Feuille d'exercices n° ? + ...?

Exercice 1. Objectifs :
- Manipulation de flux (d'images en l'occurence)
- Mise en œuvre d'algorithmes de traitement d'images
- Utilisation de Collection : List ; Comparaison LinkedList ? ArrayList

Pré-requis :
Pour manipuler les images, on utilisera la classe BufferedImage, voir sur l'API Java :
http ://docs.oracle.com/javase/7/docs/api/java/awt/image/BufferedImage.html

Lire/Ecrire une Image

```
//Lire
File f = new File("c://chemindufichier.jpg");
try {
      BufferedImage image = ImageIO.read(f);
} catch (IOException e) {
      System.out.println(e);
}

//Ecrire
try {
      File fic= new File("test"+ "." + "jpg");
      ImageIO.write(image, "jpg", fic);
} catch (IOException e) {
      System.out.println(e);
}
```

Obtenir la taille de l'image

```
int hauteur = image.getHeight();
int largeur = image.getWidth();
```

Obtenir/Modifier un Pixel en manipulant les niveau de rouge, de vert et de bleu

```
int pixel = image.getRGB(i, j); //i et j des coordonnés

//Obtenir les niveaux de couleur de ce pixel entre 0 et 255
int rouge = (pixel >> 16) & 0x000000FF;
int vert = (pixel >> 8 ) & 0x000000FF;
int bleu = (pixel) & 0x000000FF;

//Modifier la valeur de rouge
rouge = 250 ;

//Créer le nouveau pixel
pixel = (rouge << 16) + (vert << 8) + bleu;
//Mettre à jour le pixel dans l'image
image.setRGB(i, j, couleur);
```

Exercice 2.

1. Ecrire un main qui lit un fichier image "image.jpg" et en fait une copie nommée "image_copie.jpg"
2. Créer les classes `Pixel` et `ImageATraiter`. La classe `Pixel` possède des attributs *rouge*, *bleu* et *vert* de type entier. La classe `ImageATraiter` contient une `List` de `Pixel`.
3. Ecrire dans la classe `ImageATraiter` la méthode *public void obtainPixelFromImage(BufferedImage image)* qui prend une `BufferedImage` en paramètre et stocke chacun de ses pixels dans la liste.
4. Ecrire dans la classe `Pixel` une méthode qui retourne le niveau de gris du pixel. Pour calculer le niveau de gris, on utilisera la formule : 0.299*rouge + 0.587*vert + 0.114*bleu

5. Ecrire dans `ImageATraiter` la méthode *public void setNEtB(BufferedImage image)* qui prend en paramètre une `BufferedImage`, stocke tous ses pixels dans la liste, puis modifie chacun des pixels de l'image en les transformant en noir et blanc. Pour transformer un pixel en noir et blanc, on fait rouge=bleu=vert=niveau de gris.
6. Tester cette méthode en écrivant un main qui obtient une `BufferedImage` du disque dur, la transforme en noir et blanc, puis l'écrit sur le disque dur. Comparer l'efficacité de l'algorithme en implémentant la solution avec `ArrayList` puis `LinkedList` dans la classe `ImageATraiter`. Que constatez vous ?
7. Implémenter l'algorithme de *Sobel* qui permet de ne conserver que les contours d'une image dont voici une version simplifiée :

```
Pour i de 0 à largeur de image
    Pour j de 0 à hauteur de image
       sum_r=rouge(pixel(j+1, i-1)) + rouge(pixel(j+1, i)) +
       rouge(pixel(j+1, i+1)) -
       (rouge(pixel(j-1, i-1)) + rouge(pixel(j-1, i)) +
       rouge(pixel(j-1, i+1)))
       sum_b=bleu(pixel(j+1, i-1)) + bleu(pixel(j+1, i)) +
       bleu(pixel(j+1, i+1)) -
       (bleu(pixel(j-1, i-1)) + bleu(pixel(j-1, i)) +
       bleu(pixel(j-1, i+1)))
       sum_v=vert(pixel(j+1, i-1)) + vert(pixel(j+1, i)) +
       vert(pixel(j+1, i+1)) -
       (vert(pixel(j-1, i-1)) + vert(pixel(j-1, i)) +
       vert(pixel(j-1, i+1)))
       pixel(j, i) = (0.299*sum_r + 0.587*sum_g + 0.114*sum_b))
```