Domaine Sciences et Technologies

Master Bio-informatique : DLAD

Aix*Marseille université
Socialement engagée

Année 2023-24

Programmation avancée : TP bonus 2

Code UE : SBICU05

Affichage de l'ensemble de Mandelbrot

Affichage de l'ensemble de Mandelbrot

On va travailler sur ce TP sur l'affichage de l'ensemble de Mandelbrot. Pour cela, on va utiliser un projet préexistant. Malheureusement la classe `Complex` de ce projet est bourrée d'erreurs. Le but du TP sera donc de corriger la classe `Complex` en s'aidant de tests unitaires.

Récupérer le dépôt

Comme pour le TP 2, on va utiliser git pour la gestion de versions. Il vous faut donc vous reporter aux consignes du TP 2. Le lien vers le projet à forker est le suivant : lien.

Exécuter le projet du dépôt

Pour compiler et exécuter le programme, il faut passer par l'onglet *gradle* à droite et cliquer deux fois sur `mandelbrot` -> `Tasks` -> `verification` -> `test`. Cela va compiler et exécuter les tests. Pour le moment, les tests ne passeront pas, car le code de la classe `Complex` n'est pas correct.

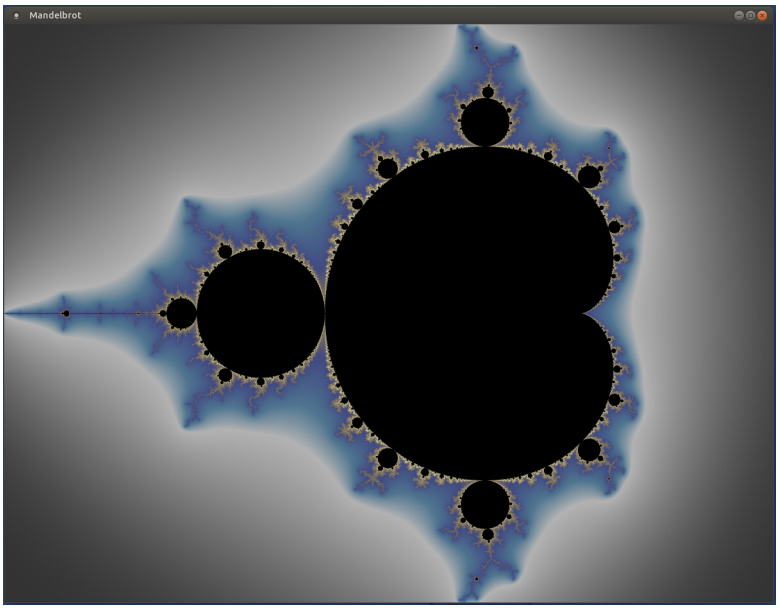
Pour exécuter l'application qui est censé afficher la fractale de Mandelbrot, il faut cliquer deux fois sur `mandelbrot` -> `Tasks` -> `application` -> `run`. Vous devriez obtenir l'affichage suivant au bout de quelques minutes (le calcul prend du temps).

Consignes pour le début du TP

Modifiez le fichier `README.md`. Mettez votre nom, votre **numéro de groupe** ainsi que le nom et le **numéro de groupe** de votre éventuel co-équipier. Faites un `commit` avec pour message “inscription d’un membre de l’équipe”, puis un `push`.

Tâches à effectuer

On cherche à corriger la classe `Complex.java` afin d’afficher correctement l’ensemble de Mandelbrot. Vous ne devez modifier que la classe `Complex.java` (sauf pour les tâches optionnelles). L’objectif du TP est d’obtenir l’affichage correct suivant :



Toutes les méthodes/classes/constructeurs/initialisation des constantes sont erronées à l’exception de la méthode `int hashCode()`.

Pourquoi des tests ?

Les calculs nécessaires à l’affichage de l’ensemble de Mandelbrot prennent plusieurs minutes alors que des tests unitaires peuvent se lancer en quelques secondes. Il est donc bien plus efficace de tester les méthodes et de ne lancer le calcul pour l’affichage que lorsque tous les tests sont passés avec succès. De plus, les résultats des tests avec les valeurs attendues donne une bien meilleure indication des erreurs que l’affichage de l’ensemble de Mandelbrot qui est dur à analyser.

Tâche 1 : corriger les méthodes pour lesquelles les tests existent

Pour cette tâche, vous devez corriger les constructeurs, méthodes et des initialisations des constantes afin qu’ils passent les tests déjà écrits dans la classe `ComplexTest` se trouvant dans le répertoire `src/test/java/mandelbrot/` :

— `Complex(double real, double imaginary)` (constructeur)
— `double getImaginary()` (méthode)
— `double getReal()` (méthode)
— `static Complex ZERO` (constante)
— `static Complex ONE` (constante)
— `static Complex I` (constante)
— `Complex negate()` (méthode)
— `Complex reciprocal()` (méthode)
— `Complex divide(Complex divisor)` (méthode)
— `Complex conjugate()` (méthode)
— `static Complex rotation(double radians)` (méthode)

Pour trouver et corriger les erreurs dans le programme, vous devrez lancer les tests. Pour cela, il faut les réactiver en enlevant les `@Disabled` avant chaque test (les tests sont désactivés par défaut dans le projet). Vous devez faire un *commit* à chaque fois que vous corrigez un élément du programme : constructeur, méthode ou initialisation d'une constante. Si dans le code des méthodes que vous avez à tester, vous utilisez une fonction qui n'est pas testée, il est plus que conseillé de la tester au préalable (voir la fin du sujet pour des explications sur les tests unitaires). Il est conseillé de ne faire des *push* que lorsque les tests passent. En effet, le projet est configuré pour lancer des tests sur le serveur et vous recevrez donc un mail de celui-ci si les tests ne passent pas.

En ce qui concerne les opérations sur les complexes, vous vous fierez à la section sur les opérations élémentaires de la page wikipedia sur les nombres complexes.

Vous pouvez vous référez à la documentation au lien suivant : https://pageperso.lis-lab.fr/~arnaud.labourel/prog_avancee_bio_info/javadoc/Complex.html pour la corriger ainsi qu'aux explications sur les tests unitaires à la fin de cette planche de TP.

Attention

Pour corriger ces méthodes, vous avez peut-être besoin de corriger d'autres méthodes.

Tâche 2 : écrire les tests puis corriger les méthodes pour lesquelles les tests n'existent pas

Pour cette tâche, vous devez corriger les méthodes suivantes (si vous ne l'avez pas déjà fait) :

— `public static Complex real(double real)`
— `public Complex add(Complex addend)`
— `Complex subtract(Complex subtrahend)`
— `Complex multiply(Complex factor)`
— `double squaredModulus()`
— `double modulus()`
— `Complex pow(int p)`
— `Complex scale(double lambda)`

Pour trouver et corriger les erreurs dans le programme, vous devrez écrire des méthodes de test dans la classe

`ComplexTest` en vous inspirant des tests déjà écrits pour les autres méthodes. Vous devez faire un commit à chaque fois que vous corrigez une méthode.

Tâches optionnelles

Quelques tâches possibles, mais optionnelles :

— Ajouter dans l'interface graphique une manière pour l'utilisateur de définir la région à afficher.
— Ajouter dans l'interface graphique une manière pour l'utilisateur de personnaliser les couleurs d'affichage de l'ensemble.