Projet : Editeur de texte

Paul Leclercq & Sacha Fernandez-Soltane

M1 MIAGE- Groupe B

Enseignant : Adrien Le Roch

Sommaire

Choix personnel de conception et tests

Le projet, étant relativement important, a été conçu avec un souci de clarté, avec des choix détaillés pour chaque version (V1, V2, V3).

Version 1 : Mise en place du projet

Pour la première version, aucune décision particulière n'a été prise en matière de conception. Les classes ont simplement été regroupées dans des paquets. Par exemple, les commandes telles que Cut, Copy, Paste ont été regroupées dans un paquet "concretecommand", l'invoker dans un paquet "invoker", et le receiver dans un paquet "receiver". Étant donné qu'il n'y a qu'une seule classe de test, la création d'un paquet de tests n'a pas encore été mise en place, mais cela sera réalisé ultérieurement.

Version 2 : Replay

Pour rester dans la même logique, les classes ont été regroupées dans des paquets tels que "memento", "caretaker", etc. Dans cette version, les choix de conception sont moins personnels, car la structure était déjà bien indiquée dans le diagramme de classe. L'effort principal a été consacré à la réflexion sur le code.

Version 3 : Undo/Redo

Dans cette itération, plusieurs possibilités se sont présentées. Dans notre cas, nous avons opté pour la première solution, jugée assez accessible et surtout en continuité avec la logique de la V1. Cela implique l'ajout de deux concreteCommands, "undo" et "redo", qui appelleront la classe "UndoManager". De plus, un nouveau memento, "EditorMemento", sera nécessaire pour stocker le buffer ainsi que les index. Enfin, des modifications seront apportées à l'interface "Engine" pour inclure deux méthodes, "setMemento()" et "getMemento()".

Tests :

Comme illustré dans la capture d'écran, les tests ont été organisés selon les commandes concrètes. CopyPaste et CutPaste sont regroupés car ces commandes sont fréquemment associées, tandis que les autres commandes telles que Delete, Insert, Redo et Undo sont également présentes. Il n'y a pas de classe dédiée à Replay car ces tests sont répartis dans différentes classes.

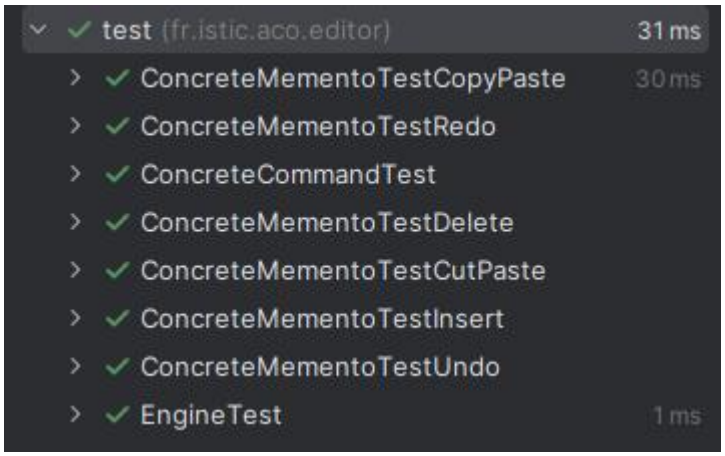
Au sein des classes, nous essayons de garder une logique pour que les tests soient faciles à trouver et lire, le but était de couvrir un maximum de possibilités sans pour autant se noyer dans un torrent de tests.

Synthèse résultat de test :

Voici le test coverage :

Element	Class, %	Method, %	Line, %
fr.istic.aco.editor.test	100% (8/8)	100% (57/57)	100% (566/566)
ConcreteCommandTest	100% (1/1)	100% (11/11)	100% (104/104)
ConcreteMementoTestCopyPaste	100% (1/1)	100% (4/4)	100% (55/55)
ConcreteMementoTestCutPaste	100% (1/1)	100% (3/3)	100% (41/41)
ConcreteMementoTestDelete	100% (1/1)	100% (2/2)	100% (26/26)
ConcreteMementoTestInsert	100% (1/1)	100% (6/6)	100% (53/53)
ConcreteMementoTestRedo	100% (1/1)	100% (9/9)	100% (100/100)
ConcreteMementoTestUndo	100% (1/1)	100% (11/11)	100% (134/134)
EngineTest	100% (1/1)	100% (11/11)	100% (53/53)

Et voici le résultat des tests :



Comment lancer les différentes versions :

A noter que le projet est en java 21 et qu'il utilise certaines méthodes de Java 21 (removeFirst() par exemple) ainsi il vous faudra au minimum Java 21 pour lancer le projet.

Pour lancer le projet , suivez ces étapes : commencez par copier le lien du dépôt Git (https://gitlab.istic.univ-rennes1.fr/sfernandezso/editor-2023-sacha-fernandez-soltane-paul-leclercq) du projet sur GitLab.

Ensuite, ouvrez IntelliJ et sélectionnez "Check out from Version Control" dans le menu d'accueil. Choisissez Git et collez le lien du dépôt dans le champ approprié.

Appuyez sur "Clone" pour télécharger le projet sur votre machine locale. Une fois le projet cloné, ouvrez-le dans IntelliJ en sélectionnant le dossier du projet.

Assurez-vous de configurer correctement le JDK (Java 21 au minimum). Enfin, pour lancer le projet, trouvez la classe principale (ici ihm) dans la structure du projet, cliquez dessus avec le bouton droit de la souris, et sélectionnez "Run" à partir du menu contextuel ou cliquez simplement sur la flèche verte (Run) dans la barre d'outils d'IntelliJ.

Cela déclenchera l'exécution du programme et affichera les résultats dans la console d'exécution.

Pour changer de version, il suffit de changer de branche avec la commande git checkout «nom_branche» et relancer la classe ihm.

Syntaxe des commandes :

Tout d'abord, lorsque vous lancer l'ihm vous pouvez apercevoir un récapitulatif des commandes si jamais vous les oublier,donc pas d'inquiétude.

Voici un récapitulatif des différentes commandes :

b : voir les infos de l'éditeur
i : insérer texte
s : sélectionner texte
c : copier sélection
v : coller
x : couper sélection
d : supprimer sélection
start : démarrer l'enregistrement
stop : stopper l'enregistrement
replay : rejouer l'enregistrement
undo : annule la commande précédente
redo : annule le undo précédent
exit : sortie de l'éditeur

Et dans le code voici les commandes :

```
mapCmd.put("changeSelection", new ChangeSelection(engine.getSelection(), invoker,
recorder,undoManager));
mapCmd.put("insert", new Insert(engine, invoker, recorder,undoManager));
mapCmd.put("copy", new Copy(engine, recorder,undoManager));
mapCmd.put("delete", new Delete(engine, invoker, recorder,undoManager));
mapCmd.put("cut", new Cut(engine, invoker, recorder,undoManager));
mapCmd.put("paste", new Paste(engine, invoker, recorder,undoManager));
mapCmd.put("start", new Start(recorder));
mapCmd.put("stop", new Stop(recorder));
mapCmd.put("replay", new Replay(recorder));
mapCmd.put("undo", new Undo(invoker,undoManager,engine));
mapCmd.put("redo", new Redo(invoker,undoManager,engine));
```

Comme vous pouvez le voir les commandes ne sont pas bien compliqués, si c'est un insert dans le code on fait insert sinon pour l'utilisateur il fait «i», «s» pour une sélection ect...