

INF2010 – Structures de données et algorithmes

Automne 2018

Travail Pratique 1

Héritage, polymorphisme et structure de données

Objectifs :

L'objectif visé par ce travail pratique est de vous familiariser avec les concepts orientés objets de la programmation en Java ainsi que les structures de données séquentielles. Au terme de ce travail, vous devriez être en mesure de comprendre et de manipuler les éléments suivants :

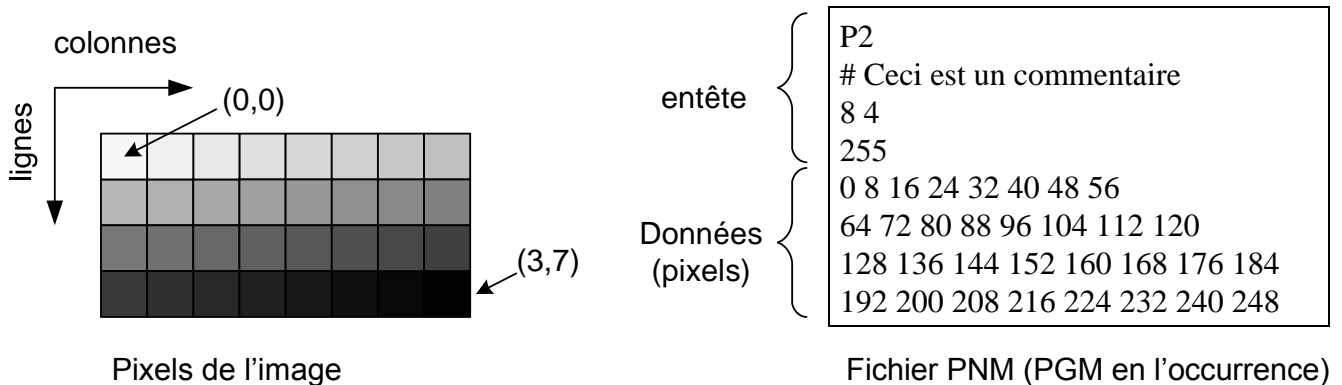
- Classe
- Héritage
- Polymorphisme
- Classes abstraites
- Liste chaînée

PROBLÈME :

Nous voulons disposer d'une librairie Java qui nous permette de manipuler des images. Ultimement, nous allons exploiter cette librairie pour sécuriser l'envoi de formulaire sur le web. Afin de simplifier notre tâche, nous utilisons le format de fichiers images dit PNM (portable anymap format). Il s'agit d'un format où les données de l'image sont transcrites en ASCII dans un fichier texte. Le lecteur intéressé pourra se documenter sur le web sur le format PNM¹. Il ne vous sera pas demandé d'écrire ou de lire ces fichiers, puisque le code nécessaire à cette tâche vous est fourni. Le code servant à afficher ces images vous est également fourni.

¹ Voir par exemple Wikipédia : http://en.wikipedia.org/wiki/Netpbm_format .

Une image peut être vue comme un tableau en deux dimensions. Chaque élément du tableau correspond à un pixel de l'image. Les coordonnées de l'image débutent en haut à gauche; le pixel à cette position est donné par les coordonnées (0,0).



On considère quatre types d'images : noir et blanc (le pixel ne peut prendre que les valeurs 0/1 (nous utiliserons un booléen)), tons de gris (le pixel prend une valeur entre 0 et 255), couleurs (le pixel est représenté par un triplet d'entiers (r, g, b) entre 0 et 255, le premier pour le rouge, le second pour le vert, le troisième pour le bleu) ou transparent (il définit une composante de plus que le pixel couleur pour représenter la transparence).

Exercice 1 : Classes abstraites, héritage et polymorphisme (1 pts)

Il vous est demandé de terminer l'implémentation des classes implémentant les quatre types de pixels considérés ici. Les classes associées aux pixels héritent de `AbstractPixel`, une classe abstraite qui vous est entièrement fournie. Les autres classes sont `BWPixel` (pour pixel noir et blanc), `GrayPixel` (pour pixel en tons de gris), `ColorPixel` (pour pixel en couleurs) et `TransparentPixel` (pour pixel transparent). La classe `BWPixel` est entièrement implémentée pour vous servir d'exemple. Il vous est demandé de compléter l'implémentation des trois autres classes. Pour ce travail, veuillez à ce que vos classes respectent le protocole de la table suivante :





		Pixel de départ			
		BW	Gray	Color	Transparent
Converti en:	BW	-	$x \leq 127 \rightarrow 0$ $x > 127 \rightarrow 1$	$\text{moyenne}(r, g, b) \leq 127 \rightarrow 0$ $\text{moyenne}(r, g, b) > 127 \rightarrow 1$	$\text{moyenne}(r, g, b) \leq 127 \rightarrow 0$ $\text{moyenne}(r, g, b) > 127 \rightarrow 1$
	Gray	$0 \rightarrow 0;$ $1 \rightarrow 255$	-	$\text{moyenne}(r, g, b)$	$\text{moyenne}(r, g, b)$
	Color	$0 \rightarrow (0,0,0);$ $1 \rightarrow (255,255,255)$	$x \rightarrow (x, x, x)$	-	$(r,g,b,a) \rightarrow (r,g,b)$
	Transparent	$0 \rightarrow (0,0,0,255);$ $1 \rightarrow (255,255,255,255)$	$x \rightarrow (x, x, x, 255)$	$(r, g, b) \rightarrow (r, g, b, 255)$	-
Négatif		$0 \rightarrow 1;$ $1 \rightarrow 0$	$x \rightarrow 255 - x$	$(r, g, b) \rightarrow (255-r, 255-g, 255-b)$	$(r, g, b, a) \rightarrow (255-r, 255-g, 255-b, a)$

Il vous est également demandé de compléter l'implémentation de la classe image : PixelMap. Il s'agit principalement d'initialisation des données et de conversion de type. À la fin de ce travail, vous devriez être en mesure d'exécuter la partie **Exercice 1** du fichier principal Main.java (mettez la partie **Exercice 2** en commentaires) et obtenir les quatre figures suivantes :



Exercice 2 : Héritage et interfaces (2 pts)

Dans cette partie, il vous est demandé de compléter l'implémentation de la classe PixelMapPlus qui hérite de la classe PixelMap et qui implémente les méthodes de l'interface ImageOperations. Le tableau qui suit explique le fonctionnement attendu de certaines des méthodes de ImageOperations.

Méthode	Description	Exemple
<code>resize(...)</code>	Redimensionne l'image aux tailles <code>h</code> et <code>w</code> . Les paramètres doivent être strictement positifs.	
<code>insert(...)</code>	Insère l'image donnée en paramètre (<code>pm</code>) à la position (<code>row0</code> , <code>col0</code>). Les pixels de l'image source qui dépassent dans l'image de destination ne sont pas recopiés.	
<code>crop(...)</code>	Découpe l'image pour qu'elle soit de hauteur <code>h</code> et de largeur <code>w</code> . Les paramètres doivent être strictement positifs. Si la nouvelle dimension est plus petite, les pixels dépassant sont perdus. Si au contraire la nouvelle dimension est plus grande, les nouveaux pixels sont blancs.	
<code>translate(...)</code>	Déplace l'image en <code>x</code> et <code>y</code> . La translation peut être négative. Les pixels non couverts sont blancs. Les pixels qui dépassent ne sont pas recopiés.	

À la fin de ce travail, vous devriez être en mesure d'exécuter la partie **Exercice 2** du fichier principal `Main.java` et obtenir ceci :



Exercice 3 : File (2 pts)

Vous devez compléter l'implémentation d'une file. Pour rappel, une file est une structure FIFO (first-in-first-out). Un élément peut donc seulement être ajouté à la fin de la file et seul l'élément en tête de file peut être retiré.

Vous allez implémenter une file en utilisant les deux structures de données suivantes :

File à partir d'un tableau (1) :

Un tableau est une séquence de cases auxquelles on accède par leur index et qui contiennent les données de la file à raison d'une donnée par case.

File à partir d'une liste chaînée (1) :

Une liste chaînée est une séquence de nœuds chaînés, c'est-à-dire d'objets distincts qui contiennent les données de la liste, à raison d'une donnée par nœud. Les implantations de listes chaînées requièrent donc généralement l'utilisation d'une classe interne `Node` contenant un champ destiné aux données et un autre destiné au chaînage.

Vous trouverez dans les fichiers `ArrayQueue.java` et `LinkedListQueue.java` la description des méthodes à compléter. La fonction `main` du fichier `QueueMain.java` vous permettra de tester vos deux classes.

Instructions pour la remise :

Le travail doit être fait par équipe de 2 personnes et doit être remis via Moodle au plus tard le :

- 27 septembre avant 23h59 pour le groupe 01.
- 20 septembre avant 23h59 pour le groupe 02.
- 26 septembre avant 23h59 pour le groupe 03.
- 19 septembre avant 23h59 pour le groupe 04.
- 01 octobre avant 23h59 pour le groupe 05.

Veillez envoyer **seulement** et **uniquement** les fichiers .java que vous avez modifiés (PixelMapPlus, GrayPixel, ColorPixel, TransparentPixel, PixelMapPlus, ArrayQueue et LinkedListQueue), dans **un seul répertoire**, le tout dans une archive de type ***.zip** (et seulement **zip**, pas de ~~rar~~, ~~7z~~, etc) qui portera le nom :

inf2010_lab1_MatriculeX_MatriculeY.zip, où $\text{MatriculeX} < \text{MatriculeY}$.

Les travaux en retard seront pénalisés de 20 % par jour de retard. Aucun travail ne sera accepté après 4 jours de retard. Si votre dépôt ne respecte pas la nomenclature définie ci-dessus, 0.5 point de pénalité sera appliqué.