PROJET POO & JAVA-BOP
Licence Informatique 2

Année 2021-2022

# PROJET : METADONNEES DE FICHIERS IMAGES ET STEGANOGRAPHIE


**Projet réalisé par** :

DAMODARANE JEAN-BAPTISTE (22101976)
ELUMALAI SRIGURU (22100203)
L2-I Groupe C

# Table des matières


I. INTRODUCTION ..... 3
1. Aperçu du Rapport et du Projet ..... 3
2. Informations techniques du Projet ..... 3
II. GESTION DU PROJET ..... 4
1. Planification de Projet (avec diagramme de Gantt) ..... 4
2. Répartitions de tâches ..... 4
III. METADONNEES D'UN FICHIER IMAGE ..... 5
1. Implémentation et Packages Utilisés ..... 5
2. Fonctionnement du Programme ..... 5
IV. STEGANOGRAPHIE ..... 6
1. Implémentation / Package utilisés ..... 6
2. Encoder ..... 6
a. Le fonctionnement d'encodage ..... 6
3. Décoder ..... 7
a. Le fonctionnement de décodage ..... 7
4. Schéma récapitulant les fonctionnements d'encodage et de décodage ..... 8
V. DIAGRAMME UML DE L'ENSEMBLE DES CLASSES DU PROJET ..... 9
VI. CONCLUSION ET BILAN DU PROJET ..... 10

# I. INTRODUCTION

## 1. Aperçu du Rapport et du Projet

Dans le cadre d'une réalisation concrète des notions de POO et JAVA abordées au cours du module de **POO&JAVA-BOP**, nous avons eu à réaliser une application en JAVA permettant la gestion de traitement des images PNG et JPEG. Notre application a pour objectif dans un premier temps d'extraire les métadonnées d'une image puis dans un second temps de dissimuler un message dans une image en utilisant la stéganographie.

Afin de mener à bien ce rapport, nous commencerons par présenter les différentes tâches que nous avons pu identifier ainsi que leurs répartitions au sein de l'équipe, ensuite nous expliquerons les différentes étapes de nos solutions qui nous ont permis d'effectuer notre application par le biais de deux grandes parties : la première portera sur le traitement des métadonnées d'une image et la deuxième sur la stéganographie.
Et nous terminerons par faire un bilan de notre travail en portant un regard critique notamment sur l'aboutissement du projet.

## 2. Informations techniques du Projet

Ce projet peut tourner sous les systèmes d'exploitation de Windows et GNU/Linux.
La version Java utilisée est la **JAVASE-11**.
Ce projet est exécutable à partir d'une console, en mode **CLI** (Command Line Interface), avec des commandes et des arguments spécifiques, ou bien en mode **GUI** (Graphical User Interface).

## II. GESTION DU PROJET

La gestion du projet est la première chose que nous avons fait avant même de commencer le projet, car elle permet de répartir les travaux à réaliser entre les membres du groupe et nous a permis d'assurer la stabilité et la continuité du projet. Cela a permis également un meilleur avancement du Projet en surveillant au fur et à mesure son évolution.

### 1. Planification de Projet (avec diagramme de Gantt)

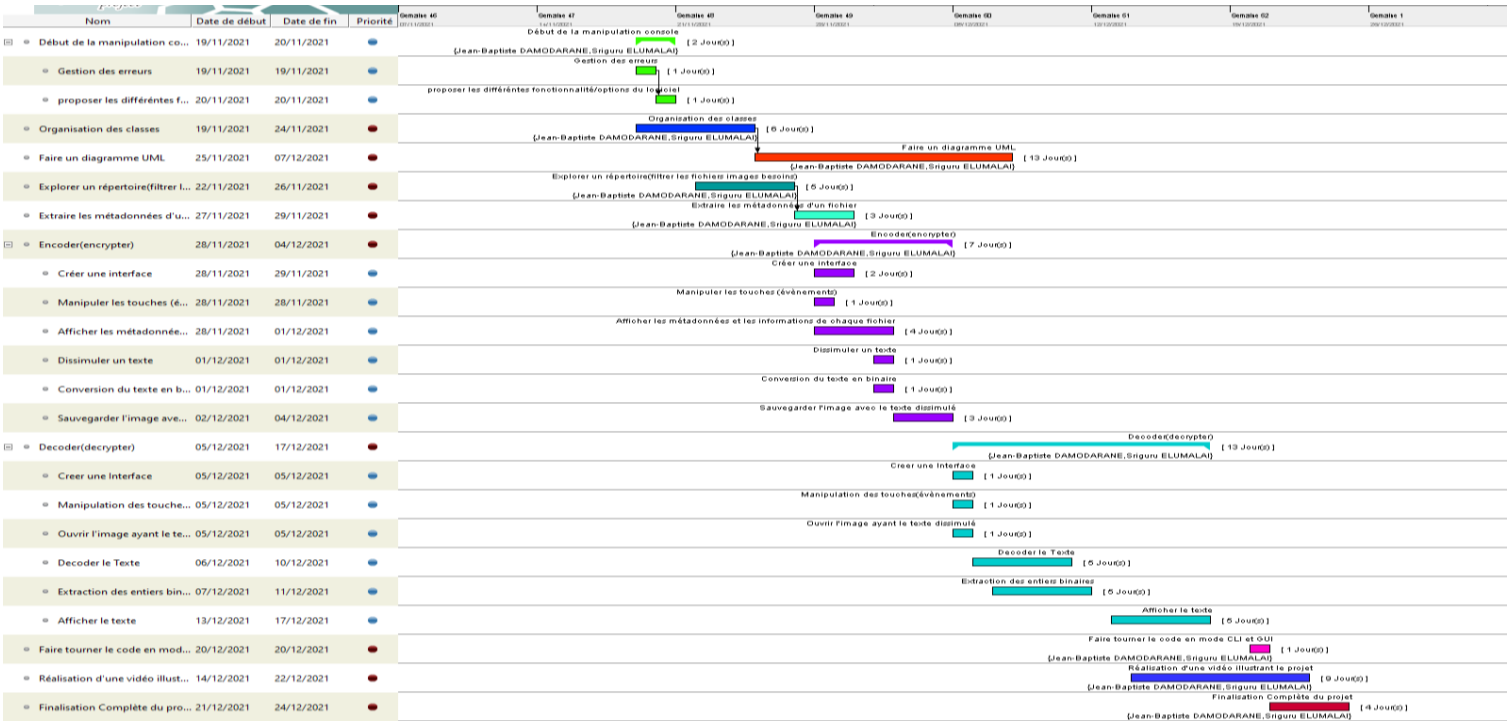



*Figure 1 : Diagramme de Gantt*

Le diagramme de Gantt permet d'avoir une vision plus globale et claire des objectifs du projet, leurs responsables et leurs niveaux d'importance.

Nous avons fait un diagramme de Gantt après avoir étudier les niveaux de faisabilités, pour ne pas avoir du retard.
Toutefois, nous avons pris du retard pendant le passage de la partie métadonnées à la partie Stéganographie, mais nous l'avons rattrapé.

### 2. Répartitions de tâches

Nous avons réparti les tâches en fonctions de nos préférences. Nous avons commencé par l'exploration d'un dossier/répertoire et l'extraction des métadonnées d'une image, pour enfin terminer par encoder et décoder (la stéganographie).

Le diagramme ci-contre indique les pourcentages de travaux effectués sur chaque grande partie du projet.

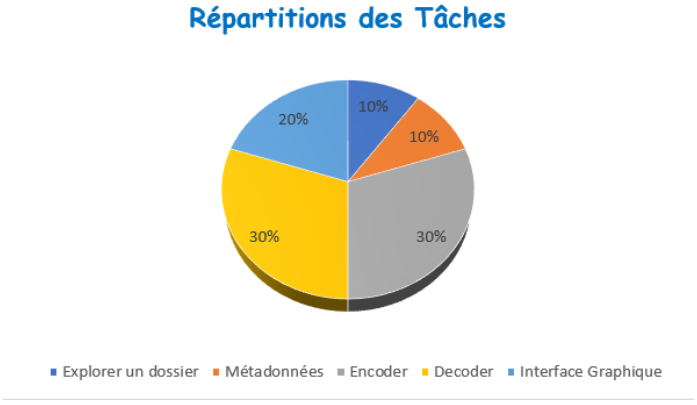



*Figure 2 : Diagramme de Répartition de Tâches*

# III. METADONNEES D'UN FICHIER IMAGE

Les métadonnées sont des informations descriptives sur des documents, des ensembles de données, des images et d'autres fichiers. Dans notre cas, nous nous concentrons uniquement sur la récupération de ces informations à partir d'une image PNG ou JPEG, dont les données sont déjà codées à l'intérieur même de cette image.

Etant donnée qu'il n'existe pas de classes en JAVA qui permettent de réaliser ce genre d'opération, nous avons décidé d'utiliser une bibliothèque tierce qui permet de faire cela. Cette bibliothèque existe en effet au site : http://www.drewnoakes.com/code/exif/.

## 1. Implémentation et Packages Utilisés

### com.drew.imaging.ImageMetadataReader;

La classe ImageMetadataReader dérivée de la classe Object permet d'obtenir les métadonnées de tous les formats de fichiers pris en charge. Si le fichier est un type particulier, il peut préférer utiliser directement le MetadataReader dédié (Ex : JpegMetadataReader pour les fichiers de type JPEG).

### com.drew.metadata.Directory;

La classe Directory dispose d'un ensemble de méthodes qui permettent de récupérer les informations souhaitées.

### com.drew.metadata.Metadata;

La classe Metadata permet d'instancier un objet qui contient plusieurs types de métadonnées comme EXIF, d'un certain fichier.

### com.drew.metadata.Tag;

La classe Tag nous fournit des méthodes permettant de retourner la valeur d'un 'tag' particulier.
Et son instance est spécifique à une donnée particulière.

## 2. Fonctionnement du Programme

Pour afficher les métadonnées d'un fichier image :

L'image dont les métadonnées seront extraites est pris en paramètre à l'aide la méthode ImageMetadataReader.readMetadata(), dans l'objet instancié par la classe Metadata.

Dans le cas où le fichier image n'est pas un fichier de type .png ou de type .jpg nous lançons une exception, et si les extensions sont vérifiées, la lecture des métadonnées est mise en place.

Nous pouvons ensuite afficher une liste de métadonnées du fichier image en le parcourant, grâce aux méthodes correspondantes de la classe Directory et Tag.

# IV. STEGANOGRAPHIE

Dans ce projet, nous nous intéresserons à la stéganographie d'une image, c'est le fait de dissimuler un message dans un fichier image de type png ou jpeg.
Il existe plusieurs techniques stéganographique pour faire cela, comme la manipulation des bits de poids faible (*Least Significant Bit (**lsb**) en anglais*) ou bien l'usage de la palette de couleurs d'une image, etc. Dans ce projet, nous allons nous intéresser à la technique du Bit de Poids Faible.

## 1. Implémentation / Package utilisés

**java.awt.Color;**

La classe **Color** appartient au paquetage AWT (Abstrawt Window Toolkit) et permet de créer une couleur avec les valeurs RGB (Red, Green, Blue). La valeur de chacun des composants de RGB varie de 0 à 255 ou de 0,0 à 0,1.

**java.awt.image.BufferedImage;**

La classe **BufferedImage** est une sous-classe de la classe Image et permet de manipuler les données d'un fichier image.

## 2. Encoder

### a. Le fonctionnement d'encodage

Pour encoder, la technique que nous avons utilisée est celle de LSB.

En premier, l'utilisateur devra entrer le nom de l'image qui va contenir le message puis il doit entrer ce message et taper le nom de la nouvelle image contenant ainsi ce message, qui sera enregistré dans le même emplacement que le fichier original.

Le message tapé sera converti en un message binaire et sera reconverti plus tard en un message de String dans la partie de décodage.

Pour encoder :
Dans l'objet BufferedImage, nous avons pris l'image originale en paramètre, à l'aide de la méthode ImageIO.read() et le message qui est converti en binaire est également pris en paramètre.

Le message pris en paramètre est stocké dans un tableau de bytes. Et chaque bit de ces bytes sera encodé au niveau du bit de poids faible.

Nous avons ensuite une image contenant le message encodé et sauvegardée sous un nouveau nom dans le même emplacement que l'image originale et sous un format .png en utilisant la méthode ImageIO.write().

**Exemple avec le mode GUI :**
Dans l'image ci-dessous, nous avons caché le message « ceci est un message secret» dans l'image nommée *CERGY.png* et le nom du fichier image qui va avoir ce message encodé est nommé *nouvImage.png* (à choisir par l'utilisateur).

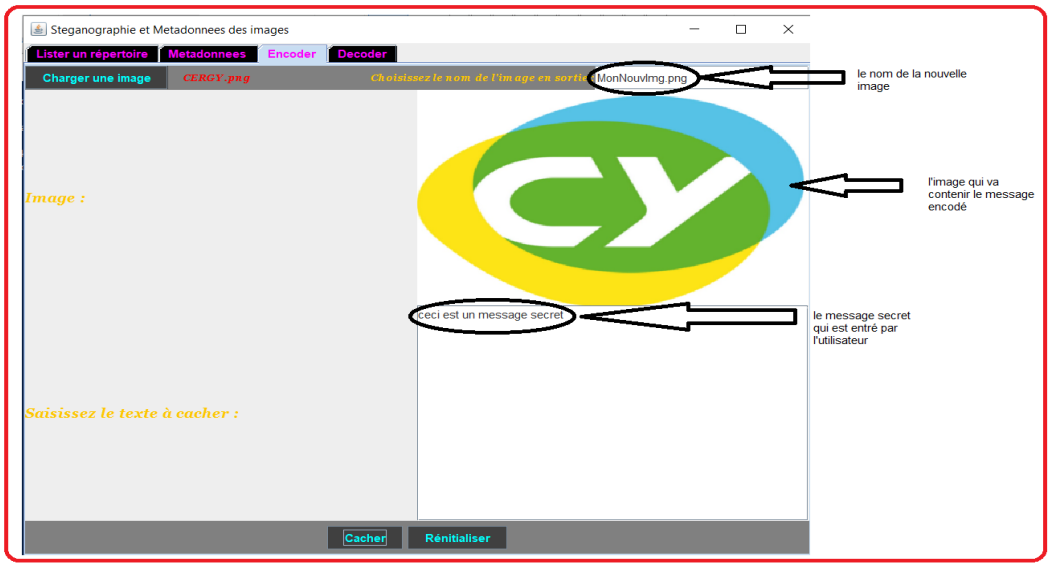



*Figure 3 : Diagramme illustrant l'onglet "encoder" de l'application*

## 3. Décoder

### a. Le fonctionnement de décodage

Pour décoder, nous prenons en paramètre de la méthode correspondante, l'image ayant le message encodé, à l'aide de la méthode ImageIO.read().

Ensuite, nous extrayons le message en binaire qui est caché dans l'image et pour avoir ce message binaire sous une forme bien construite, nous avons utilisé la classe StringBuilder qui a pour objectif de créer une chaine mutable et dans notre cas il s'agit d'une séquence binaire.

La séquence binaire est ensuite récupérée puis stocké dans un tableau de byte.
Ensuite, le message est converti en char (chaîne de charactère) à l'aide du casting, c'est le fait de forcer le compilateur à prendre une variable déclaré d'un certain type en un type différent.

Et nous avons enfin notre message en chaine charactère qui s'affiche sur la fenêtre.

**Exemple avec le mode GUI :**
Dans l'image ci-dessous, l'utilisateur doit choisir une image contenant un message secret et ensuite, il peut décoder.

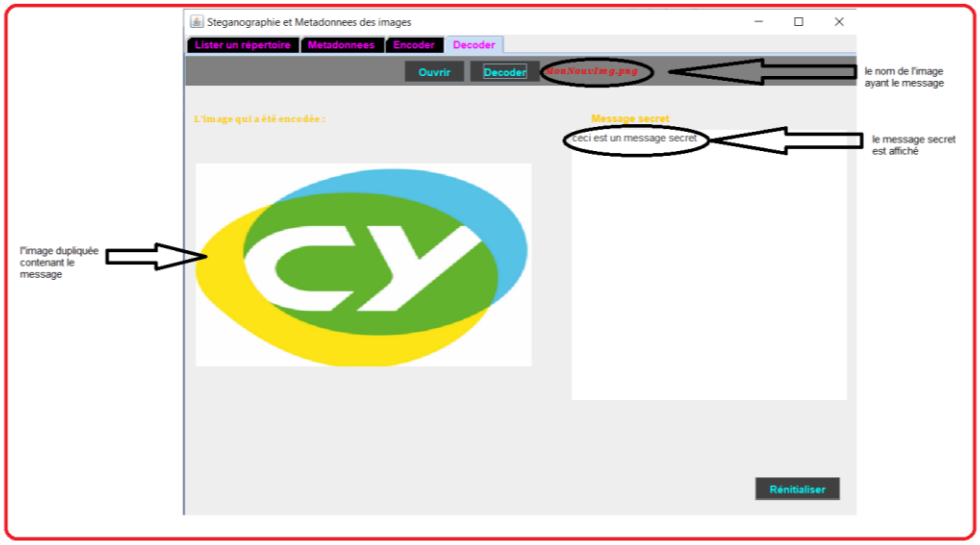
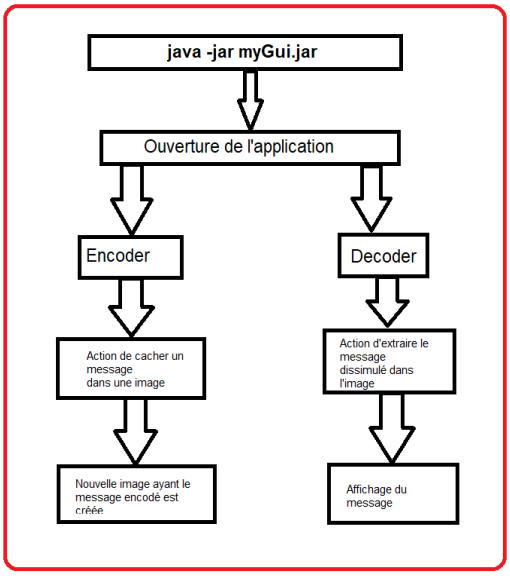



*Figure 4 : Diagramme illustrant l'onglet "décoder" de l'application*

## 4. Schéma récapitulant les fonctionnements d'encodage et de décodage




Comme dit précédemment, ce projet est exécutable en mode CLI et GUI.
Nous pouvons l'exécuter en mode GUI par un fichier jar nommé « myGui.jar » sur une console.
La console que nous avons utilisée est Ubuntu (WSL).

*Figure 5 : Récapitulatif des principes de la Stéganographie*

## V. DIAGRAMME UML DE L'ENSEMBLE DES CLASSES DU PROJET

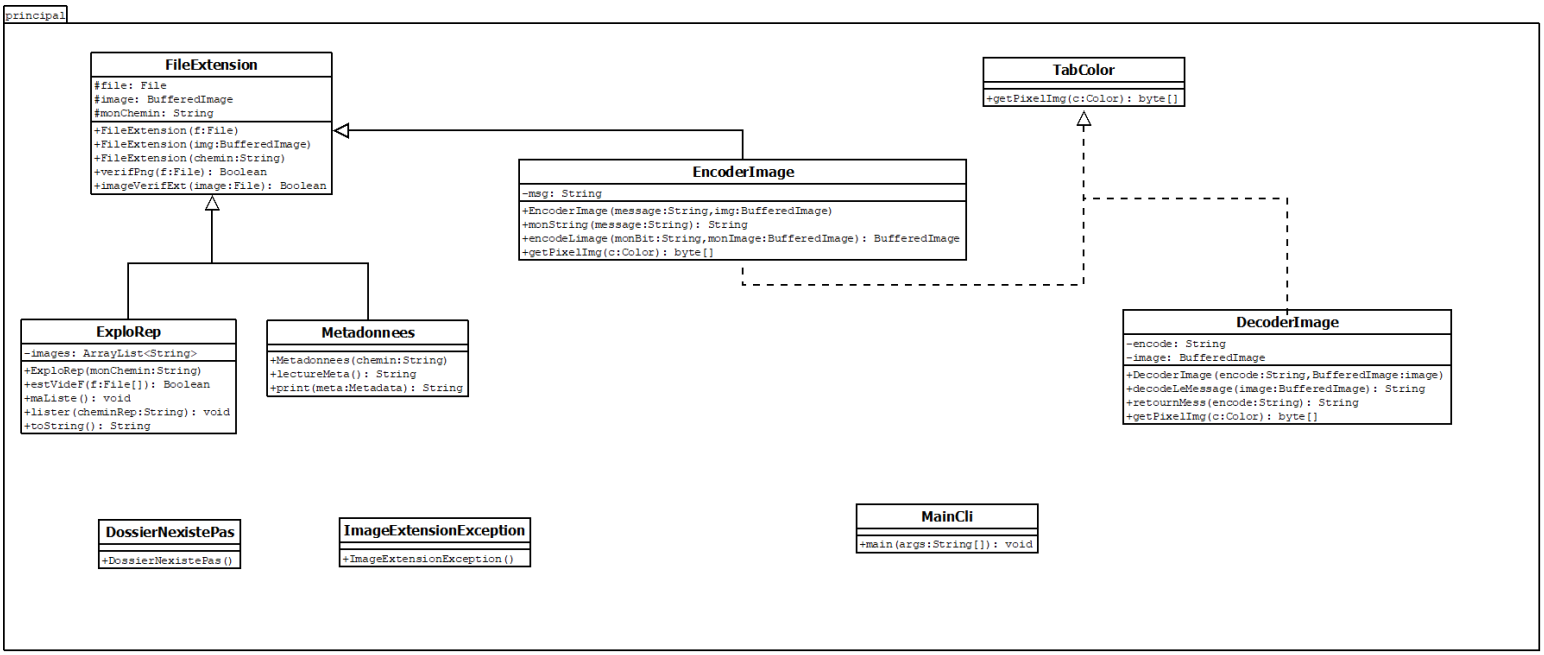



*Figure 6 : Diagramme UML*

## VI. CONCLUSION ET BILAN DU PROJET

La stéganographie est un sujet très intéressant et peut être utilisé pour cacher des messages ou même parfois un fichier dans un autre fichier (mais dans ce projet, nous nous limiterons à cacher un message/texte dans une image).

Notre application a pour but d'afficher les métadonnées d'un fichier image et de dissimuler un message en utilisant la stéganographie.
La méthode utilisée pour la stéganographie est celle de LSB.

Notre application est remarquable pour sa capacité à supporter tous les fichiers de type de png et jpeg (ou jpg), pouvant ainsi extraire les métadonnées des fichiers images et enfin réaliser la stéganographie.
Cependant, l'application ne peut traiter qu'un fichier à la fois.

Malgré cela, notre application présente des inconvénients car l'absence d'un filtre pouvant détecter une image ayant un message encodé pourrait provoquer un arrêt complet de notre application si l'utilisateur essaie de décoder une image qui n'a pas été encodé et le fait de ne pas pouvoir afficher les métadonnées uniquement sous le format EXIF reste un point négatif.

Néanmoins, durant ce projet, nous avons appris à mieux utiliser GitHub et cette plateforme nous a été très utile pour ce projet collaboratif.

A travers ce projet, nous avons pu comprendre que gérer un projet nécessite un travail constant et très sérieux.

Nous tenons à remercier l'ensemble de nos professeurs et en particulier :
Mr. Marc LEMAIRE
Mr. Tianxiao LIU
Mr. Jean Luc BOURDON
pour nous avoir donné des conseils pour le projet.