d'utilisation .....	5
2.3. Diagramme de classe .....	6
2.4. Diagrammes de séquence .....	8
2.4.1. Couper la sélection .....	8
2.4.2. Entrer du texte .....	9
2.5. Implémentation du code .....	10
3. Version 2 .....	11
3.1. Diagramme de cas d'utilisation .....	12
3.2. Description des cas d'utilisation .....	13
3.3. Diagramme de classe .....	14
3.4. Diagramme de séquence .....	15
3.5. Implémentation du code .....	16
4. Conclusion .....	16

## 1. Introduction

Ce deuxième travail pratique consiste à concevoir un mini-éditeur de texte. Tout comme pour le premier travail, il débutera par la création des différents diagrammes, mais contrairement à lui, ce projet est plus complet avec deux versions à réaliser et l'implémentation du code. Pour commencer, nous allons analyser différents design patterns et en choisir un qui correspond le mieux à notre problème. Une fois ce design pattern adapté à notre problématique, nous implémenterons la première version. Ensuite, nous ajusterons notre architecture pour intégrer les fonctionnalités supplémentaires de la seconde version, que nous implémenterons à son tour.

## 2. Version 1

Dans la première version, les fonctionnalités présentes sont :

- La sélection du texte
- L'écriture de texte
- L'effacement du texte
- Le déplacement du curseur
- La copie de la sélection (sauvegardée dans le presse-papier)
- Le coupage de la sélection (sauvegardée dans le presse-papier)
- Le collage de la sélection qui fut au préalable stockée dans le presse-papier.

### 2.1. Diagramme de cas d'utilisation

Au vu des fonctionnalités que doit proposer la première version de l'éditeur de texte, voici un diagramme de cas d'utilisation envisageable :

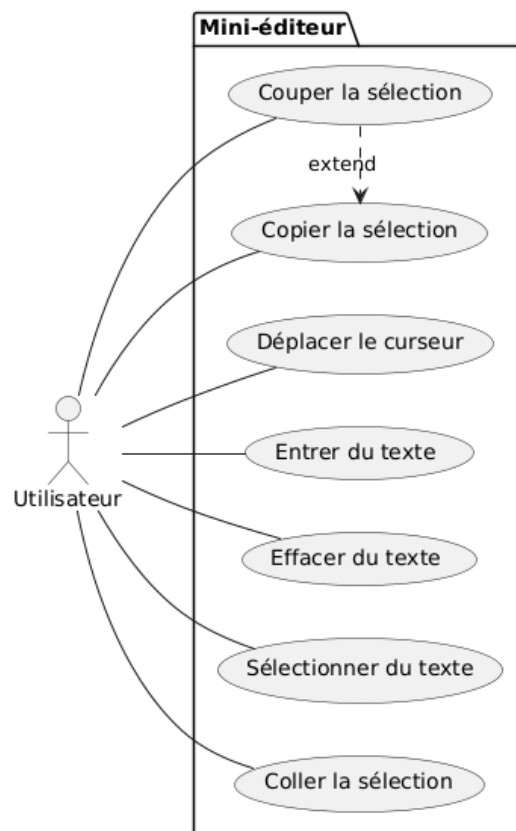


Figure 1: Diagramme de cas d'utilisation pour la première version.

## 2.2. Description des cas d'utilisation

Voici la description de chaque cas d'utilisation. Nous ne détaillerons pas les acteurs puisque nous pensons qu'il n'y a qu'un acteur qui est l'utilisateur de l'application.

Cas d'utilisation	Scénario nominal	Scénario alternatif	Scénario exception
Déplacer le curseur	a) L'utilisateur appuie sur les touches directionnelles. <ul style="list-style-type: none"> <li>gauche/droite pour aller à gauche/droite</li> <li>bas/haut pour aller à début/à la fin</li> </ul> b) Le curseur se déplace c) Sa nouvelle position est affichée.		a) La position cible du curseur est invalide (en dehors du texte) : le curseur s'arrête à la limite valide.
Entrer du texte	a) L'utilisateur saisit des caractères. b) Les caractères sont ajoutés à la position du curseur c) La zone de texte est mise à jour.	a) Le texte est ajouté à une sélection existante : remplace la sélection.	a) Caractère non autorisé (caractères spéciaux) : remplacement par dièse
Effacer du texte	a) L'utilisateur appuie sur la touche "Supprimer" ou "Backspace". b) Le caractère ou le texte sélectionné est supprimé c) a zone de texte est mise à jour.	a) Aucun texte à supprimer (curseur au début du texte).	
Sélectionner du texte	a) L'utilisateur utilise Shift + touches directionnelles. (gauche/droite : sélectionne à gauche/droite) b) La portion du texte souhaitée est sélectionnée.	a) L'utilisateur peut sélectionner tout le texte avec Ctrl + A	a) Tentative de sélectionner au-delà des limites du texte : sélection seulement jusqu'aux limites autorisées
Copier la sélection	a) L'utilisateur sélectionne du texte. b) L'utilisateur appuie sur Ctrl+C.	a) Aucun texte sélectionné : vide le presse-papier et rien n'est copié.	a) Le presse-papier est plein.
Coller la sélection	a) L'utilisateur positionne le curseur à l'endroit souhaité. b) L'utilisateur appuie sur Ctrl+V. c) Le contenu du presse-papier est inséré à la position du curseur.	a) Le contenu du presse-papier est ajouté à une sélection existante : remplace la sélection.	a) Le presse-papier est vide : rien ne se passe
Couper la sélection	a) L'utilisateur sélectionne du texte. b) L'utilisateur appuie sur Ctrl+X. c) Le texte est supprimé de la zone de texte et copié dans le presse-papier.	a) Aucun texte sélectionné : vide le presse-papier et rien n'est copié.	

## 2.3. Diagramme de classe

Au vu du problème que nous avons à modéliser, après avoir consulté le [catalogue](#) des design pattern présent sur [Refactoring GURU](#), nous avons décidé de nous baser sur le design pattern nommé Command.

Ce design pattern est de type comportemental, c'est-à-dire qu'ils concernent les algorithmes et l'attribution des responsabilités entre les objets. Plus précisément, le design-pattern Command permet de transformer une requête en un objet qui contient toutes les informations relatives à la requête. Cette transformation est utile car elle permet de transmettre des requêtes en tant qu'arguments de méthode, de retarder ou de mettre en file d'attente l'exécution d'une requête et de prendre en charge des opérations annulables. C'est exactement le type de structure que l'on recherche pour notre mini-éditeur.

Voici la structure exposée par ce design pattern :

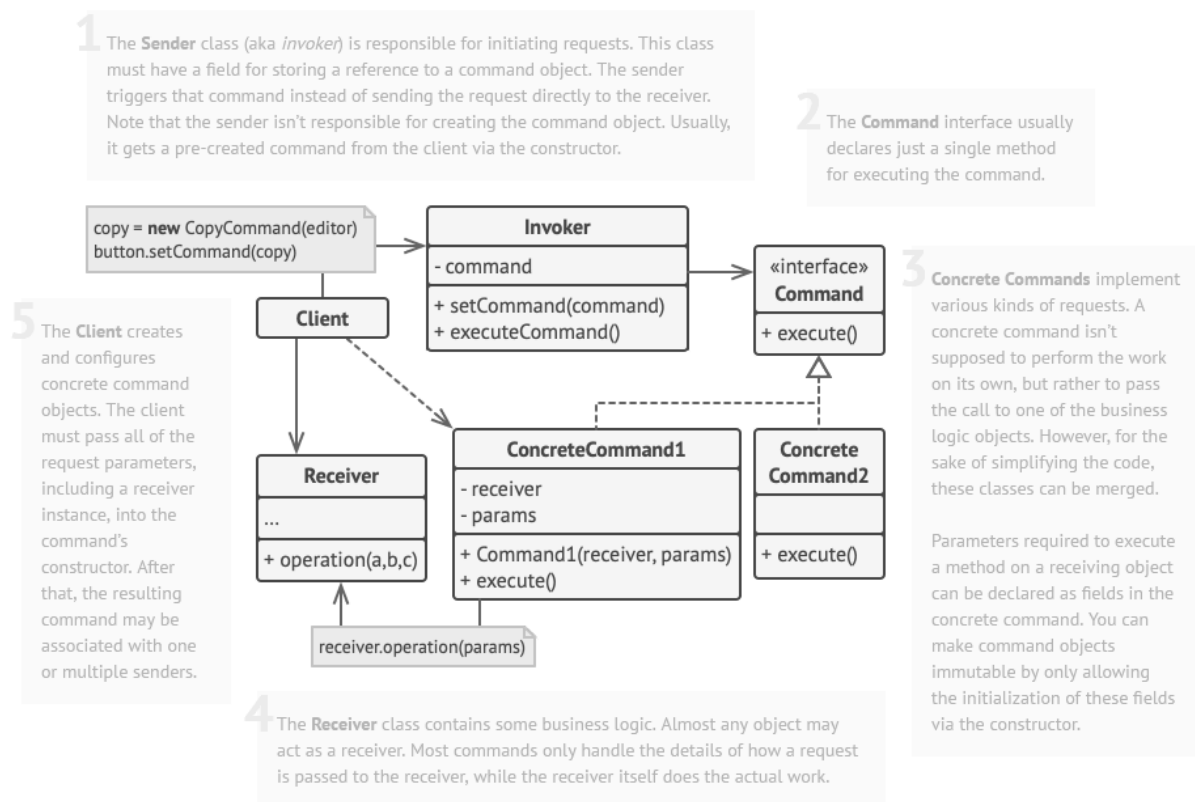


Figure 2: Design Pattern Command

Cette structure est très générale. Nous devons l'adapter à notre problème. Pour ce faire, nous avons du comprendre la structure ci-dessus. Une fois cela fait, nous avons pu la customiser pour produire notre propre diagramme de classes modélisant notre problème. Voici comment nous l'avons adapté :

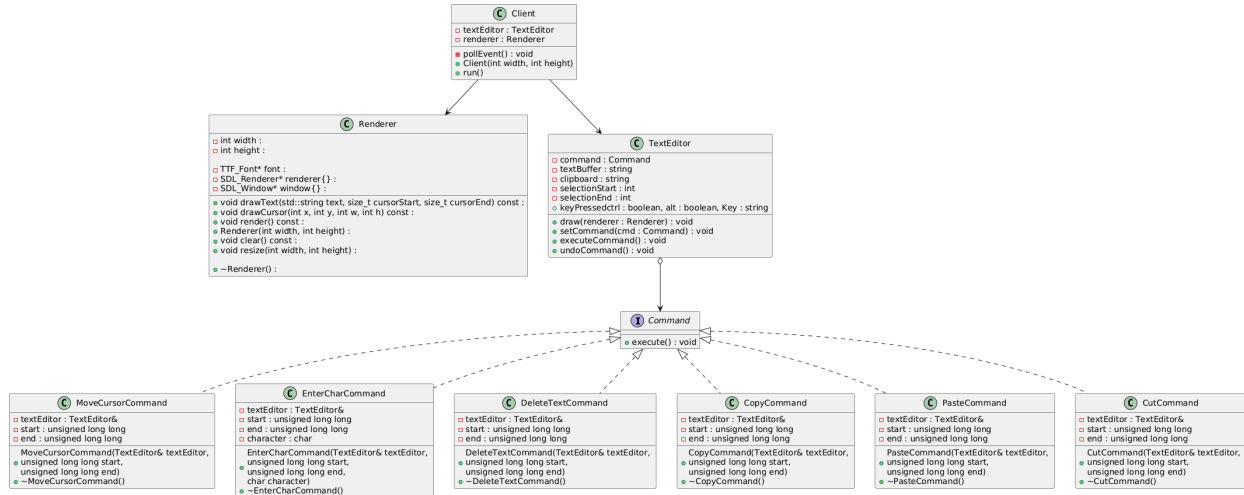


Figure 3: Diagramme de classe basé sur le Design Pattern Command pour la Version 1

Comme vous pouvez le constater, nous avons repris la même structure que le Design Pattern Command, adapté au langage de programmation choisi. Nous avons choisi C++ qui nous semblait être un bon choix étant donné que nous pouvons utiliser la SDL.

## 2.4. Diagrammes de séquence

Pour bien comprendre comment vont se produire les interactions avec les classes lors de l'utilisation de l'application, nous avons produit deux diagrammes de séquences.

### 2.4.1. Couper la sélection

Voici notre diagramme de séquence pour la situation où l'on souhaite couper la sélection.

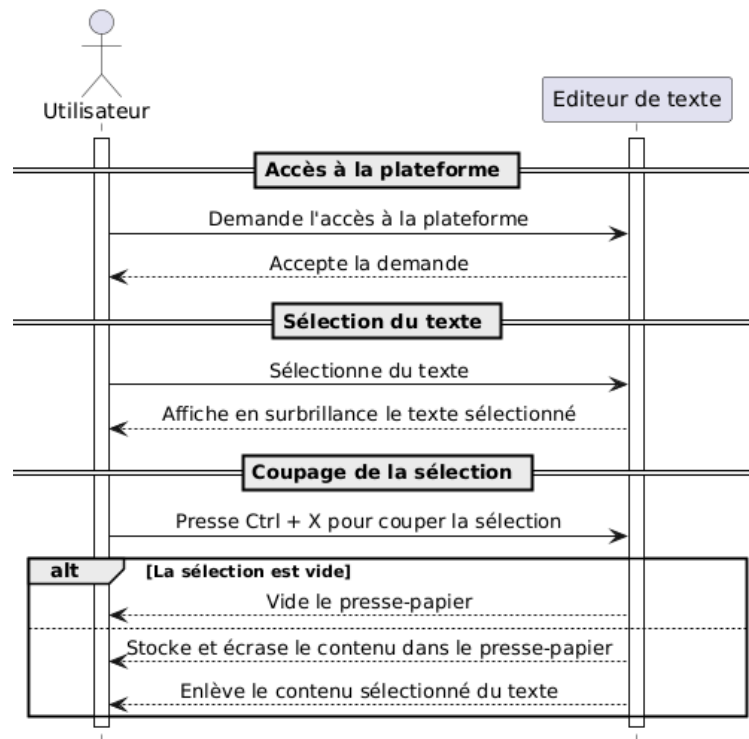


Figure 4: Diagramme de séquence pour couper la sélection

expliquer



### 2.4.2. Entrer du texte

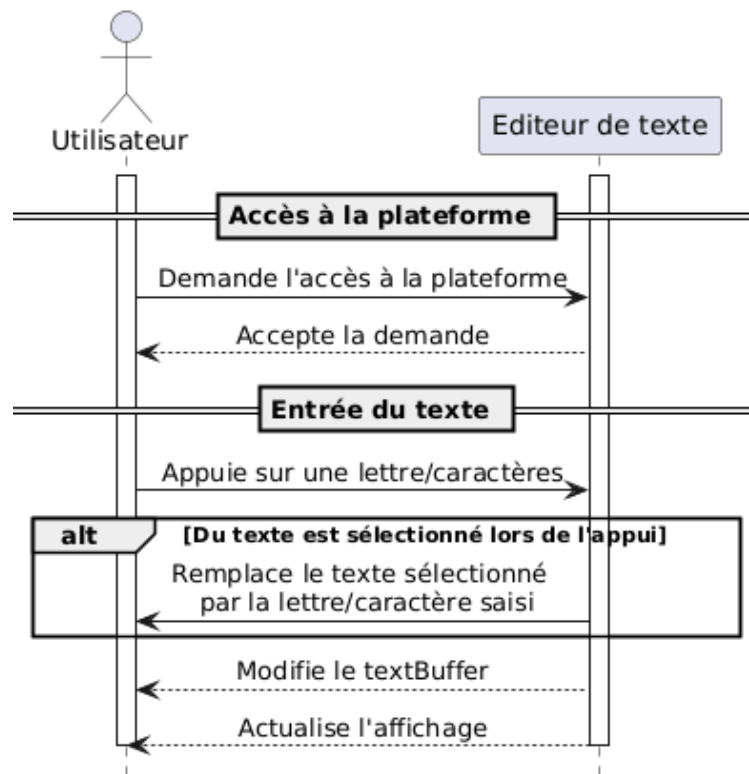


Figure 5: Diagramme de séquence pour écrire du texte

expliquer

## 2.5. Implémentation du code

Conformément aux consignes et aux diagrammes que nous avons pu établir, nous avons réalisé la première version de l'éditeur de texte. Vous pourrez le trouver **dire où**

**il faudra pas non plus oublier de séparer les deux versions et de revoir les diagrammes si besoin -> surtout le diagramme des classes**

### 3. Version 2

Dans la seconde version, les fonctionnalités ajoutées par rapport à la première version sont les suivantes :

- L'enregistrement des actions de l'utilisateur pour pouvoir les rejouer, par exemple dans un script
- L'enregistrement de ses actions
- L'annulation possibilité d'annuler de ses actions

### 3.1. Diagramme de cas d'utilisation

### 3.2. Description des cas d'utilisation

ceux de base + les autres

### 3.3. Diagramme de classe

### 3.4. Diagramme de séquence

### 3.5. Implémentation du code

## 4. Conclusion

faire (max 10 lignes) il fallait bien penser dès la v1 à la v2 pour avoir un code maintenable

<https://refactoring.guru/design-patterns> <https://refactoring.guru/design-patterns/catalog>

<https://refactoring.guru/design-patterns/memento>