

## Manipulation d'images<sup>1</sup>

*Respectez les noms des méthodes proposées afin de permettre l'utilisation des classes fournies, notamment pour l'affichage.*

On s'intéresse à la représentation et la manipulation d'images. Ces images seront constituées de pixels caractérisés par une couleur représentant un niveau de gris.

Pour un aperçu du travail à réaliser : `java -jar image.jar /images/fruit.pgm 15 16`, ou remplacer `fruit.pgm` par un autre fichier fourni dans le dossier `images` l'archive sur le portail.

**Les couleurs** Les niveaux de gris considérés seront codés par un entier sur 8 bits, et donc entre 0 et 255. De telles couleurs sont représentées par la classe `GrayColor` du paquetage `image.color`, que l'on supposera connue et définie ainsi :

<b>GrayColor</b>
+ <code>WHITE</code> : <code>GrayColor</code>
+ <code>BLACK</code> : <code>GrayColor</code>
- <code>level</code> : <code>int</code>
+ <code>GrayColor(level : int)</code>
+ <code>getLevel()</code> : <code>int</code>
+ <code>equals(o : Object)</code> : <code>boolean</code>

où l'attribut `level` représente le niveau de gris (entre 0 et 255) de cette couleur. Les constantes `BLACK` et `WHITE` correspondent respectivement aux couleurs dont le niveau de gris est 0 et 255.

Codez la classe `GrayColor`.

**Les Pixels** Les pixels sont modélisés par des objets de la classe `Pixel`. Les instances de cette classe dispose d'un attribut représentant leur couleur, comme définie ci-dessus.

**Q 1 .** Codez la classe `Pixel` :

- la classe `Pixel` appartient au paquetage `image`
  - son attribut (de type `image.color.GrayColor`) et ses modificateur (`setColor`) et accesseur (`getColor`)
  - un constructeur qui prend en paramètre la valeur d'initialisation de l'attribut
  - la méthode `equals` qui considère que deux pixels sont égaux si leurs couleurs sont égales
  - une méthode `colorDifference` qui a pour résultat un entier positif qui correspond à l'écart entre le niveau de gris de ce pixel et d'un autre.
- Pour obtenir la valeur absolue d'un nombre vous pourrez utiliser la méthode `abs` de la classe `java.lang.Math`.

**Les images** Une image est modélisée par des instances de la classe `Image` du paquetage `image`.

Les instances de cette classe sont caractérisées par un tableau à deux dimensions de pixels, tels que définis ci-dessus. Cette classe dispose

- d'un constructeur qui prend en paramètre la largeur et la hauteur de l'image en nombre de pixels. L'image créée est initialement composée uniquement de pixels de couleur blanche.
- des méthodes `getWidth` et `getHeight` d'accès à la largeur et à la hauteur de l'image,
- d'une méthode `getPixel` qui a deux paramètres entiers  $x$  et  $y$  et a pour résultat le pixel de coordonnées  $(x, y)$  de cette image, c'est-à-dire du tableau à deux dimensions de pixels. On considèrera que le pixel «en haut à gauche» de l'image a pour coordonnées  $(0, 0)$ .

Cette méthode lève une exception `ArrayIndexOutOfBoundsException` si le pixel de coordonnée  $(x, y)$  n'existe pas pour cette image.

- d'une méthode `changeColorPixel` de signature :

```
public void changeColorPixel(int x, int y, GrayColor color)
```

<sup>1</sup>Inspiré de «Introduction à la science informatique» dirigé par Gilles Dowek, p113.

qui pour effet d'attribuer la couleur `color` au pixel de coordonnées  $(x,y)$ .

Cette méthode lève une exception `ArrayIndexOutOfBoundsException` si le pixel de coordonnée  $(x,y)$  n'existe pas pour cette image.

- d'une méthode

```
public Image edges(int threshold)
```

qui permet de créer une nouvelle image obtenue à partir de l'image initiale par *extraction de contours*. Un exemple du résultat souhaité est présenté à la figure ??.

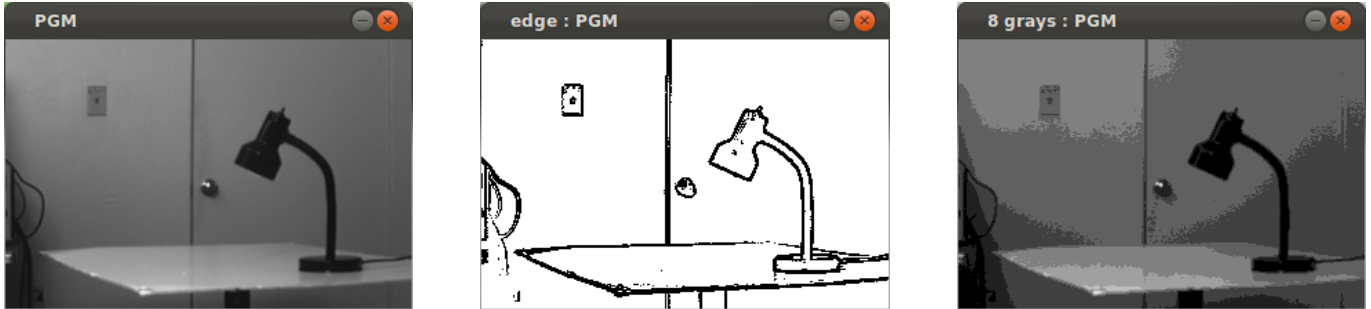


Figure 1: A gauche l'image initiale, au centre l'image résultat obtenue par extraction de contours avec un seuil de 10 et à droite l'image obtenue en diminuant à 8 le nombre de niveaux de gris.

Il n'est pas très compliqué d'obtenir l'image résultat (l'image des «*contours*») à partir de l'image initiale. En effet, une manière de procéder<sup>2</sup> est la suivante : un pixel de l'image résultat est noir s'il appartient au contour de l'image initiale, c'est-à-dire s'il est *très différent* de l'un des deux points situés à sa droite ou en-dessous de lui dans l'image initiale. «*Très différent*» signifie que la différence entre les niveaux de gris de deux pixels est supérieure à un *seuil* fixé. Les autres pixels de l'image résultat (ce qui ne sont pas identifiés comme appartenant à un contour) sont blancs.

Le seuil à prendre en compte pour l'extraction des contours est le paramètre<sup>3</sup> de la méthode `edges`.

- d'une méthode

```
public Image decreaseGrayLevels(int nbGrayLevels)
```

qui produit une nouvelle image obtenue à partir de l'image initiale en utilisant un nombre de niveaux de gris limité, déterminé par `nbGrayLevels`. On pourra supposer que `nbGrayLevels` est une puissance de 2 comprise entre 2 et 128.

Un exemple du résultat recherché peut être observé à la figure ??.

On peut à nouveau travailler de manière assez simple : on décompose l'intervalle  $[0,255]$  en `nbGrayLevels` sous-intervalles de taille  $t$ .

Dans la nouvelle image, chaque pixel de l'image initial de couleur  $c \in [k \times t, (k+1) \times t]$  est remplacé par un pixel de couleur  $k \times t$ <sup>4</sup>.

**Q 2 .** Codez la classe `Image`

**Q 3 .** Définissez une classe `Test` disposant d'une méthode `main` qui

- crée une image  $\mathcal{I}$  (blanche) de taille (100, 200)
- crée («dessine») dans cette image des rectangles :
  - noir de taille  $20 \times 30$  à partir du point (10,30)
  - gris, niveau 64, de taille  $(20 \times 50)$  à partir du point (50,50)
  - gris, niveau 230, de taille  $(20 \times 50)$  à partir du point (20,110)
- affiche cette image, vous utiliserez la classe `ImageDisplay` fournie, et sa méthode `display`
- crée l'image obtenue en extrayant les contours de  $\mathcal{I}$  pour un seuil fixé et affiche cette image. Vous testerez plusieurs valeurs de seuil, notamment pour vérifier qu'une valeur trop importante peut entrainer la «disparition» d'une forme.

<sup>2</sup>imparfaite mais simple à mettre en œuvre

<sup>3</sup>threshold=seuil

<sup>4</sup>Avec cette méthode une image à 2 niveaux de gris ne sera pas en noir et blanc, mais en noir et gris (128)

**Q 4 .** Définissez une classe `Main` disposant d'une méthode `main` pour créer une image à partir d'un fichier au format `pgm`. Vous exploiterez pour cela la méthode `initImagePGM` fournie pour et pourrez utiliser l'un des exemples proposés dans le répertoire `images` (à placer dans le répertoire `classes` ou dans le `jar`).

Dans ce `main` vous afficherez l'image initiale et les images obtenues par extraction des contours de celle-ci et en en diminuant le nombre de niveaux de gris utilisés.

Le nom du fichier image, le seuil d'extraction de contours et le nombre de niveaux de gris à utiliser pourront être fournis comme paramètres de la ligne de commande d'exécution du programme.

Par exemple :

```
java image.Main /images/fruit.pgm 15 16
```