TP 1 : CORRECTION DE PHOTO

INTRODUCTION

Pour le premier tp, vous allez construire un logiciel qui corrige les couleurs sur une photo. Le logiciel va construire deux images résultantes pour chaque photo. Ainsi, deux algorithmes différents vont être comparés.

DESCRIPTION

ENTRÉE

Votre logiciel doit demander un nom de fichier à l'utilisateur. Ce fichier contiendra une image (bmp, gif, jpeg, jpg, jpe, png, tif, tiff, wbmp) qui sera chargée, analysée, corrigée et finalement sauvegardée. S'il y a une erreur lors de la lecture de l'image, alors un message d'erreur est affiché à l'aide de la commande `System.err.println( "votre message d'erreur ici" );` et ensuite le programme termine à l'aide de la commande `System.exit( -1 );`. Remarquez, la méthode *exit* peut prendre n'importe quelle valeur. Il est conventionnel de placer une valeur négative lors d'erreur.

Voici un code permettant de lire l'information qu'un utilisateur entre au clavier :

```
Scanner sc = new Scanner( System.in );

String nomFichierEntrees = sc.nextLine();
```

Voici le code permettant de lire une image dans un fichier et la placer dans une variable de type `BufferedImage`.

```
BufferedImage image = null;

try {
    image = ImageIO.read( new File( nomFichier ) );
} catch ( IOException e ) {
    … ERREUR …
}
```

TRAITEMENT

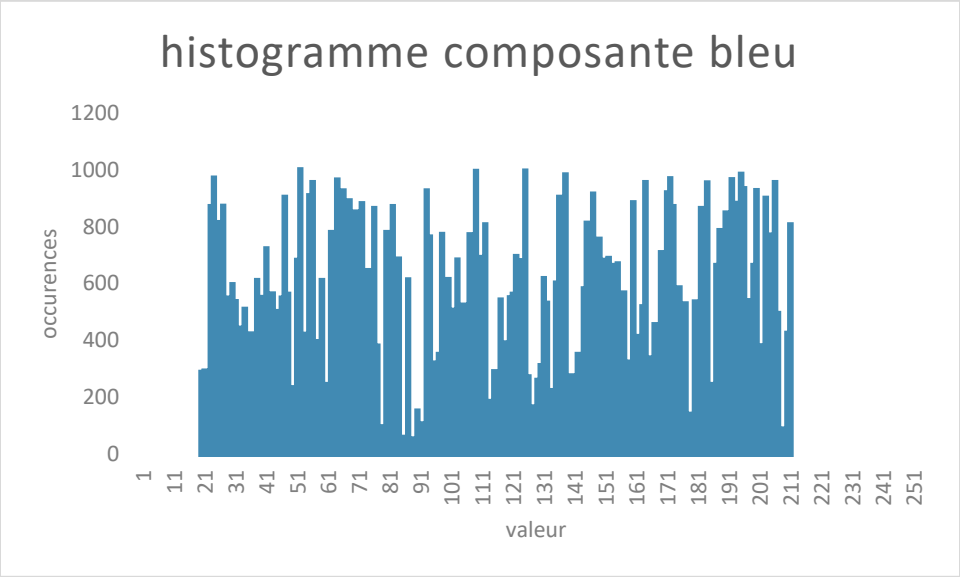
Lorsque l'image est chargée, vous allez devoir l'analyser. Une image est une matrice de point. Chaque point est une couleur. Dans notre cas, les couleurs seront représentées par 4 valeurs (composantes alpha, rouge, verte et bleu). Notre correction n'utilisera pas la composante alpha. Une composante est représentée par une valeur entre 0 et 255.

Le logiciel doit premièrement construire un histogramme pour chaque composante. C'est-à-dire, il doit calculer combien de points ont la composante rouge à 0, combien ont la composante rouge à 1, combien ont la composante rouge à 2, et ainsi de suite pour les valeurs de 0 à 255 et pour les trois composantes.

Lorsque les trois histogrammes sont construits, nous sommes prêts à corriger l'image. Votre logiciel va construire 2 images résultantes à partir de l'image de base. Nous voulons comparer deux corrections différentes. Pour les deux corrections, les composantes sont considérées comme indépendantes aux autres. Donc, chaque composante va être analysée indépendamment des autres.

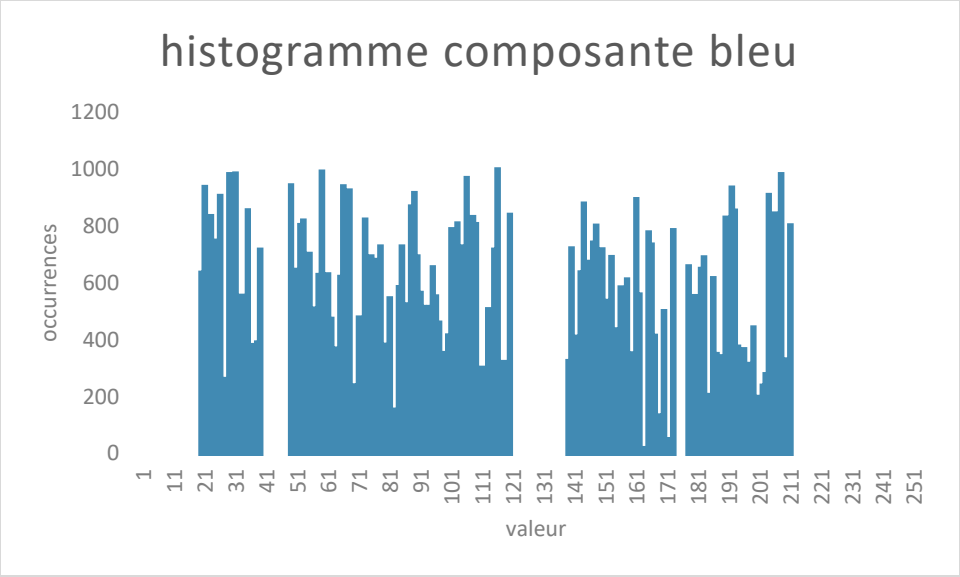
ANALYSE DE L'HISTOGRAMME D'UNE COMPOSANTE

Votre histogramme, pour une composante, contient les **occurrences** pour chaque **valeur** de 0 à 255. Pour l'analyse, nous sommes intéressés par les **occurrences** à zéro et celles qui ne sont pas à zéro. Par exemple, les **valeurs** de 0 à 18 et de 211 à 255 ont une **occurrence** de zéro dans l'histogramme suivant.



i	v_i^d	v_i^f
1	19	210

Les valeurs dans l'histogramme sont donc comprises entre $v_1^d = 19$ et $v_1^f = 210$. Il arrive qu'il y ait des zones à 0 à l'intérieure de l'histogramme. Par exemple :



Ici, nous avons 5 zones à zéro et 4 zones non à zéro ($k = 4$).

i	v_i^d	v_i^f
1	19	38
2	48	119
3	138	172
4	177	210

Votre analyse doit trouver ces zones et ensuite appliquer les deux algorithmes de correction.

CORRECTION

Vous devez ensuite transformer l'image de base en deux images corrigées. Pour cela, nous utilisons deux équations indépendantes qui seront appliquées à l'image de base. En résumé, pour chaque composante (rouge, vert, bleu), vous devez calculer l'histogramme, faire l'analyse, parcourir chaque point de l'image afin de transformer la valeur v de la composante en appliquant l'équation appropriée.

PREMIER ALGORITHME

Cet algorithme utilise seulement v_1^d et v_k^f. Il suffit simplement de prendre la valeur v d'une composante de l'image et la placer dans l'équation c_1 suivante afin d'obtenir la nouvelle valeur pour la composante :

$$c_1(v) = \frac{256(v - v_1^d)}{v_k^f - v_1^d + 1}$$

DEUXIÈME ALGORITHME

Pour le deuxième algorithme, vous devez premièrement calculer les valeurs milieu m_i de la façon suivante :

$$\forall i \in [1..k+1] \begin{cases} si\ i = 1\ \ alors\ \ m_i = 0 \\ si\ i = k+1\ \ alors\ \ m_i = (v_{i-1}^f + v_i^d)/2 \\ sinon\ \ m_i = 255 \end{cases}$$

Donc, pour l'exemple précédant, nous obtiendrions les valeurs suivantes :

i	v_i^d	v_i^f	m_i
1	19	38	0
2	48	119	$(38+48)/2 = 43$
3	138	172	$(119+138)/2 = 128$
4	177	210	$(172+177)/2 = 174$
5			255

Ensuite, vous devez trouver à quelle zone (i) votre valeur v appartient :

$$v_i^d \leq v \leq v_i^f$$

Et finalement, vous appliquer la formule c_2 sur votre valeur v en utilisant le i approprié.

$$c_2(v) = m_i + \frac{(m_{i+1} - m_i + 1)(v - v_i^d)}{v_i^f - v_i^d + 1}$$

SORTIES

Votre logiciel doit finalement sauvegarder les deux images finales. Pour le premier algorithme, ajouter les lettres 'cls' au nom du fichier d'entrées et pour le deuxième algorithme, ajoutez les lettres 'clc' au nom du fichier d'entrées. La ligne de code suivante permet de sauvegarder une `image` de type `BufferedImage` dans un fichier.

```
try {
      ImageIO.write( image, "jpg", new File( nomFichierCls ) );
} catch ( IOException e ) {
    … ERREUR …
}
```

CONSTRUCTION

Voici quelques suggestions pour la construction du logiciel. Vous n'êtes pas obligé de suivre ces suggestions.

- Construire une classe Histogramme qui sera utilisée pour recevoir les valeurs pour une composante et en faire l'analyse.
- Vous pouvez utiliser la classe Couleur que je vous ai donnée. Les valeurs dans un `BufferedImage` sont encodées dans un **`int`** et la classe Couleur permet de les décoder. Par exemple, le code suivant extrait

une couleur de l'image, la décode pour obtenir la composante `rouge`, modifie la composante `rouge` et ensuite replace la couleur modifiée dans l'image.

```
Couleur couleur = new Couleur( image.getRGB( x, y ) );

int rouge = couleur.getRouge();

rouge = c1( rouge );

couleur.setRouge( rouge );

image.setRGB( x, y, couleur.getArgb() );
```

- Allez lire sur la classe `BufferedImage`. Vous allez devoir construire des nouvelles instances afin de ne pas perdre l'image de base.
- Une classe pour vos messages d'erreurs. Peut-être un type énuméré.
- Vous avez deux algorithmes qui utilisent une équation différente (spécialisé) avec le même principe de base (généralisé) qui est de parcourir toute l'image pour modifier chaque composante. Une classe mère pour l'algorithme générique et des classes filles pour les sections spécialisées de l'algorithme. En profiter pour pratiquer les classes et méthodes abstraites.

DIRECTIVES

1. Le tp est à faire seul ou en équipe de deux.
2. Vous devez construire des classes appropriées.
3. Code :
    a. Vos méthodes ne devraient pas contenir plus de 5 instructions au total parmi : `if, for, while, try, switch.`
    b. Pas de `goto`, `continue`.
    c. Les `break` ne peuvent apparaître que dans les `switch`.
    d. Un seul `return` par méthode.
4. Indentez votre code.

COMMENTAIRES

- Commentez l'entête de chaque classe et méthode.
- Une ligne contient soit un commentaire, soit du code, pas les deux.
- Utilisez des noms d'identificateur significatif.
- Utilisez le français.
- Une ligne de commentaire ne devrait pas dépasser 80 caractères. Continuez sur la ligne suivante au besoin.
- Nous utilisons Javadoc :
    - La première ligne d'un commentaire doit contenir une description courte (1 phrase) de la méthode ou la classe.
        - Courte.
        - Complète.
        - Commencez la description avec un verbe.

        - Assurez-vous de ne pas simplement répéter le nom de la méthode, donnez plus d'information.
    - Ensuite, au besoin, une description détaillée de la méthode ou classe va suivre.
        - Indépendant du code. Les commentaires d'entêtes décrivent ce que la méthode fait, ils ne décrivent pas comment c'est fait.
        - Si vous avez besoin de mentionner l'objet courant, utilisez le mot 'this'.
    - Ensuite, avant de placer les **tags,** placez une ligne vide.
    - Placez les **tag** @param, @return et @throws au besoin.
        - @param : décris les valeurs acceptées pour la méthode.
    - Dans les commentaires, placer les noms de variable et autre ligne de code entre les tags <code> … </code>.
    - Écrivez les commentaires à la troisième personne.

REMISE

Remettre le tp par l'entremise de Moodle. Placez vos fichiers `'*.java'` dans un dossier compressé de Window, vous devez remettre l'archive. Le tp est à remettre avant le 11 octobre 23 :59.

ÉVALUATION

- Fonctionnalité (7 pts) : des tests partiels vous seront remis. Un test plus complet sera appliqué à votre tp.
- Structure (2 pt) : veillez à utiliser correctement le mécanisme d'héritage et de méthode.
- Lisibilité (2 pts) : commentaire, indentation et noms d'identificateur significatif.