Licence Informatique
Université Bordeaux 1
2006-2007

UE INF203
Programmation par objets
Travaux Dirigés
feuille 8

Partie 1 : Ecriture et Lecture dans un fichier texte

Exercice 1.

La classe `FichierTexteV0` fournie en annexe permet la lecture et l'écriture caractère par caractère dans un fichier texte. À l'aide de la documentation de l'API java, étudier cette classe. On étudiera plus particulièrement la documentation des classes `FileInputStream` et `FileOutputStream`.

Exercice 2.

La classe `FichierTexteV1` fournie en annexe permet la lecture et l'écriture caractère ligne par ligne dans un fichier texte. À l'aide de la documentation de l'API java, étudier cette classe.On étudiera plus particulièrement la documentation des classes `FileReader BufferedReader`, `FileWriter`, `FileWriter`, `BufferedWriter` et `PrintWriter`.

Exercice 3.

Étudier les classes `FichierTexte` et `TestFichierTexte`.
Compléter la classe FichierTexte en introduisant une notion d'ouverture pour ajout en fin de fichier (option 'a').

Partie 2 : Images

Une image est une matrice à deux dimensions de pixels. On considère ici les images en niveaux de gris ce qui signifie que la valeur d'un pixel est un entier compris entre 0 (noir) et 255 (blanc).

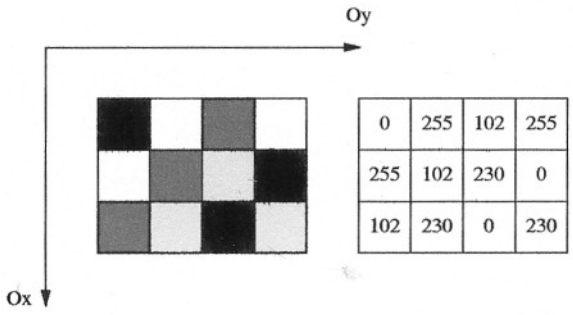

FIG. 1 – Exemple d'image en niveaux de gris

Exercice 4. Lecture/écriture d'images au format pgm

On peut stocker une image en niveaux de gris dans un fichier *pgm ascii*. Un tel fichier contient les informations suivantes :

```
P2
# commentaires
<largeur image> <hauteur image>
255
<niveau de gris 1> <niveau de gris 2> ...
```

La première ligne correspond au “magic number”, P2 dans le cas du format pgm ascii. Cette indication permet de distinguer plusieurs formats proches. Les commentaires (lignes commençant par #) sont facultatifs. Les niveaux de gris successifs sont obtenus par balayage des pixels de l'image de gauche à droite et de haut en bas. Exemple :

```
P2
100 76
255
245 245 245 245 245 245 245 245 245 245 245 245 245 245 245 245 245 245
245 245 245 245 245 245 245 245 245 245 245 245 245 245 239 245 241
241 212 244 244 244 244 245 245 245 245 245 245 245 245 245 245 245 245
245 245 245 245 245 245 245 245 245 245 245 245 245 245 245 208 245
245 245 245 170 245 180 180 245 245 150 150 245 209 245 245 245 245
193 171 171 184 237 245 245 245 245 240 245 245 245 245 164 164 245
173 174 174 245 245 245 245 245 245 209 245 245 235 245 245 245 245
...
```

Regardez le contenu du fichier pgm donné en exemple et visualisez le avec *gimp* par exemple.

On vous donne le début de la classe ImageNG correspondant à une image en niveaux de gris. Lisez attentivement le contructeur de cette classe en vous aidant de la documentation de l'API java. Complétez la méthode de sauvegarde d'une image au format pgm en utilisant la classe *FileWriter*.

Exercice 5. Traitements élémentaires

Le but de cet exercice est d'écrire des fonctions de traitement d'images en niveaux de gris. Pour cela, vous complèterez la classe *ImageNG* en implémentant l'interface *TraitementImage* :

```
public interface TraitementImage {
    public void couper (int x0, int y0, int nouvelleLargeur, int nouvelleHauteur);
    public void retailler (float echelleX, float echelleY);
    public void seuiller (int seuil);
    public void lisser ();
}
```

- La méthode *couper* découpe l'image initiale à partir du pixel (x_0, y_0). L'image transformée est de dimension $nouvelleHauteur \times nouvelleLargeur$ et son coin en haut à gauche correspond au point (x_0, y_0) de l'image initiale.
- La méthode *retailler* agrandit ou réduit l'image. Les coefficients $echelleX$ et $echelleY$ sont les facteurs d'échelle en abscisse et en ordonnée respectivement.
- La méthode *seuiller* remplace tous les niveaux de gris inférieurs au seuil par du noir et tous les niveaux de gris supérieurs au seuil par du blanc.
- La méthode *lisser* remplace la valeur de chaque pixel par la moyenne des valeurs de son voisinage dans l'image. Ce voisinage est composé du pixel considéré et de ses huits voisins dans l'image. Ce traitement rend l'image “floue”.

Ecrivez un programme de test.

Exercice 6. Convolution par masque

Dans une transformation par convolution, le nouveau niveau de gris d'un pixel est calculé en fonction du niveau de gris des pixels situés dans son voisinage. Le voisinage considéré est sélectionné par une grille de taille $n \times m$ (appelée masque) centrée sur le pixel traité. Chaque case du masque contient une information qui décrit comment le niveau de gris du pixel qu'il cache intervient dans le calcul. Plus précisément, le nouveau niveau de gris d'un pixel est obtenu en calculant la somme des niveaux de gris des pixels de son voisinage pondérée par les coefficients du masque, que l'on divise ensuite par la somme des valeurs absolues des coefficients du masque.

Le lissage écrit à l'exercice précédent peut être interprété comme une convolution par le masque 3×3 suivant :

1	1	1
1	1	1
1	1	1

FIG. 2 – Masque de convolution pour le lissage 3×3.

Ajouter dans l'interface *TraitementImage* et implémentez dans la classe *ImageNG* la méthode de convolution *void convoluer(int[][] masque)*. On supposera que le masque est carré et de taille impaire. Réécrivez *lisser* en utilisant convoluer. Testez.

Exercice 7. Extension aux images couleurs

Dans une image couleur, chaque pixel est codé par trois valeurs entières comprises entre 0 et 255 correspondant respectivement au niveau de rouge, de vert et de bleu (RGB). On peut stocker une telle image au format *ppm ascii* très similaire au format pgm :

```
P3
# commentaires
<largeur image> <hauteur image>
255
<niveau de rouge 1> <niveau de vert 1> <niveau de bleu 1>
<niveau de rouge 2> <niveau de vert 2> <niveau de bleu 2> ...
```

Ecrivez une classe *ImageCouleur* fournissant les mêmes services que la classe *ImageNG*. Repérez les redondances de code. Comment factoriser au mieux le code ?