[![Review Assignment Due Date](https://classroom.github.com/assets/deadline-readme-button-24ddc0f5d75046c5622901739e7c5dd533143b0c8e959d652212380cedb1ea36.svg)](https://classroom.github.com/a/G801g4IR) = Gestionnaire de Fichiers en Commandes en Ligne avec Annotations

MOUSSAOUI, Iness, 22006171,51

DJEBBOUR , Elias ,22108396, 50

https://github.com/ensYeh/miniprojet-grp-51_50

L'objet de ce mini-projet est de développer un gestionnaire de fichiers en commande en ligne. Le gestionnaire de fichier s'appuie sur le système de fichier de votre système d'exploitation. Le gestionnaire de fichier doit offrir la possibilité d'annoter les éléments contenus dans un répertoire (i.e.fichiers, répertoires). Vous devrez respecter les contraintes fonctionnelles et techniques mentionnées ci-dessous.

IMPORTANT

Vous respecterez les contraintes suivantes:

- Ce mini-projet est à réaliser en Java par groupe de 2 étudiants.
  - les 2 étudiants doivent coopérer à travers un même git. Un des deux étudiants devra permettre à l'autre d'utiliser son git.
- Il devra comporter une documentation dans ce git.
  - La documentation devra décrire l'usage de l'application (*manuel utilisateur*) ainsi que la conception du jeu (*manuel technique*).
  - La documentation doit être un fichier .md ou .adoc et accessible depuis le `git`
- Vous utiliserez `git` en effectuant des commits réguliers comportant des messages informatifs. L'usage des "pull requests" est également fortement conseillé, montrant ainsi votre collaboration au sein du groupe.
- Le *build* sera assuré par Maven et plus précisément Maven wrapper (déjà intégré dans le projet). Aucune manipulation en dehors de Maven ne devra être nécessaire.
- La version de Java à utiliser est la version 17.
- Le *build* devra intégrer `checkstyle` pour la vérification des règles de codages Google. Le projet devra donc les respecter. `checkstyle` devra être exécuté automatiquement durant la phase `validate` du cycle de vie par défaut.
- Des tests unitaires JUnit 5 (version *5.9.1*) devront être disponibles pour la plupart des méthodes développées.
- Un outil de *Code Coverage* devra être intégré au *build*
- Les fonctionnalités du langage Java devront être utilisées au mieux (POO, exceptions, librairie de collections, I/O, ...).
- L'application devra pouvoir être exécutée à partir d'un `jar` incluant

toutes les dépendances.

Description du gestionnaire de fichiers

- L'interface proposera une visualisation du contenu d'un répertoire en mode texte. A chaque élément du répertoire (**ER**), on associera un numero (**NER**) permettant de le designer lors de l'utilisation d'une des commandes définies ci-dessous.
- Les actions de l'utilisateur seront saisies au clavier sous la forme `[<NER>] [<commande>] [<nom>]`. Les crochets signifient "optionnel"
  - Par exemples:
    - `3 cut` ; efface le troisième fichier.
    - `3` ; l'utilisateur désigne le troisième élément du répertoire.
  - Si l'utilisateur ne place pas de NER, c'est le dernier NER utilisé qui sera utilisé pour une commande qui en nécessite. Exemple `visu`
  - Les commandes du gestion de fichiers à implémenter sont:
  - `[<NER>] copy`
  - `past` ; si l'élément existe, alors le nom du nouvel élément sera concaténé avec "-copy"
  - `[<NER>] cut`
  - `..` ; pour remonter d'un cran dans le système de fichiers
  - `[<NER>] .` ; pour entrer dans un répertoire à condition que le NER désigne un répertoire. Exemple "4 ."
  - `mkdir <nom>` ; pour créer un répertoire
  - `[<NER>] visu` ; permet de voir le contenu d'un fichier texte. Si le fichier n'est pas de type texte, vous afficherez sa taille.
  - `find <nom fichier>` ; Recherche dans toutes les sous répertoires du répertoire courant, le(s) fichier(s) et les affiches.
- L'application permettra à l'utilisateur d'annoter un ER
  - l'annotation consiste à ajouter ou retirer un texte associé à un ER.
  - Par exemples:
    - 3 + "ceci est un texte" ; le texte est ajouté ou concaténé au texte existant sur l'ER
    - 3 - ; retire tout le texte associé à l'ER 3

Description de l'application à réaliser

- L'interface proposera une visualisation du répertoire courant en mode texte en quatre parties. La partie:
  1. présente le cheminement depuis la racine du système de fichier de votre système de fichier.
  2. affiche la note associée à l'élément courant NER si elle existe. L'élément courant correspond

à la désignation du NER par l'utilisateur ou désigné dans la dernière commande utilisée. Exemple `3 copy`, 3 devient l'élément courant.

3. affiche les ER du répertoire avec leurs NER
4. présente un prompt invitant l'utilisateur à saisir une des commandes présentées ci-dessus.

- Une interface spécifique pourra être proposée pour des situations particulières (visualiser le contenu d'un fichier, une aide, ...).
- Toutes les notes associées aux éléments du répertoire courant seront stockés dans un seul fichier appelé "notes" dans le répertoire courant. Ce fichier peut contenir des objets "sérialisés" ou d'autres formats.

Références

- Article Wikipedia Gestionnaire de fichiers
- Quelques bibliothèques : JAnsi (couleur dans un terminal), JLine (gestion des saisies)

Manuel utilisateur

> À compléter : Comment l'utilisateur peut savoir quelle commande taper pour un élément du répertoire désigné ? Quelles sont les mises à jours du fichier des annotations à effectuer en fonction des types de commandes ? Y a t il des bibliothèques Java qui permettront de prendre en charge la visualisation d'une image png si l'utilisateur veut l'afficher ? Quelles sont les commandes qui seraient utiles de rajouter ? Quelles améliorations peut on envisager pour rendre l'usage de l'interface clavier plus souples/efficaces pour l'utilisateur ? Quelles évolutions peut-on envisager ?

Manuel technique

Compiler le projet

Sous Linux

```
$ ./mvnw package
```

Sous Windows

```
> mvnw.cmd package
```

Exécuter l’application

```
$ java -jar target/explorer-1.0.jar
```

> À compléter : Comment consulter le rapport de couverture de code par les tests ? Quelles bibliothèques ont été utilisées et pourquoi ? Quel est le rôle des différentes classes ? Quels traitements sont réalisés pour gérer une commande saisie par l’utilisateur ? Donnez un exemple. Quelles améliorations peut-on envisager ?