POO-Java : Projet “Images & métadonnées avec Java”

Eagle Extractor

Par Kenan Ammad et Gauthier Defrance Groupe C4

Introduction

Le projet de POO-Java intitulé : *“Images & métadonnées avec Java”* a pour objectif initial de pousser les étudiants à la conception d’une petite application travaillant avec des images et leurs métadonnées. C’est une application avec beaucoup de fonctionnalités qui est demandée. Qui par ailleurs doit exister en deux modes : un CLI et un GUI.

Il faut que l’application réponde à des fonctionnalités minimales telles que le fait de pouvoir accéder aux statistiques et informations d’une image ou d’un répertoire, de lister tous éléments présents dans un répertoire avec une arborescence inconnue. Mais également comporter un système de recherche d’images.

C’est avec toutes ces conditions en tête que nous avons donc programmé “Eagle Extractor”.

Index

Numéro Page : Titre

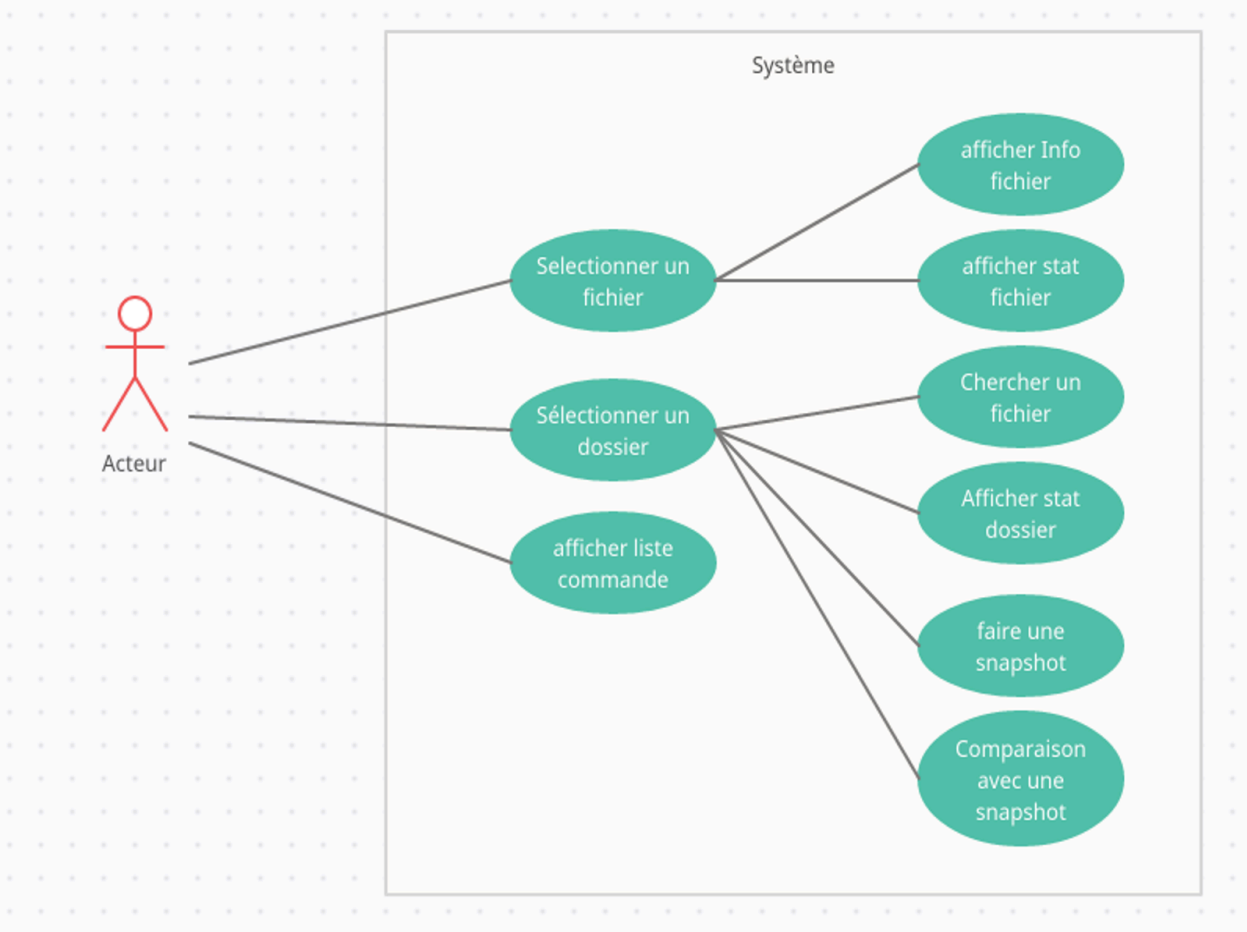
Diagramme de Gantt

Voici ci-dessous le diagramme de Gantt, qu'on a créé au tout début du projet. Afin de mieux planifier l'organisation.

Le projet se découpe en trois grandes parties : Le Noyau (ou Core), le CLI et le GUI. Kenan et Gauthier se sont chargés ensemble du Noyau. Une fois le noyau presque terminé (~80%), Kenan s'occupe du GUI dont il s'est chargé en grande partie, tandis que Gauthier programme le CLI. Bien sûr, Gauthier et Kenan ont beaucoup communiqué ensemble au fur et à mesure que le projet avancé côté CLI et GUI afin de s'entraider pour utiliser au maximum les capacités de chacun. Le noyau a subi des modifications au fur et à mesure des besoins provenant du GUI et CLI.

Diagramme de cas d'utilisation

Voici ci-dessous le diagramme de cas d'utilisation.

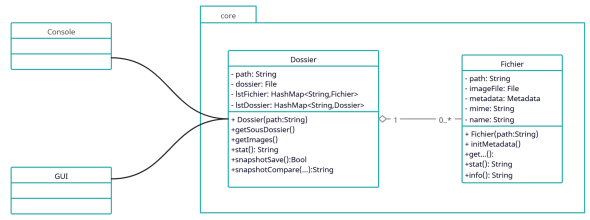
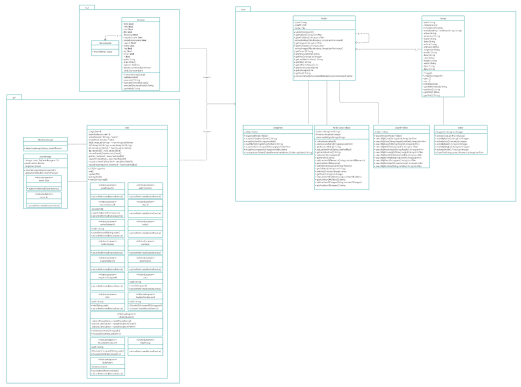


Autant sur le CLI et le GUI, l'utilisateur peut effectuer les mêmes actions. La différence réside dans le fait que dans l'un, on utilise une souris et dans l'autre un terminal.

Diagramme de classe

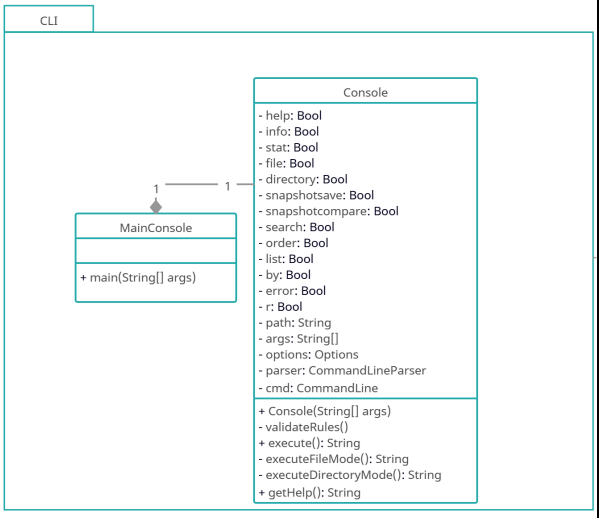
Le diagramme de classe est disponible dans le Zip envoyé.

Ci dessous les diagrammes de classe

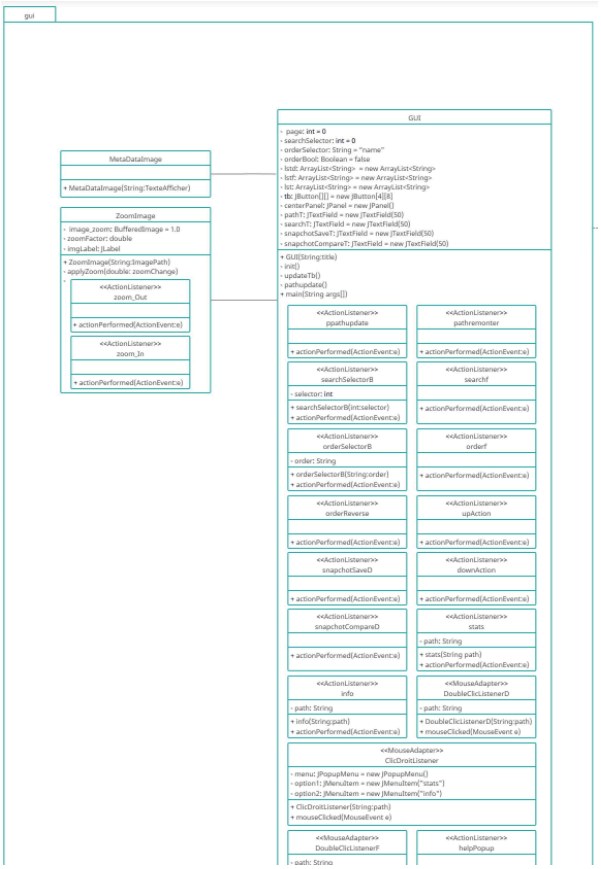
Au départ	Après
	

On remarquera que la structure de départ du noyau est restée avec le rapport Folder-Image.

Regardons de plus près le CLI :

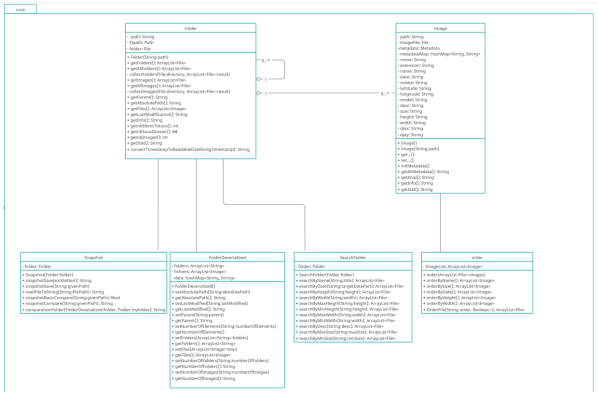
	On a ici le Main qui se trouve dans la classe MainConsole, et qui crée un objet Console se chargeant de comprendre les arguments passés. Un traitement relatif aux arguments est alors effectué.

Regardons ici de plus près le GUI :



On a ici la classe GUI qui contient la classe Main mais aussi de nombreuses sous classes qui sont des boutons, actionneurs, zone de texte en tout genre.
Il y a d' autres classes également qui sont MetaDataImage et ZoomImage. Respectivement, une fenêtre affichant du texte avec des barres de scroll et une fenêtre de visualisation d'images avec la possibilité de dé/zoomer sur l'image.

Enfin regardons le "Core" :



Il se découpe donc en deux classes principales qui sont la classe Folder et Image. Qui servent à représenter les dossiers et images sur votre ordinateur.
Les classes Snapshot, FolderDeserialized, SearchFolder et order servent à effectuer des opérations de traitement pour plus de flexibilité du code.

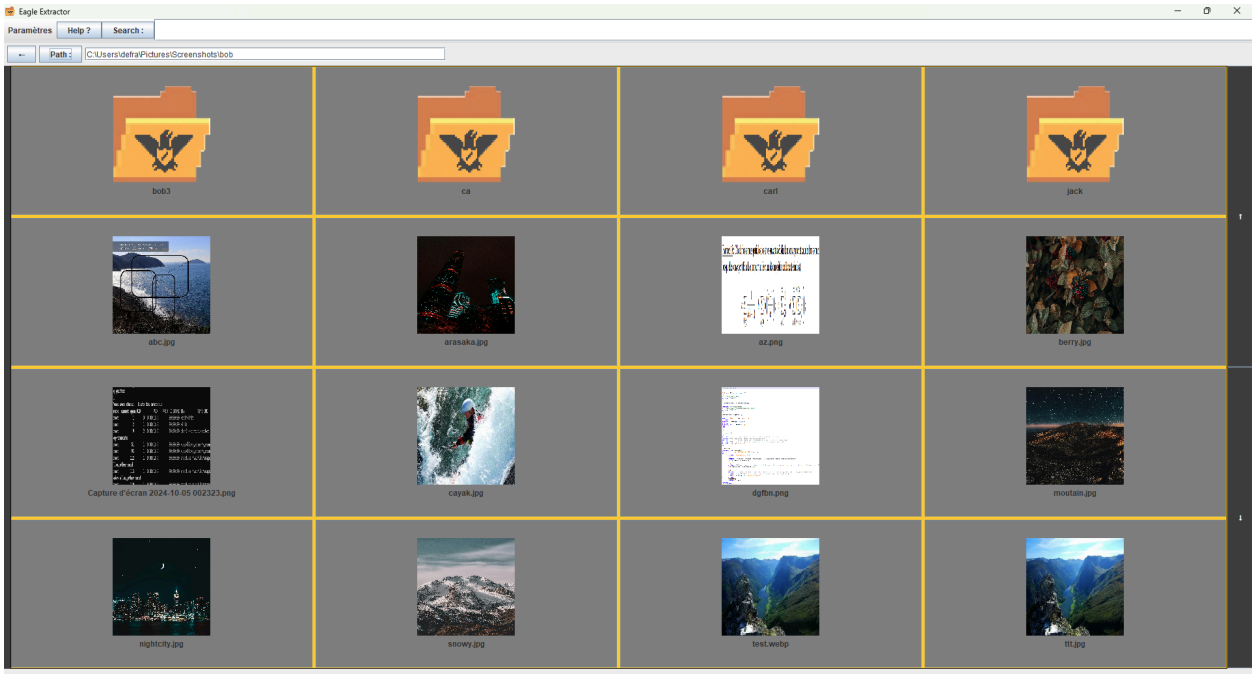
Développement

Analyse du projet dans son état actuel :

1) Le Noyau
    a) Utilise deux classes principales Folder et Image, qui servent à la représentation des dossiers et images. Afin de collecter les métadonnées des images, la classe Image utilise la librairie : “metadata extractor DrewNoakes” https://github.com/drewnoakes/metadata-extractor
    b) Les données XMP et EXIF sont également récupérées, via cette même librairie.
    c) La classe Snapshot et FolderDeserialized servent pour la sauvegarde de l’état d’un dossier et pour sa comparaison. La sauvegarde s’effectue avec un JSON qui est lisible par les humains. On utilise la librairie “*Jackson*” pour la sauvegarde. https://github.com/FasterXML/jackson
    d) Tout le long du projet on travaille avec dossiers et images en tant que Chemin absolue. On a donc utilisé la classe File de Java pour cela.
    e) Le système de recherche et listage des images se fait récursivement via une recherche dans les sous-dossiers.
2) Le CLI
    a) La classe Console utilise la librairie “*common cli*” pour simplifier le traitement des arguments. https://github.com/apache/commons-cli
    b) Il possède deux modes de fonctionnement : Le FileMode et le DirectoryMode qui sont déterminés grâce aux objets File qui permettent de déterminer si un Chemin existe, s' il correspond à une Image ou à un Dossier.
    c) Il y a également une méthodes validateRules() qui sert à vérifier si certaines pré conditions sont remplies.
3) Le GUI
    a) Le GUI fonctionne essentiellement à l’aide de la librairie Swing de Java. Mais on utilise également l’extension provenant de “*Twelves monkey*” pour les webp qui s’attache alors à imageio. https://github.com/haraldk/TwelveMonkeys
    b) Le gui a été pensé de pour être user friendly. Il rappelle l’explorateur de fichiers typique.

    c) L'affichage des dossiers et des images fonctionne à l'aide d'un GridLayout pouvant comprendre 32 éléments et de trois ArrayList.
    Une comprenant tous les dossiers, une comprenant toutes les images et la dernière comprenant tout ce qui doit être affiché dans le GridLayout.
    d) Seulement 32 éléments peuvent être affichés en même temps pour choisir les 32 éléments dans la liste on utilise deux boutons + et -. Le reste de l'interface est principalement constitué de jbouton et jmenu.

Voici ci dessous une image du GUI



Point positifs et négatifs

Les plus :

- Le programme a été bridé aux 4 formats d'images : GIF, WEBP, PNG et JPG. Il serait facile d'implémenter des formats supplémentaires grâce à la flexibilité du code.
- Le CLI est capable de parcourir la racine afin de chercher toutes les images présentent (et compatibles) sur la machine relativement vite grâce aux optimisations du code, qui évitent des initialisations inutiles et de la redondance.
- Interface ergonomique et User Friendly avec une prévisualisation des images pour le GUI.
- Il est facilement possible d'explorer l'arborescence de l'ordinateur grâce aux boutons permettant de la remonter, et les dossiers apparaissent qui permettent de s'y enfoncer.
- En plus des fonctionnalités de base demandées, la possibilité de trier les éléments selon certaines conditions à été implémenté après discussion interne dû à l'utilité et le confort que la fonctionnalité apporte à l'utilisateur.
- Le système de recherche d'éléments par rapport à ses informations à été pensé également flexible, de sorte à pouvoir implémenter facilement, au besoin de nouvelles méthodes de recherches.
- Logo personnalisé pour l'application

Les moins :

- L'application peut prendre un temps conséquent de plusieurs secondes pour traiter une grande quantité d'images (+1000).
- La fonctionnalité de zoom prend plusieurs secondes pour traiter des images en très haute définition (8K,16K) ou quand on zoom beaucoup.
- Les images dans la prévisualisation sont étirées ce qui peut être assez moche et réduit le confort utilisateur.
- Le GUI est très procédural lors de l'initialisation dans le code.
- FolderDeserialized ne devrait pas exister, mais a été conçu pour éviter une récursivité auto appelante.

Améliorations possible

Les points négatifs vu précédemment peuvent être résolus. Le premier, deuxième et troisième point peuvent être résolus en utilisant d'autres librairies plus optimisées, ou bien en faisant appel à un autre langage de programmation pour cette partie spécifique du code. Le quatrième et cinquième point peuvent être résolus en faisant une refonte de certaines parties du code.

Les trois premiers points n'ont pas été faits par manque de temps ou pour respecter les exigences.
Les deux derniers points n'ont pas été fait car leur impact est superficielle sur l'efficacité du code.

Conclusion

Nous estimons que le projet *Eagle Extractor* répond aux attentes demandées. Des améliorations sont envisageables et comme dit de nombreuses fois, le code est flexible. Ce qui lui permettrait de s'adapter à de nouvelles contraintes facilement.

Commentaire de Kenan et Gauthier :

Produire ce projet nous a permis d'enrichir nos connaissances sur les métadonnées et nous a permis de découvrir de nouvelles librairies Java. Nous sommes tous les deux satisfaits de notre projet et avons apprécié le développer. La liberté et à la fois les contraintes imposées par le cahier des charges était également une expérience enrichissante. Les connaissances apprises en CM et TP de Java ont pu être mises en pratique. Afin de travailler en équipe plus efficacement nous avons également approfondi nos connaissances sur de nouveaux outils tels que Github et IntelliJ.