Rapport de Conception : Mini-éditeur de Texte

Noms du binôme :

- Nom 1: Fatoumata Thioro Bah
- Nom 2: Oumou Sadio Bah

Formation : Master 1 MIAGE

Groupe de TP : C
Nom de l'encadrant : Adrien le Roch

1. Présentation Générale du Projet

Ce projet consiste à développer un mini-éditeur de texte avec les fonctionnalités suivantes :

1. Gestion d'un buffer contenant le texte à éditer.
2. Possibilité pour l'utilisateur de déterminer une sélection dans ce texte.
3. Fonctionnalités d'insertion, suppression, copie, coupe, et collage de texte.
4. Extension vers des versions avancées permettant l'enregistrement et la relecture des actions (V2) ainsi que les opérations d'annulation et de restauration (V3).

Le développement suit une approche incrémentale en spirale, avec trois versions principales :

- **Version 1 (V1)** : Fonctionnalités de base (buffer, insertion, coupe, copie, collage).
- **Version 2 (V2)** : Enregistrement et relecture des commandes.
- **Version 3 (V3)** : Gestion de l'historique complet des actions avec annulation et rétablissement.

2. Architecture UML et Choix de Conception

L'architecture de l'éditeur repose sur les concepts suivants :

1. **Buffer et Clipboard** :
    - Le **buffer** représente le texte à éditer.
    - Le **clipboard** permet de stocker temporairement les sélections copiées ou coupées.

2. **Commandes et Design Pattern Command** :
    - Les opérations (insert, cut, copy, paste, undo, redo) sont encapsulées dans des objets commande.
    - Chaque commande implémente une interface commune pour assurer une exécution et un enregistrement cohérents.
3. **Memento et Originator** :
    - Le patron Memento permet de sauvegarder et restaurer les états du buffer.
    - L'interface **Originator** gère les sauvegardes et restaurations via des instances de memento.

Diagramme de Classe : V1

UML représentant les interfaces comme Invoker, Engine, Selection, Command, et leurs commandes concrètes.

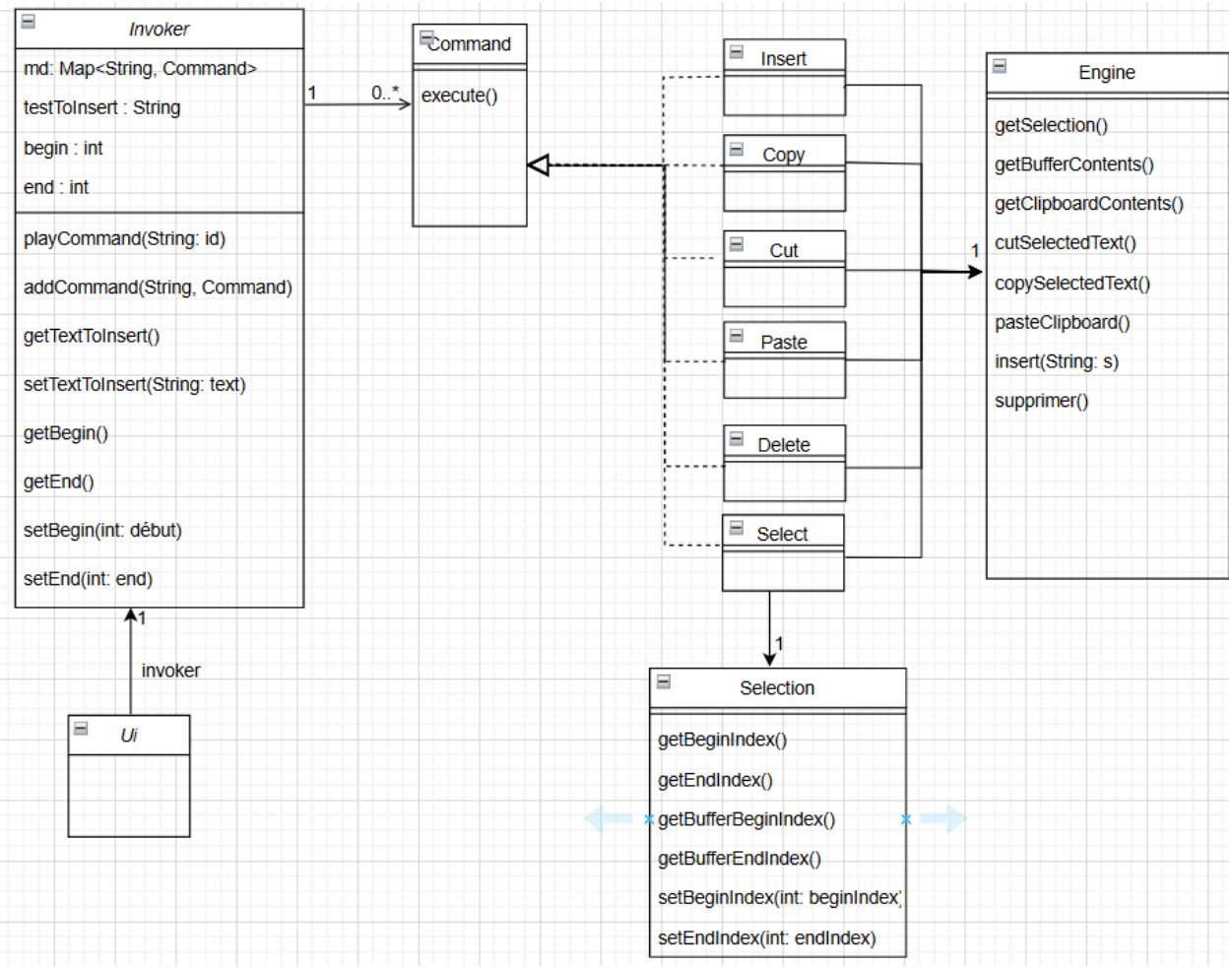


Diagramme de Classe : V2

UML représentant les interfaces comme Invoker, Engine, Selection, Command, Recorder et leurs commandes concrètes

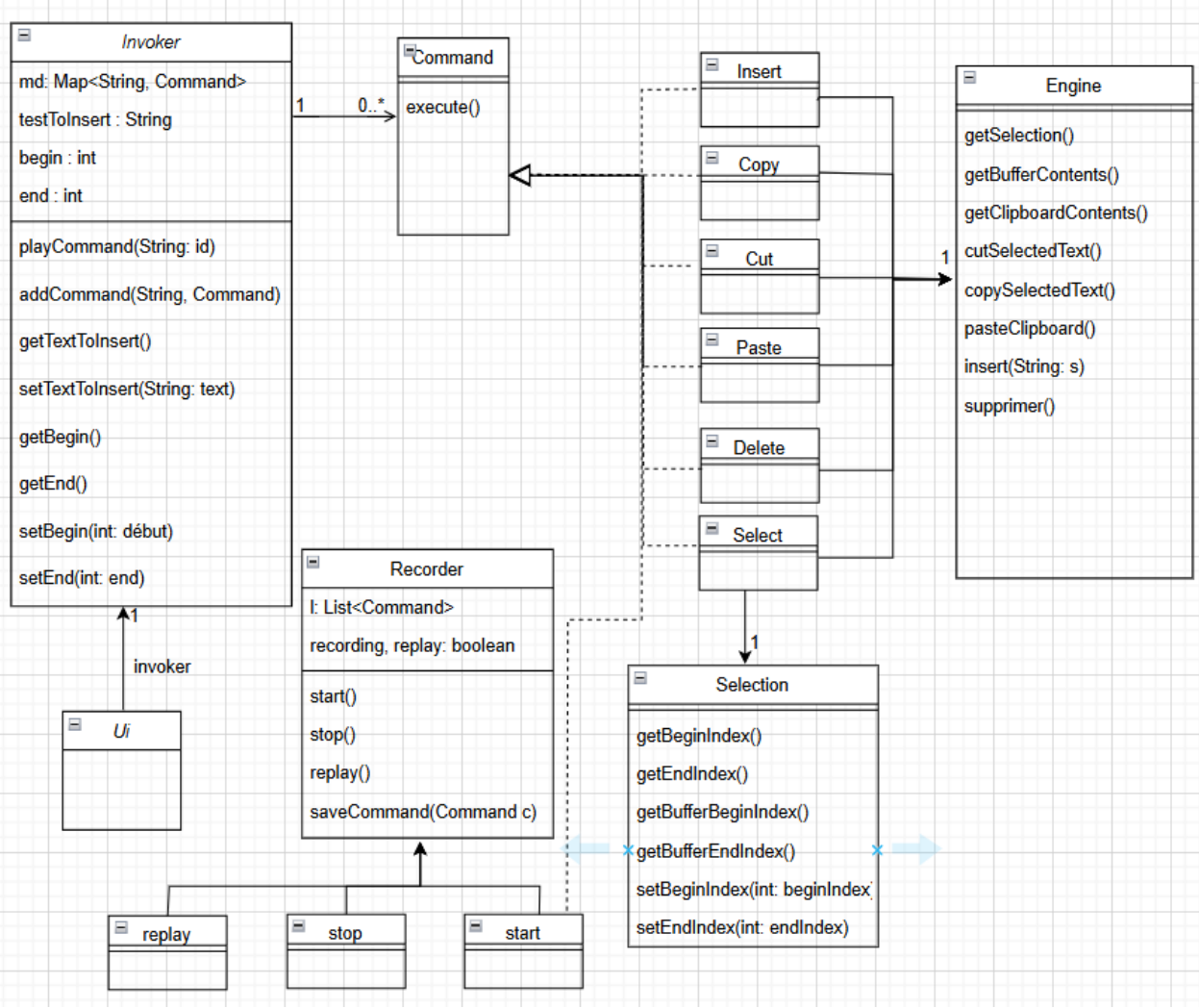
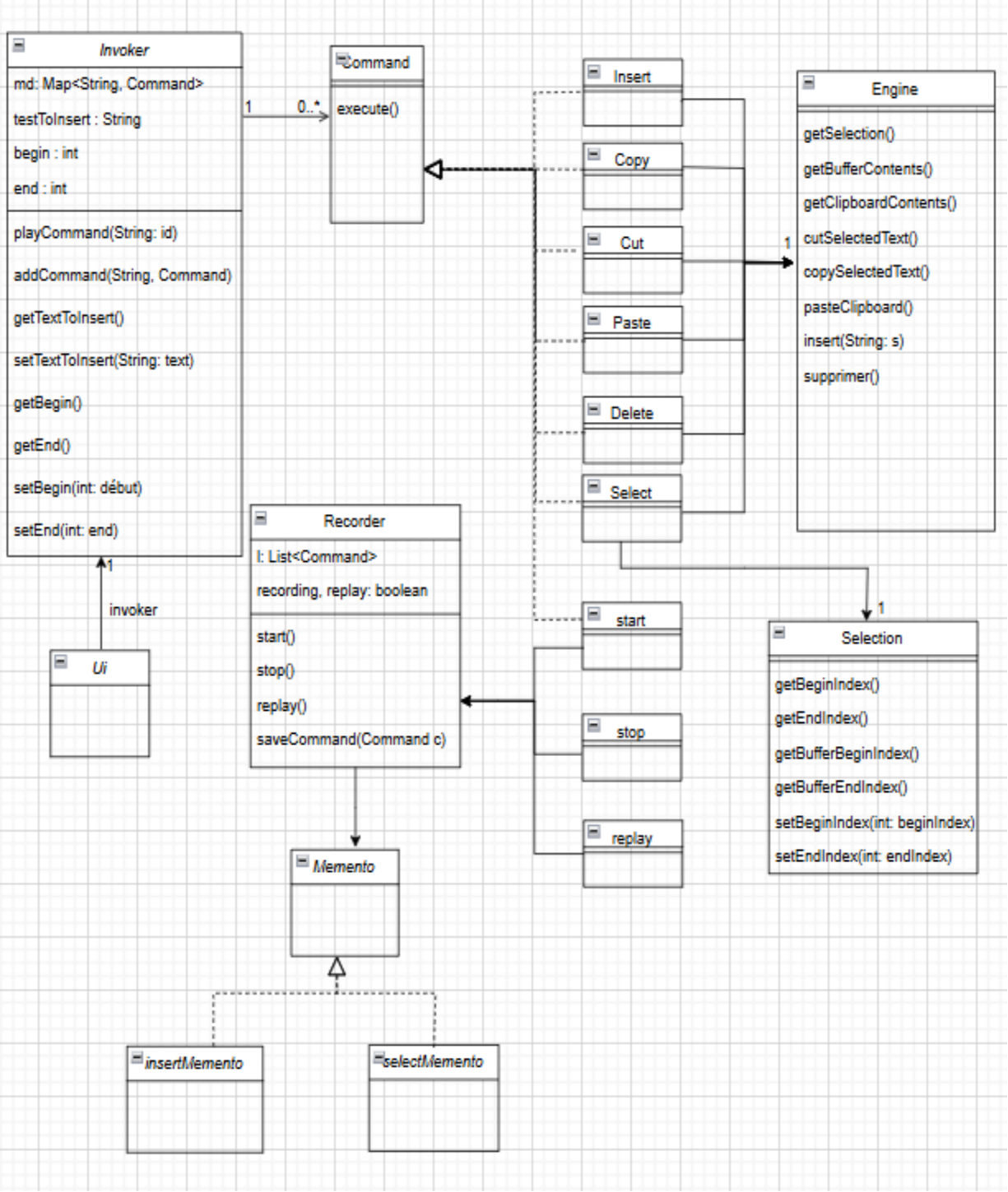


Diagramme de Classe : V3

UML représentant les interfaces comme Invoker, Engine, Selection, Command, Recorder, memento et leurs commandes concrètes



3. Évolution des Versions

Version 1 : Fonctionnalités de Base

- **Fonctionnalités implémentées**:
    - Gestion du buffer.
    - Insertion de texte.
    - Sélection et remplacement de texte.
    - Copier, couper, coller avec gestion du presse-papier.
- **Test et Validation**:
    - Tests unitaires sur chaque commande (e.g., InsertCommand, CutCommand, PasteCommand).
    - Scénarios de test pour valider les transitions d'état dans le buffer.

Version 2: Enregistrement et Relecture

- **Fonctionnalités ajoutées**:
    - Enregistrement des commandes exécutées.
    - Relecture des commandes dans l'ordre d'exécution.
- **Choix techniques**:
    - Stockage des commandes dans une liste pour rejouer les actions.
    - Utilisation d'une interface textuelle pour visualiser la relecture.

Version 3 : Annulation et Rétablissement

- **Fonctionnalités ajoutées** :
    - Annulation de la dernière commande (« undo »).
    - Rétablissement d'une commande annulée (« redo »).
- **Choix techniques** :
    - Piles distinctes pour les actions effectuées et annulées.
    - Intégration avec le patron Memento pour la restauration des états.

4. Synthèse des Tests

Les tests unitaires ont été réalisés à chaque étape pour valider le bon fonctionnement des commandes et des états :

- Cas de test pour la gestion du buffer (e.g., insertion, remplacement).
- Scénarios pour valider l'enregistrement et la relecture.
- Validation des piles d'annulation et de restauration pour les opérations « undo » et « redo ».

5. Lancer les Différentes Versions

Prérequis

- Java 8+
- JUnit 5
- IDE compatible (Eclipse, IntelliJ, etc.)

Commandes pour chaque version

1. **Version 1**:
    - Pour la version 1 je n'ai pas mis en place l'interface textuelle
2. **Version 2**:
    - Lancer Editor.java pour tester l'enregistrement et la relecture.
3. **Version 3**:
    - Lancer Editor.java pour tester les fonctions « undo » et « redo ».

6. Commandes Utilisateur

- **insert** : Insérer du texte.
- **cut** : Couper la sélection.
- **copy** : Copier la sélection.
- **paste** : Coller le texte du presse-papier.
- **Selection** : sélectionner une portion de texte.
- **start** : Démarrer l'enregistrement.
- **stop** : Arrêter l'enregistrement.

- **replay** : Rejouer les commandes enregistrées.
- **undo** : Annuler la dernière commande.
- **redo** : Rétablir une commande annulée.
- **Exit** : quitter l'éditeur

Ce rapport fournit une vue complète de la conception et du développement du mini-éditeur de texte, ainsi que des instructions pour l'exécution et la validation. Les sources Java et les cas de test sont disponibles dans le dépôt avec la Javadoc générée pour une documentation détaillée.