Binôme 1 : Djaber Solimani
Binôme 2 : Yaoyi Luo
Le 5/01/2019

Rapport du projet sur les ensembles de Julia

Le présent document est 1 rapport du projet de POO sur les ensembles de Julia. Il est composé de 2 parties. La première partie présente les fonctionnalités de l'application finale et la seconde les technologies utilisés et les choix de développement effectués. Aucun mode d'emploi n'est donné car l'application est réellement très simple à utiliser et intuitive. L'archive julia.zip a été obtenu en exportant 1 projet Java fait sur Eclipse nommé «julia2». L'importation de ce projet sur Eclipse suffit donc pour exécuter l'application.

1. Fonctionnalités

L'application permet de dessiner des images représentant des ensembles de Julia ou de Mandelbrot. Elle est disponible à la fois avec 1 interface graphique et sans (on l'utilise alors directement sur la console).En «mode console», l'utilisateur peut saisir le nombre complexe de départ, le nombre d'itérations à effectuer les dimensions de l'image et le nom du fichier contenant l'image.Il obtient en sortie 1 fichier au format png contenant l'image de l'ensemble correspondant situé dans son projet Java.
L'utilisation avec la GUI (interface graphique) est similaire mais aucun fichier n'est enregistré et l'image a 1 taille fixe. L'utilisateur peut en revanche zoomer et dé-zoomer sur l'image et se déplacer. Il peut également afficher autant d'ensemble qu'il souhaite sans devoir relancer l'application.

2. Technologies utilisées et choix de développement

2.1. API utilisées

L'interface a été développé avec l'API **JavaFx** réputée pour sa qualité.Nous avons également utilisé 1 fichier fxml (basé sur 1 syntaxe xml) pour définir plus simplement l'interface. L'application est multi-thread afin d'augmenter la vitesse d'affichage des images. Nous nous sommes servi pour cela de l'API **ForkJoinPool** et notamment de la classe abstraite **RecursiveAction** qui en plus de sa simplicité et de sa performance se prêtait bien à la structure du logiciel.

Binôme 1 : Djaber Solimani
Binôme 2 : Yaoyi Luo
Le 5/01/2019

2.2. Architecture du logiciel

La classe **Complexe** définit les informations liées aux nombres complexes.
La classe **EnsembleDeJulia**, contenant 1 nombre complexe possède 1 classe imbriquée **Dessin** héritant de **RecursiveAction**.On C'est dans cette classe que sont effectuées les calculs et les mises à jour liés à la couleur des pixels. On divise le
L'ensemble de Mandelbrot étant 1 cas particulier d'ensemble de Julia, nous avons choisi que la classe **EnsembleDeMandelbrot** hériterait de EnsembleDeJulia et redéfinirait la méthode indiceDivergence en partant du complexe nul.
La classe **Controleur** contient les composants graphiques de la GUI ainsi que les méthodes liées aux événements utilisateurs sur ces composants.Elle contient également 1 objet de type EnsembleDeJulia. Lorsque l'utilisateur appuie sur le bouton de validation, cet objet est instancié en 1 nouvel ensemble (de Julia ou de Mandelbrot) en fonction des paramètres saisis.
Enfin, la classe **Main** contient la fonction main, qui demande de choisir d'utiliser l'application en mode console ou en mode graphique.
L'application est organisée en 2 packages selon le principe Modèle-Vue-Controleur : il y'a 1 package **Modele** contenant la partie «Données» de l'application avec les classes Complexe, EnsembleDeJulia et EnsembleDeMandelbrot et 1 package **VueEtControleur** contenant les classes Controleur et Main (sachant que la classe Controleur gère les 2 aspects, nous avons voulu simplifié en utilisant qu'1 seul package).

2.3. Choix de développement

Afin de favoriser l'encapsulation et d'éviter l'aliasing, la plupart des attributs sont final private (sauf ceux de EnsembleDeJulia qui sont protected pour l'héritage) avec des accesseurs et des mutateurs.Les entrées utilisateurs sont toutes gérées par des exceptions afin d'éviter des problèmes lors de l'exécution.