TP9

Exercice – lecture d'un fichier

Le fichier `departements.txt` contient des informations sur les départements français. La première ligne de ce fichier indique le nombre total de départements. Puis, pour chaque département, le fichier précise, dans cet ordre, le numéro de département (attention aux départements de la Corse), le nom, la région, la préfecture, le nombre d'habitants et la superficie (en km^2). Chaque information est écrite sur une ligne.

1. Créez une classe `Departement` qui mémorise toutes les caractéristiques d'un département, et écrivez les méthodes qui définissent la forme canonique de la classe (*i.e.*, constructeurs, accesseurs, `toString`, `equals`, `clone`).
2. Écrivez une classe `Departements` avec :
   (a) Un constructeur qui initialise un tableau de `Departement` à partir des informations du fichier passé en paramètre (dont le format doit être semblable à celui du fichier `departements.txt`). Si la lecture du fichier rencontre un problème, une exception de type `IOException` est levée.
   (b) Une méthode `getDepartement` qui prend en paramètre un numéro de département et retourne le département correspondant. S'il n'existe pas de département avec ce numéro, une exception de type `DepartementException` (à créer) est levée.
3. Écrivez un programme de test qui :
   (a) Crée une instance de `Departements` à partir du fichier `departements.txt`.
   (b) Demande à l'utilisateur d'entrer un numéro de département et affiche dans la console les caractéristiques du département correspondant à ce numéro, et ce jusqu'à ce qu'il tape le mot `FIN` pour quitter l'application.

Exercice – écriture dans un fichier

Nous souhaitons créer des images de fractales de Julia. Il s'agit de l'ensemble des nombres complexes z_0 tels que la suite $z_{n+1} = z_n^2 + c, c \in \mathbb{C}$ ne diverge pas en module.

1. Créez une classe `Complexe` et écrivez les méthodes qui définissent la forme canonique de la classe (*i.e.*, constructeurs, accesseurs, `toString`, `equals`, `clone`).
2. Ajoutez des méthodes pour calculer le module d'un nombre complexe, et pour additionner et multiplier deux nombres complexes.
3. Pour calculer l'appartenance d'un nombre complexe à l'ensemble fractal, on fixe un nombre maximum d'itérations, ainsi qu'un module maximum. On calcule ensuite les valeurs successives de la suite. Si le module du complexe reste inférieur au module maximum après le nombre maximum d'itérations, on considère qu'il appartient à l'ensemble fractal.
   Écrivez une classe `EnsembleDeJulia` avec :
   (a) Un constructeur qui fixe le complexe c, le module maximum et le nombre maximum d'itérations.

(b) Une méthode `boolean appartient(double r, double i)` qui teste si le complexe de partie réelle `r` et de partie imaginaire `i` appartient à l'ensemble fractal.

4. Pour créer une image à partir d'un ensemble fractal défini dans tout le plan complexe, il faut savoir quelle partie du plan complexe on souhaite visualiser. Si on suppose que cette fenêtre de visualisation est carrée, elle peut être décrite par son centre et son demi-côté. On doit alors mettre en correspondance cette fenêtre de visualisation et l'image binaire (contenant des 0 et des 1) de l'ensemble fractal sur cette fenêtre de visualisation.

   Écrivez une classe `ImageFractale` avec :

   (a) Un constructeur qui fixe les coordonnées du centre de la fenêtre de visualisation et son demi-côté (dans le plan complexe), et l'ensemble fractal.

   (b) Une méthode `int getTailleImage()` qui retourne la taille de l'image (*i.e.*, le nombre de pixels de son côté). Cette taille peut être soit définie dans une constante (*e.g.*, fixée à 1000), soit donnée en paramètre à la construction.

   (c) Une méthode `int getValeur(int i, int j)` qui calcule la correspondance entre le pixel (i, j) dans $[0..tailleImage - 1]$ et les points du plan complexe correspondant. Autrement dit, cette méthode retourne 1 si le point dans le complexe de partie réelle $i * 2 * demiCote/(tailleImage - 1) + centreX - demiCote$ et de partie imaginaire $j * 2 * demiCote/(tailleImage - 1) + centreY - demiCote$ appartient à l'ensemble fractal, 0 sinon.

5. Pour obtenir un fichier image à partir d'une image fractale, une solution simple est de créer un fichier au format `PBM ascii`. Un tel fichier permet de stocker une image binaire. Il contient les informations suivantes :

   ```
   P1
   largeurImage hauteurImage
   valeur1 valeur2 valeur3 ...
   ```

   Les valeurs sont 0 (blanc) ou 1 (noir). Elles sont lues en parcourant l'image de gauche à droite et de haut en bas. Écrivez une méthode statique qui permet de sauvegarder une image d'un ensemble fractal dans un fichier `PBM`.

6. Écrivez un programme de test qui :

   (a) Crée l'ensemble de Julia avec les paramètres suivants :
   - Nombre maximum d'itérations : 100
   - Module maximum : 50
   - $z_0 = -0.8 + 0.156i$ (testez également avec $z_0 = 0.285 + 0.01i$)

   (b) Crée l'image fractale de l'ensemble de Julia avec les paramètres suivants :
   - Centre de la fenêtre de visualisation : $(0.3, 0.2)$
   - Demi-côté de la fenêtre de visualisation : 2.0

   (c) Sauvegarde l'image fractale de l'ensemble de Julia dans un fichier `PBM`.

   (d) Zomme sur une zone du plan complexe en divisant par deux le demi-côté de la fenêtre de visualisation (la taille de la fenêtre de visualisation diminue alors que la taille de l'image reste la même), puis crée et sauvegarde la nouvelle image fractale, et ce à plusieurs reprises.

7. Visualisez les images `PBM` obtenues avec un logiciel adéquat (*e.g.*, GIMP) ou en utilisant un convertisseur d'image en ligne (*e.g.*, `http://image.online-convert.com/fr/`) afin de convertir les images vers un format que vous pouvez visualiser (*e.g.*, `PNG`).

8. Nous souhaitons maintenant créer des images de fractales de Mandelbrot. Il s'agit de l'ensemble des nombres complexes défini par la suite : $z_{n+1} = z_n^2 + z_0$. Pensez à réorganiser correctement votre projet afin d'éviter les duplications de code.