ScalaTextEditor

Rapport Mini-Editeur

Maxime Pauvert & David Perrai

8 décembre 2014

Table des matières

Résumé

Dans le cadre du cour de Génie logiciel il nous a été demandé de mettre en oeuvre un projet de mini-editeur de texte. Ce projet a pour but de mettre en place les connaissances apprises durant le cours. Ainsi nous avons développé le mini-editor àà l'aide d'une modélisation UML pour la conception en amont et au cour du développement, avec la mise en place de patrons de conception. Nous vous présenterons le programme et ses fonctionnalités puis la conception de l'architecture à travers les diagrammes UML et enfin nous expliciterons les différents patrons et leurs rôles dans le programme ainsi que certains choix d'implémentation.

1 Présentation de ScalaTextEditor

ScalaTextEditor est un mini-editeur de texte en ligne de commande, l'utilisateur a comme actions disponible :

- Ecriture de texte.
- Déplacement du curseur.
- Supression du texte.
- Copie dans un presse-papier d'une sélection effectué sur le texte.
- Collage d'une sélection du presse-papier.
- Revenir en arrière sur les dernières actions effectués.

2 Analyse du domaine

Pour concevoir cette éditeur de texte nous avons identifié dans un premier temps les concepts principaux du programme :

1. l'espace de travail de l'utilisateur est le coeur de l'application où sont éxécutées des actions.
2. un curseur simple permettant à l'utilisateur de ce déplacer dans le texte
3. un curseur de selection permettant de sélectionner un passage du texte
4. le presse-papier permettant à l'utilisateur d'enregistrer une sélection
5. les commandes que l'utilisateur pourra effectuer (écriture, effacement, copie, collage...)

A l'aide de cette analyse nous avons extrait les différentes classes composant le système. Pour le comportement et l'architecture de notre programme nous avons utilisé différents patrons de conception. (voir l'architecture générale sur le projet papyrus).

3 Conception détaillée

3.1 Patron de conception : Observer

Le patron de l'observer sert dans ce cadre à mettre à jour "l'espace de travail" (*Workspace*) de l'utilisateur (*UI*) lorsqu'il effectue des modifications dans le texte. La classe UI est donc un observeur qui observe la classe observable workspace qui la notifie lors de son changement.

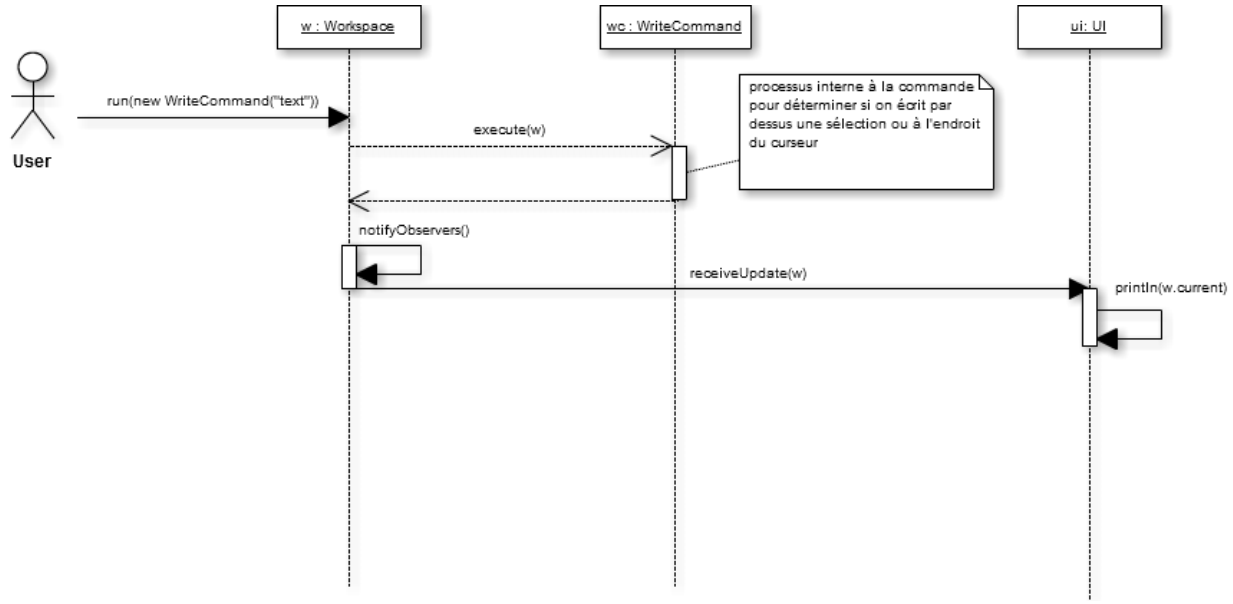

FIGURE 1 – diagramme de Séquence de l'observer

Dans ce diagramme de séquence l'utilisateur execute une commande d'écriture (*WriteCommand*) en remplacement d'une sélection ou à l'endroit de son curseur. Le workspace est alors mis à jour et contient désormais le texte écrit par l'utilisateur. L'espace de travail met ensuite à jour l'interface de l'utilisateur en lui notifiant de son changement.

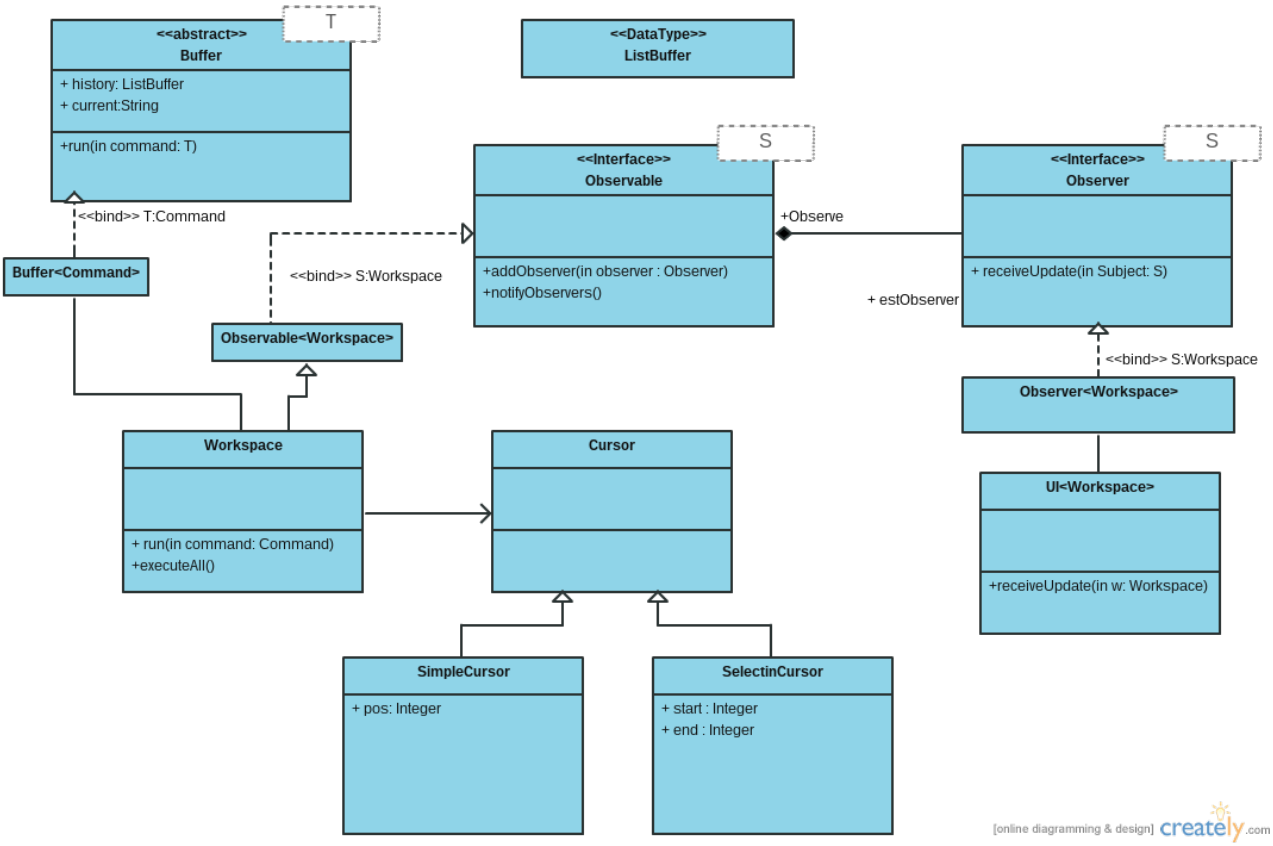

FIGURE 2 – diagramme de classe de l'observeur

3.2 Patron de conception : Commande

Le patron de conception commande est utilisé dans ce contexte pour l'éxecution des actions (les commandes) de l'utilisateur à partir de "son espace de travail" (*workspace*). Il sert entre aute à conserver une sélection copier dans le presse-papier (*clipboard*) lors de l'éxecution d'une commande de type copie ou encore à coller un contenu du presse-papier à un endroit du texte (par dessus une sélection ou à la position du curseur)

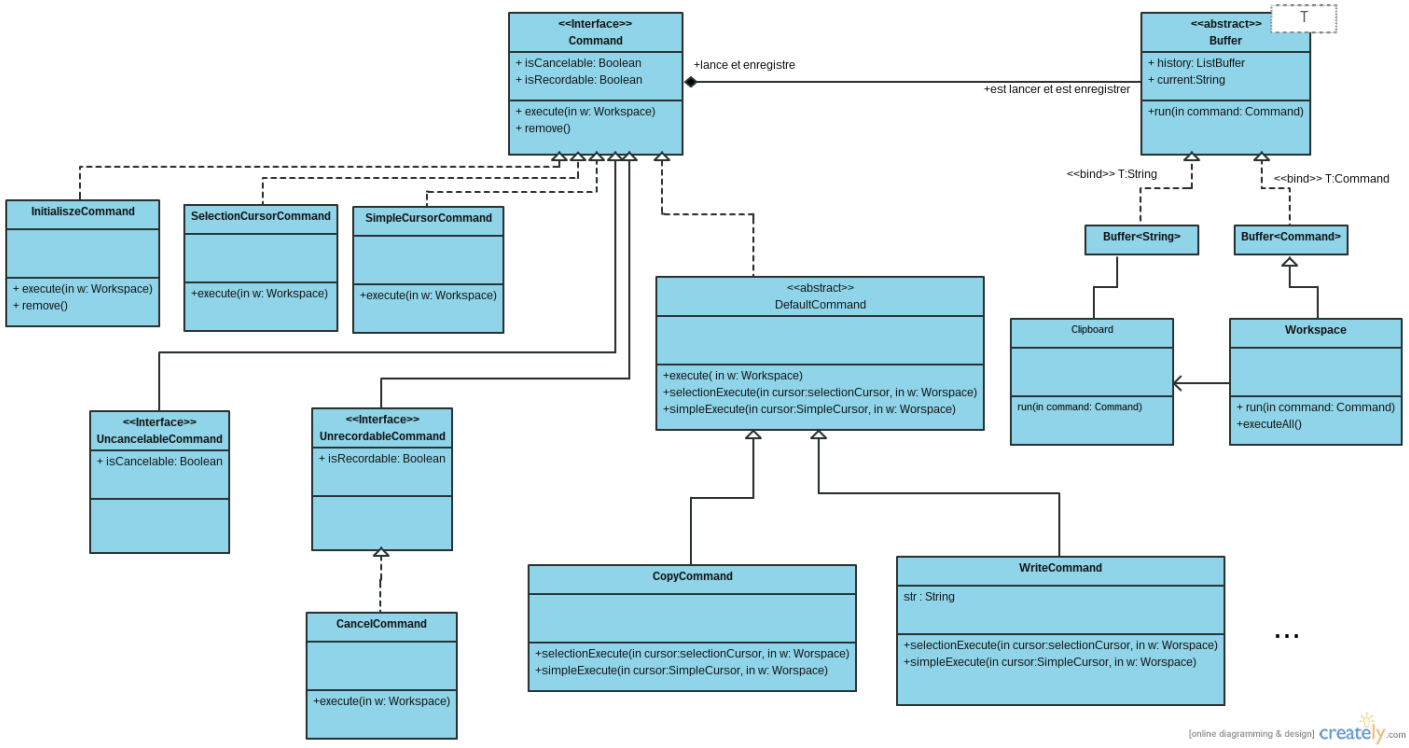

FIGURE 3 – diagramme de classes du patron Commande

A noter que les "..." représentent d'autres commandes (la commande *couper*, *supprimer*, *coller*) mais qui n'apportent rien dans la modélisation et pour la compréhension. Cependant ces classes sont présentes dans le diagramme de l'architecture générale du projet papyrus.

3.3 Patron de conception : Composite

Le patron de conception composite permet de composer les commandes entre elles car certaines commandes sont contruites à partir d'autres commandes.

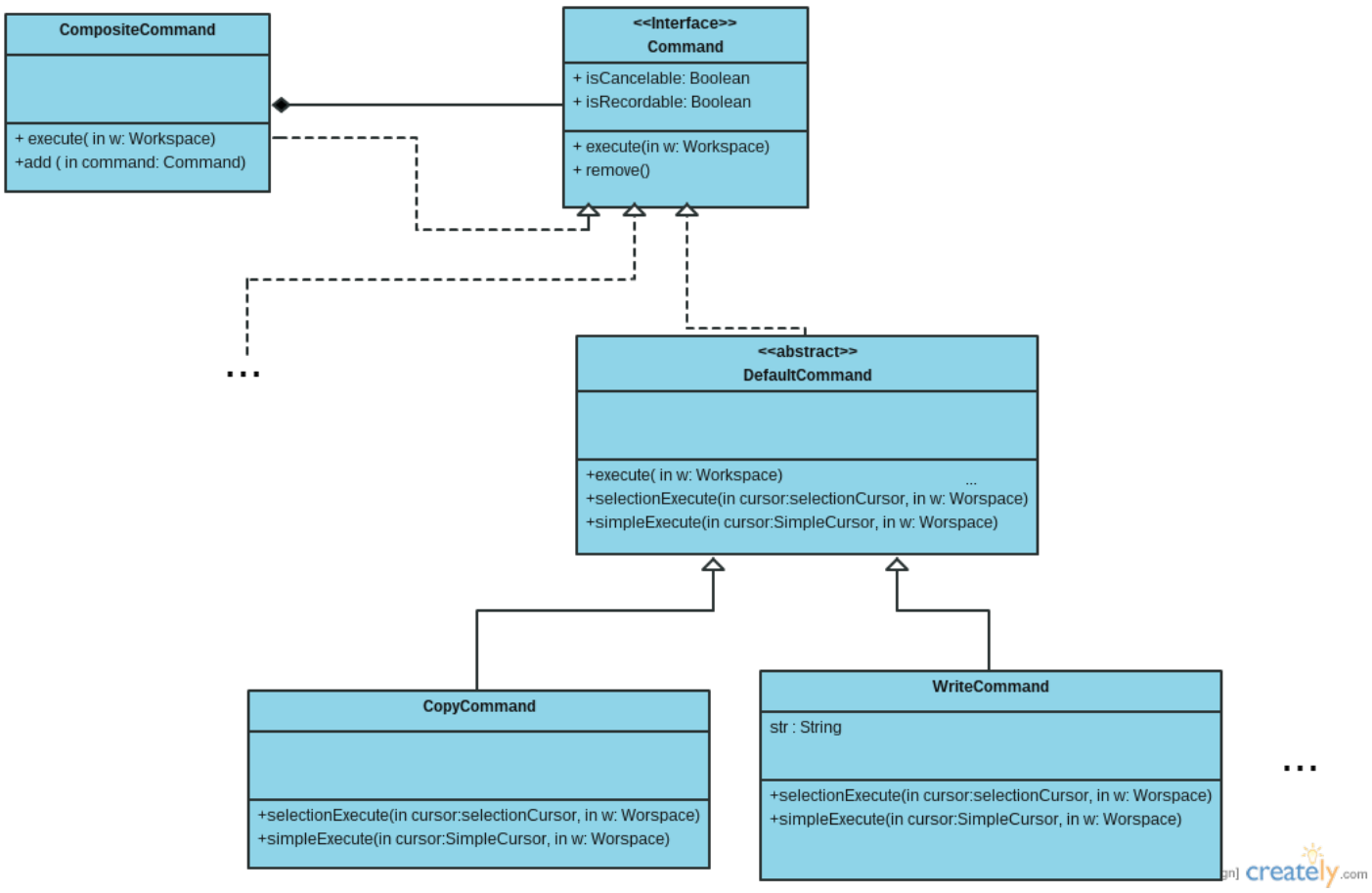

FIGURE 4 – diagramme de classes du patron composite

Comme pour le diagramme précédent la présence des "..." représentent d'autres classes qui n'apportent rien à la compréhension.

4 Choix d'implémentation

4.1 Commande d'initialisation

La commande *InitializeCommand* est une commande particulière. Elle permet d'initialiser le système en mettant le curseur à une position initialise de 0, c'est à dire au début d'une ligne. En outre elle initialise le texte courant de l'espace de travail à vide. Cette commande est executé à l'instanciation de l'espace de travail (*Worspace*).Si elle est réexcuté, le texte courant sera réinitialisé et le curseur sera replacé en début de ligne. La commande sera alors conservée dans l'historique des commandes et pourra être annulée. Du fait qu'elle place le curseur en début de ligne il n'est pas possible d'executé une commande de suppression (*DeleteCommand*) dans le cas échéant une exception sera levé pour prévenir l'utilisateur.

enregistrable

4.2 Commande d'annulation

La commande d'annulation est une autre commande particulière. Elle permet d'annuler une ou plusieurs commandes executées au-paravant et ce jusqu'à la dernière commande enregistrée dans l'historique des commandes. Etant une commande qui annule une autre commande elle ne doit pas être enregistrable dans l'historique, dans le cas échéant on ne pourrait annuler que la dernière commande qui serait remplacée dans l'historique par une commande annulée qui serait elle même remplacée par une commande annulée dans le cas d'une autre annulation.

Pour évité ce comportement nous avons mis en place d'un attributs booléen (isRecordable) qui sera interprété lors de l'execution de commandes pour éviter l'enregistrement et donc une annulation d'une commande d'annulation.

Dans un premier temps nous avions aussi rajouté un autre attribut booléen pour géré l'annulation (isCancellable) cependant nous avons déterminé que si une commande n'était pas enregistrée dans l'historique elle ne pourrait pas être annulée car seule les commandes présentent dans l'historique peuvent l'être. Le comportement obtenu avec l'attribut isRecordable inclue donc le comportement de l'attribut isCancellable. Nous avons tout de même conservée l'attribut dans le cas ou l'on souhaiterai rajouter une commande qui serait enregistrable dans l'historique mais qui ne serait pas annulable.