Rapport de projet – Informatique

ENSMM

Voydie Dorian

1ère année

Année scolaire 2018-2019

École Nationale Supérieure de Mécanique et des Microtechniques
26, rue de l'Épitaphe
25030 Besançon cedex – France
+33 381 40 27 32
http://www.ens2m.fr

Sommaire :

1- Analyse des besoins et spécifications fonctionnelles

Ce logiciel a pour but d'utiliser différentes méthodes de stéganographie et cryptage afin de permettre la transmission d'informations de manière discrète et sécurisée. Ce genre de logiciel sert par exemple dans les domaines de la cybersécurité.

1-1. Cacher une image dans une autre image

Cette partie consiste à se servir du codage d'une image à partir des bits de ses pixels afin de placer une image dans une autre et ainsi envoyer cacher une image dans une autre. Ce principe consiste à récupérer les bits de poids fort de l'image à cacher et de les injecter dans les bits de poids faible de l'image contenant. Le destinataire pourra ainsi observer l'image cachée en révélant uniquement les bits de poids faible du contenant.

1-2. Cacher un texte dans une image

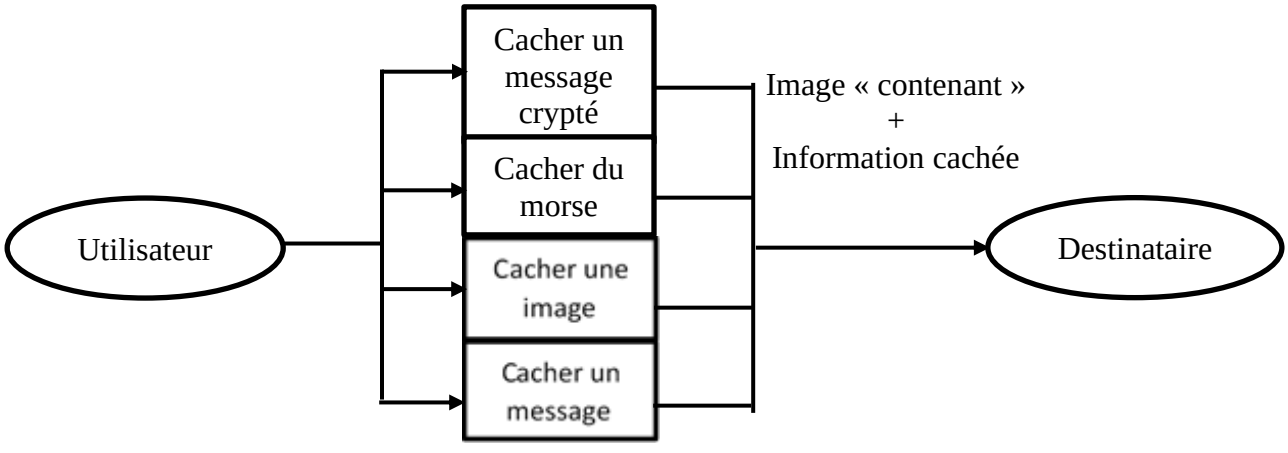
Ici, ce n'est pas une image, mais un message texte que nous allons cacher dans l'image contenant à l'aide du codage ASCII des caractères. On injectera donc dans chaque bit de poids faible chaque bit du codage ASCII de notre message.

1-3. Cacher un texte crypté à l'aide d'une clé dans une image

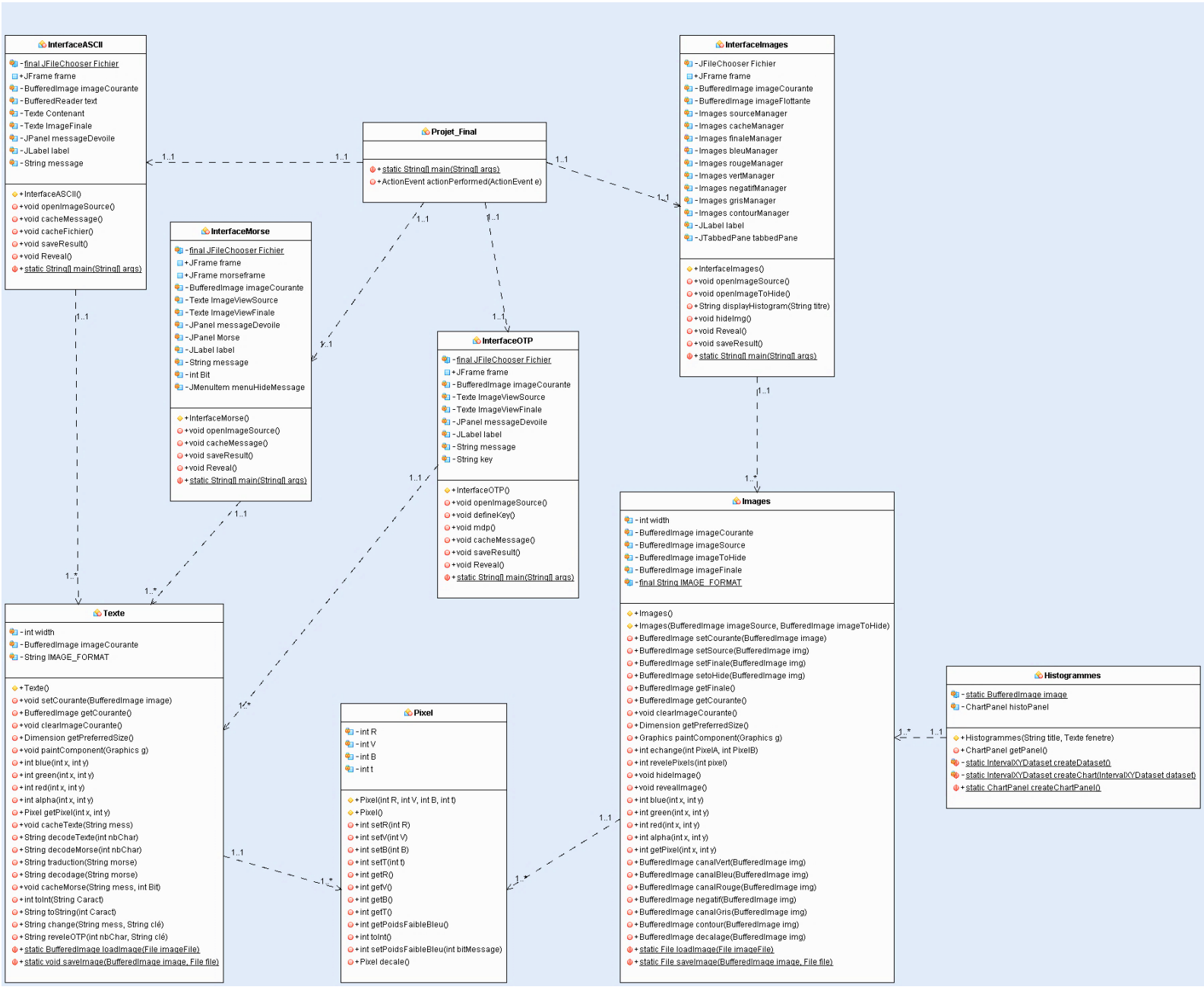
Il s'agit d'une alternative permettant de crypter le message à l'aide d'une clé prédéfinie par l'utilisateur.

1-4. Cacher un texte écrit en morse dans une image

Nous entrons ici le message à cacher en morse pour une sécurité supplémentaire.



2- Conception architecturale



Notre projet se compose de 9 classes : Cinq d'entre elles sont des « main class » correspondant à des interfaces différents, les quatre autres correspondent au cœur du programme. La classe « Projet_Final » correspond au menu de démarrage, puis permet l'accès aux 4 interfaces correspondant à chacun des modules explicités plus haut (1.1 à 1.4). Les opérations de stéganographie sont réalisées par les classes « Texte » et « Images », en utilisant la classe « Pixel » créée au préalable. La classe « Histogramme » vient se greffer sur la classe « Images » afin de permettre la stéganalyse statistique d'une image.

3- Conception détaillée

3-1. La classe « Images »

La stéganographie entre deux images réside en 2 méthodes :

```
public int echange(int PixelA, int PixelB){
    //return (PixelA & 0xFFFEFEFE) | (PixelB & 0x00808080)>>7; //Avec 1 LSB
    return (PixelA & 0xFFFCFCFC) | (PixelB & 0x00C0C0C0)>>6; //Avec 2 LSB
    //return (PixelA & 0xFFF8F8F8) | (PixelB & 0x00E0E0E0)>>5; // Avec 3 LSB
}
```

La première permet de récupérer un certain nombre de bits de poids fort (1, 2 ou 3) du PixelB, de les décaler en position de bits de poids faible, et venir remplacer ceux du PixelA.

```
public void hideImage(){
    for(int x =0 ; x< imageContenante.getWidth();x++){
        for(int y =0 ; y< imageContenante.getHeight();y++){
            if(x<imageToHide.getWidth() && y<imageToHide.getHeight()){
                imageFinale.setRGB(x, y, echange(
                imageContenante.getRGB(x, y),imageToHide.getRGB(x, y))
                );
            }
            else{
                imageFinale.setRGB(x,y,imageContenante.getRGB(x, y));
            }
        }
    }
}
```

La seconde permet de cacher dans l'image contenante une image à cacher, en remplaçant pixel par pixel les bits de poids faible de l'image contenante par les bits de poids fort de l'image à cacher.

La stéganalyse visuelle repose également sur 2 méthodes :

```
public int revelePixels(int pixel){
    //return (pixel & 0xFF010101)<<7; //Avec 1 LSB
    return (pixel & 0xFF030303)<<6; //Avec 2 LSB
    //return (pixel & 0xFF070707)<<5; //Avec 3 LSB
}
```

L'image à cacher étant dissimulée dans les bits de poids faible du contenant, cette méthode permet de ramener les bits de poids faible en poids fort. Ainsi si une image est dissimulée, elle sera dévoilée.

```
public void revealImage(){
    for (int i = 0; i < width; i++) {
        for (int j = 0; j < height; j++) {
            imageCourante.setRGB(i, j,revelePixels(imageCourante.getRGB(i, j)));
        }
    }
}
```

Il suffit alors de parcourir toute l'image et de décaler pixel par pixel les bits de poids faible, afin de faire ressortir une potentielle image dissimulée.

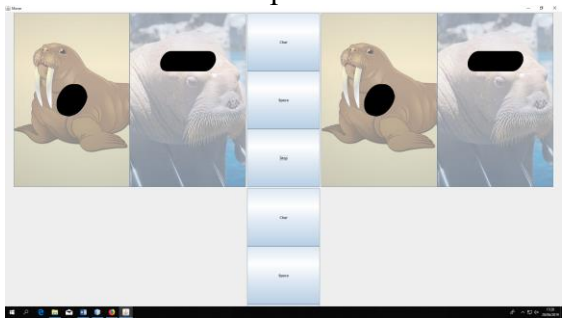
4- Codage et test

Dans cette partie nous verrons les limites de notre codage et les problèmes rencontrés.

Dans notre interface permettant de cacher une image dans une autre image, l'image à cacher ne peut pas être plus grande que l'image contenant. Pour résoudre ce problème, nous pourrions créer un algorithme permettant de rétrécir l'image à cacher en divisant par 4 le nombre de pixels (on associe un pixel dans l'image rétrécie à un carré de 2x2 pixels).

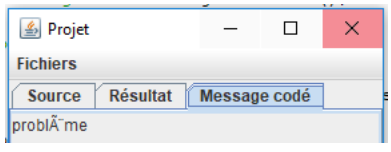
Dans notre interface permettant de cacher un message en morse, nous n'affichons pas le texte écrit par l'utilisateur lorsqu'il entre un message, il peut donc vite se perdre ou rater un caractère s'il perd un minimum d'attention. Nous pourrions tenter d'ouvrir une fenêtre d'affichage du texte, permettant ainsi à l'utilisateur de savoir où il en est dans son message à tout moment (rafraichissement).

Pour l'instant nous observons une erreur dans la fenêtre d'écriture du morse : après avoir entré son message en morse nous pouvons reprendre l'écriture du message à tout moment, cependant la fenêtre d'écriture présente une anomalie probablement due à une mauvaise régénération de la fenêtre :



Par manque de temps, nous n'avons pas encore résolu ce problème puisque nous nous concentrions d'abord sur le bon déroulement des interfaces principales (« Images » et « ASCII »), sachant qu'il n'a pas de réelle influence sur notre programme, hormis pour l'esthétique.

Nous avons aussi remarqué que les caractères avec des accents ne sont pas correctement traduits dans les messages récupérés. En effet, le codage ASCII traduit les caractères avec des accents sur 16 bits, or dans notre méthode nous effectuons les traductions en UTF-8 sur 8 bits, nous ne pouvons donc pas encoder/décoder les caractères avec des accents (é / è / à / …) comme ici par exemple dans le message « problème » :



5- Tests d'acceptation et guide d'utilisation

Avant de commencer, suivre le guide « INSTALLATION RAPIDE DES LIBRAIRIES EXTERNES.pdf » afin de rendre effective la classe « Histogrammes ».

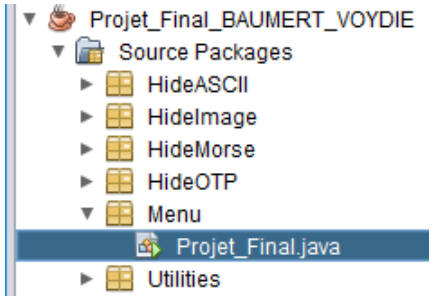
Tout d'abord, nous allons lancer le programme principal permettant d'accéder à toutes les méthodes :

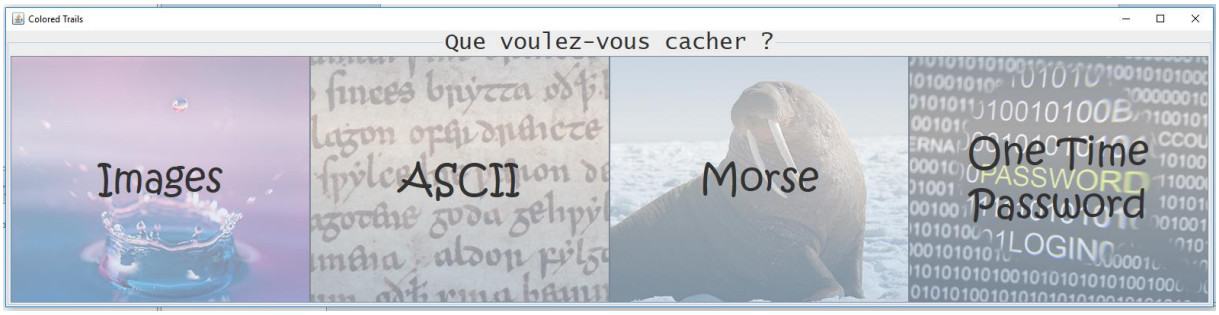
Dans NetBeans, dans la fenêtre dédiée aux projets, allez dans :

« Projet_Final_BAUMERT_VOYDIE > Source Packages > Menu »

Exécutez la seule main class du package : « Projet_Final »

La fenêtre principale du programme va alors s'ouvrir

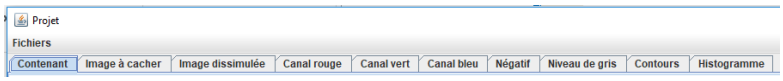




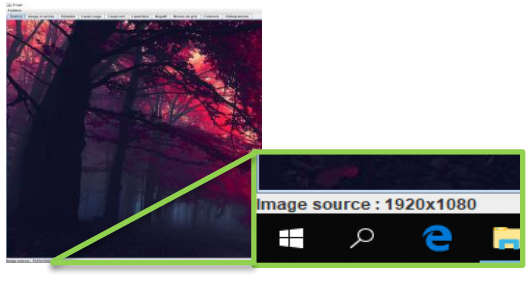
Si vous fermez une des quatre fenêtres annexes, le menu demeurera ouvert. Si vous fermez cependant le menu, toutes les autres fenêtres se fermeront et le programme s'arrêtera.

5-1. « Image »

Dans la fenêtre principale, cliquez sur « Images » afin d'ouvrir l'interface permettant de cacher une image dans une autre :

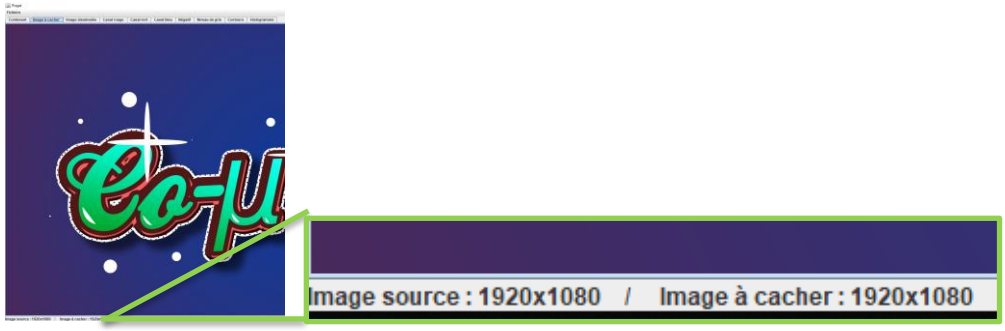


Pour ouvrir une image « Contenant », faites « Fichiers > Ouvrir un contenant », puis sélectionnez dans vos fichiers l'image que vous souhaitez insérer dans l'interface. Elle se trouvera alors dans l'onglet « Contenant » avec dans le coin inférieur gauche de l'interface, la taille de l'image choisie :



Vous pouvez explorer les différents onglets de « Canal rouge » à « Contours » pour voir les différents filtres que l'on peut appliquer à l'image « Contenant » (les filtres s'appliqueront toujours et uniquement à l'image située dans le JPanel « Contenant »).

Pour choisir l'image à cacher, faites Fichiers > Ouvrir une image secrète, puis sélectionnez dans vos fichiers l'image que vous souhaitez insérer dans l'interface. Elle se trouvera dans l'onglet « Image à cacher », avec à la suite de la taille de l'image « Contenant », la taille de l'image à cacher



Il vous suffira ensuite de faire « Fichiers > Cacher Image » afin de cacher l'image dans le contenant.

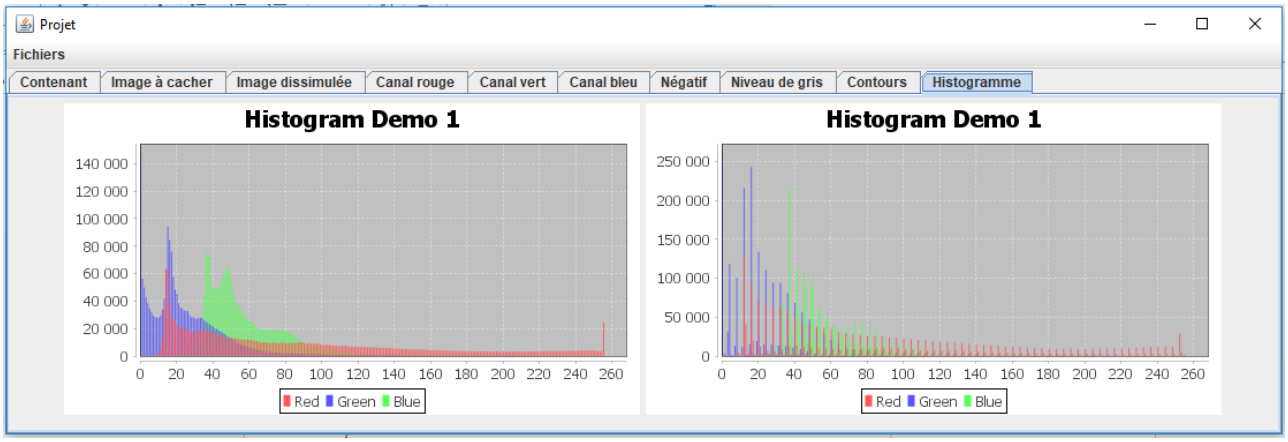
Vous pouvez ensuite sauvegarder votre image en faisant « Fichiers > Sauvegarder ».

Pour révéler une image cachée dans un contenant, faites « Fichiers > Effectuer une stéganalyse visuelle », puis choisissez dans vos fichiers l'images contenant l'image cachée. L'image cachée apparaîtra alors dans l'onglet « Image dissimulée » :

```
public int echange(int PixelA, int PixelB){
    //return (PixelA & 0xFFFEFEFE) | (PixelB & 0x00808080)>>7; //Avec 1 LSB
    return (PixelA & 0xFFFCFCFC) | (PixelB & 0x00C0C0C0)>>6; //Avec 2 LSB
    //return (PixelA & 0xFFF8F8F8) | (PixelB & 0x00E0E0E0)>>5; // Avec 3 LSB
}
public int revelePixels(int pixel){
    //return (pixel & 0xFF010101)<<7; //Avec 1 LSB
    return (pixel & 0xFF030303)<<6; //Avec 2 LSB
    //return (pixel & 0xFF070707)<<5; //Avec 3 LSB
}
```

Plus on augmente le nombre de LSB, plus la stéganographie est flagrante, mais la qualité de l'image cachée récupérée est meilleure. Vous pouvez changer le nombre de LSB pris en compte dans « Source Packages > Utilities > Images ».

Vous pouvez comparer les histogrammes des pixels de votre image « Contenant » avant et après insertion de l'image cachée en faisant Fichiers > Ouvrir l'histogramme et en sélectionnant l'image de base, puis en répétant l'opération pour séléctionner l'image modifiée que vous aurez enregistré au préalable dans vos dossiers. Vous pourrez alors observer les deux histogrammes dans l'onglet « Histogramme » : La stéganographie est flagrante.



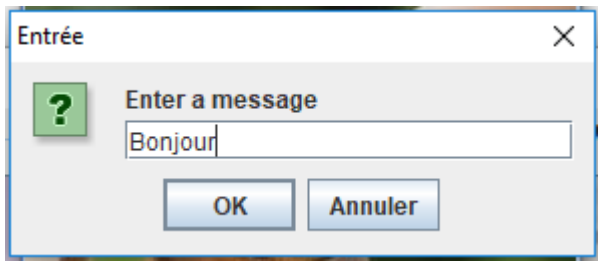
5-2. « ASCII »

Dans la fenêtre principale, effectuez un clic gauche sur « ASCII » afin d'ouvrir l'interface permettant de cacher un message dans une image.

Comme dans l'interface « Images », ouvrez le contenant en faisant « Fichiers > Ouvrir un contenant ».

Vous avez ensuite le choix :

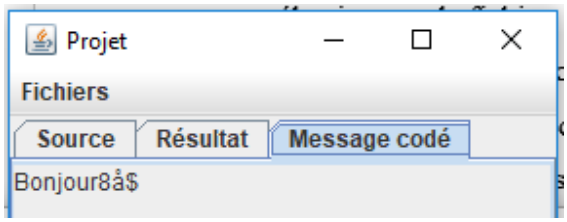
- Vous pouvez cacher un message que vous tapez au clavier en faisant « Fichiers > Cacher un message », puis en tapant votre message dans la fenêtre qui s'ouvrira :
- Vous pouvez cacher un fichier texte en faisant « Fichiers > Cacher un fichier texte », puis en sélectionnant le fichier (Ex : « test.txt »)



Dans l'onglet « Résultat » de notre interface, vous pouvez observer l'image contenant le texte caché (on peut alors remarquer pour un long texte caché quelques pixels dans l'image qui contrastent totalement avec leurs pixels environnants).

Vous pouvez sauvegarder l'image modifiée avec « Fichiers > Sauvegarder ».

Enfin pour réveler le message caché, faites « Fichiers > Dévoilez un potentiel message caché », puis en entrant le nombre de caractères tirés de l'image que vous souhaitez afficher. Votre message apparaîtra alors dans l'onglet « Message Codé » (par exemple ici nous avons demandé l'affichage de 10 caractères, nous retrouvons donc notre « Bonjour » au début du texte) :

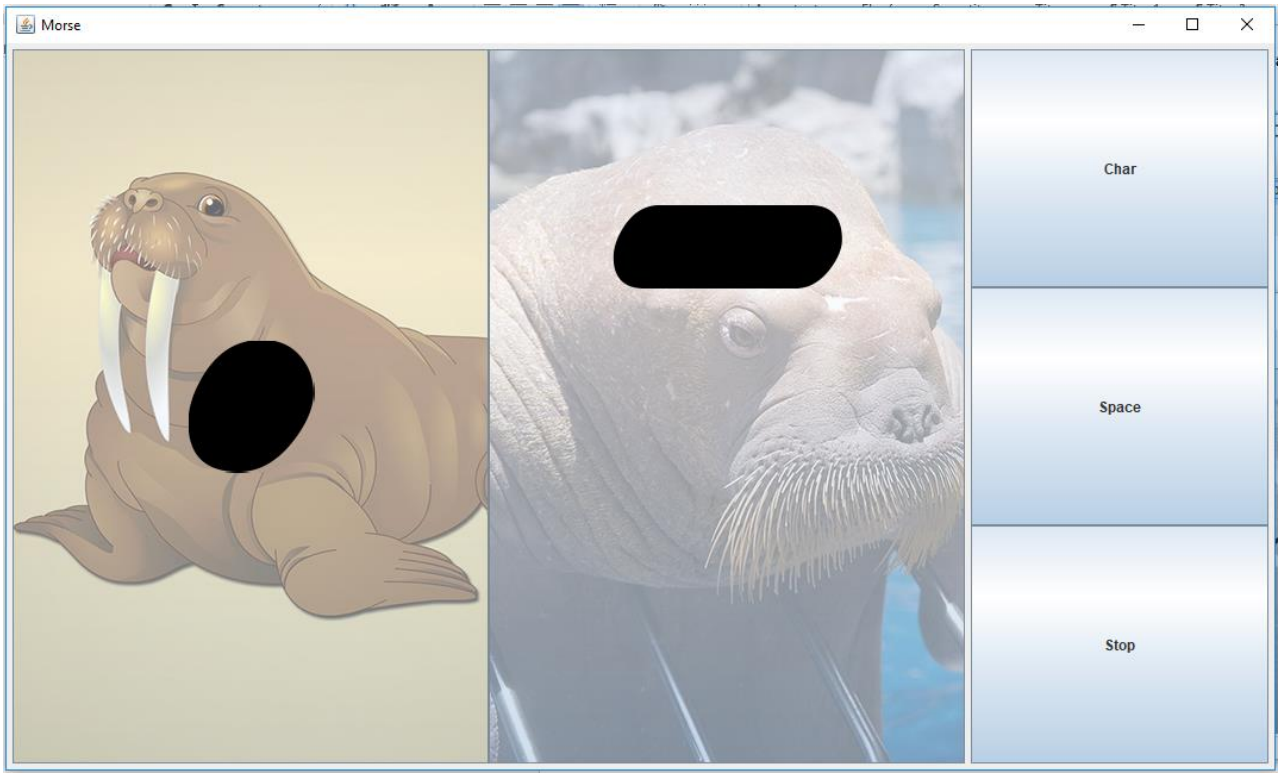


5-3. « Morse »

Dans la fenêtre principale effectuez un clic gauche sur « Morse » afin d'ouvrir l'interface permettant de cacher un message en morse dans une image.

Comme dans les interfaces précédentes, ouvrez le contenant en faisant « Fichiers > Ouvrir un contenant ».

Pour cacher votre message, faites » Fichiers > Cacher un message », une fenêtre prévue pour entrer votre message en morse va alors s'ouvrir :



Le point « . » correspond à un signal court et le tiret du haut « - » un signal long (vous trouverez la bibliothèque de codage morse aux lignes 454 à 567 pour traduire votre message en morse).

Pour entrer un caractère en morse, effectuez la combinaison correspondante à votre caractère puis :

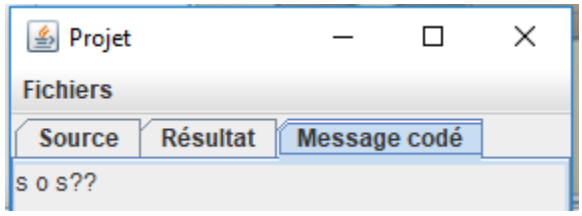
- Cliquez sur « Char » si vous souhaitez enregistrer le caractère entré et passer au **caractère** suivant (au sein d'un même mot).
- Cliquez sur « Space » si vous souhaitez enregistrer le caractère entré et passer au **mot** suivant.
- Cliquez sur « Stop » pour arrêter l'écriture du message qui sera alors dissimulé dans l'image.

Dans notre exemple nous avons entré le fameux « S O S » en faisant trois appuis courts, Space, trois appuis longs, Space, trois appuis court et **Char avant Stop** afin de bien sauvegarder le dernier caractère entré**.**

Votre image modifiée apparaîtra alors dans l'onglet « Résultat ».

Vous pouvez sauvegarder l'image modifiée comme sur les autres interfaces.

Pour afficher un potentiel message caché, faites « Fichiers > Dévoilez un potentiel message caché », puis entrez le nombre de caractères que vous souhaitez afficher (les caractères sont en morses, nous vous conseillons donc de demander 3 à 4 fois le nombre de caractères du message que vous voulez, par exemple pour « S O S » il y a 2 espaces, 3 tirets du haut « - » et 6 points « . » donc un total de 11 caractères) :



5-4. « One Time Password »

Dans la fenêtre principale effectuez un clic gauche sur « One Time Password » afin d'ouvrir l'interface permettant de cacher un message crypté avec une clé dans une image.

Comme dans les interfaces précédentes, ouvrez le contenant en faisant « Fichiers > Ouvrir un contenant ».

Pour cacher votre message, vous devez d'abord définir une clé de cryptage en faisant « Fichiers > Définissez une clé », puis en entrant la clé que vous souhaitez dans la fenêtre prévue à cet effet.

Après avoir défini votre clé, vous pourrez entrer le message à cacher en faisant « Fichiers > Cacher un message ». L'image contenant le message crypté apparaîtra alors dans l'onglet « Résultat ».

Vous pouvez sauvegarder l'image modifiée comme sur les autres interfaces.

Pour dévoiler un potentiel message crypté dans l'image, faites « Fichiers > Dévoilez un potentiel message caché », entrez le nombre de caractères que vous souhaitez afficher, puis vous devrez entrez la clé qui a servi à crypter le message, sinon vous ne retrouverez pas le message d'origine.

Le message apparaîtra alors dans l'onglet « Message codé ».