Haute école d'ingénierie et d'architecture Fribourg
Hochschule für Technik und Architektur Freiburg

Filière Informatique 2020-2021

Intégration des nombres complexes et des Unum en Java avec COJAC

Classe I3

Cahier des charges

Travail de bachelor

Version : 0.1

Student : Cédric Tâche

Professor : Frédéric Bapst

Expert : Baptiste Wicht

Table des versions

Version	Date	Auteur	Description
0.1	01.06.2021	Cédric Tâche	Création

Table des matières

1 Introduction

La plupart des langages de programmation offrent les mêmes types pour stocker des nombres. Il s'agit notamment des nombre entiers et des nombres réels. Ces nombres réels sont toujours en virgule flottante.

Ces types ont tout de même des limitations. Les nombres sont limités en taille et au domaine du réel. Quant à eux, les nombres à virgule flottante sont inexacts. Par conséquent, lorsque les erreurs s'accumulent, le résultat d'un calcul peut être complètement faux sans qu'on le sache.

Pourtant, d'autres alternatives existent. Les nombres complexes sont couramment utilisés en mathématique et en physique. Quant à eux, les nombres réels peuvent aussi être représentés avec des Unum.

2 Contexte

En Java, aucun type ni aucune classe dans le JDK permet de gérer des nombres complexes ou des Unum.

Si on veut ajouter ces types, on pense naturellement à créer ces objets. Cependant il est aussi possible de **TODO: ...**

COJAC [1] est une librairie Java permettant de modifier les capacités arithmétiques d'un programme Java sans en modifier le code.

3 Objectifs

Le but de ce projet est d'ajouter deux nouvelles fonctionnalités à COJAC [1].

3.1 Intégration des nombres complexes

Une option de COJAC permettra de changer le comportement des calculs dans l'application de l'utilisateur. Les calculs se feront ainsi avec des nombres complexes.

3.2 Intégration des Unum

Une option de COJAC permettra de changer le format de stockage des nombres réels. Les Unum seront utilisés à la place de la virgule flottante.

3.3 Démonstration des deux fonctionnalités

Des démonstrations seront réalisées pour les montrer l'utiliser des deux fonctionnalités.

4 Objectifs secondaires

4.1 Mise à jour des librairies

COJAC [1] utilise plusieurs vielles librairies. Ce serait bien de les mettre à jour si possible.

4.2 Tests de performance

Des tests de performance peuvent aussi être effectués pour tester l'efficacité de l'implémentation.

5 Tâches

Cette section détaille les tâches qui devront être effectuées pour réaliser chacun des objectifs définis précédemment.

5.1 Intégration des nombres complexes

Les tâches suivantes seront effectuées pour réaliser cet objectif :

- Analyse des implémentations existantes.
- Analyse de COJAC et de comment intégrer un nouveau type.
- Analyse des nombres complexes.
- Concevoir l'algorithme.
- Implémenter l'algorithme.
- Tester l'algorithme.

5.2 Intégration des Unum

Les tâches suivantes seront effectuées pour réaliser cet objectif :

- Analyse des implémentations existantes.
- Analyse de COJAC et de comment intégrer un nouveau type.
- Analyse des Unum.
- Concevoir l'algorithme.
- Implémenter l'algorithme.
- Tester l'algorithme.

5.3 Démonstration

Les tâches suivantes seront effectuées pour réaliser cet objectif :

- Imaginer une situation pour les démonstrations.
- Réaliser les démonstrations.

5.4 Mise à jour des librairies

Les tâches suivantes seront effectuées pour réaliser cet objectif :

- Chercher les mises à jour de chaque librairie.
- Regarder les changements avec la nouvelle version.
- Modifier la version et adapter le code.
- Tester les modifications.

5.5 Tests de performance

Les tâches suivantes seront effectuées pour réaliser cet objectif :

- Analyser les tests de performance actuels.
- Réaliser les tests de performance.

6 Planification

La figure 1 montre le diagramme de Gantt avec la liste des jalons et des tâches.

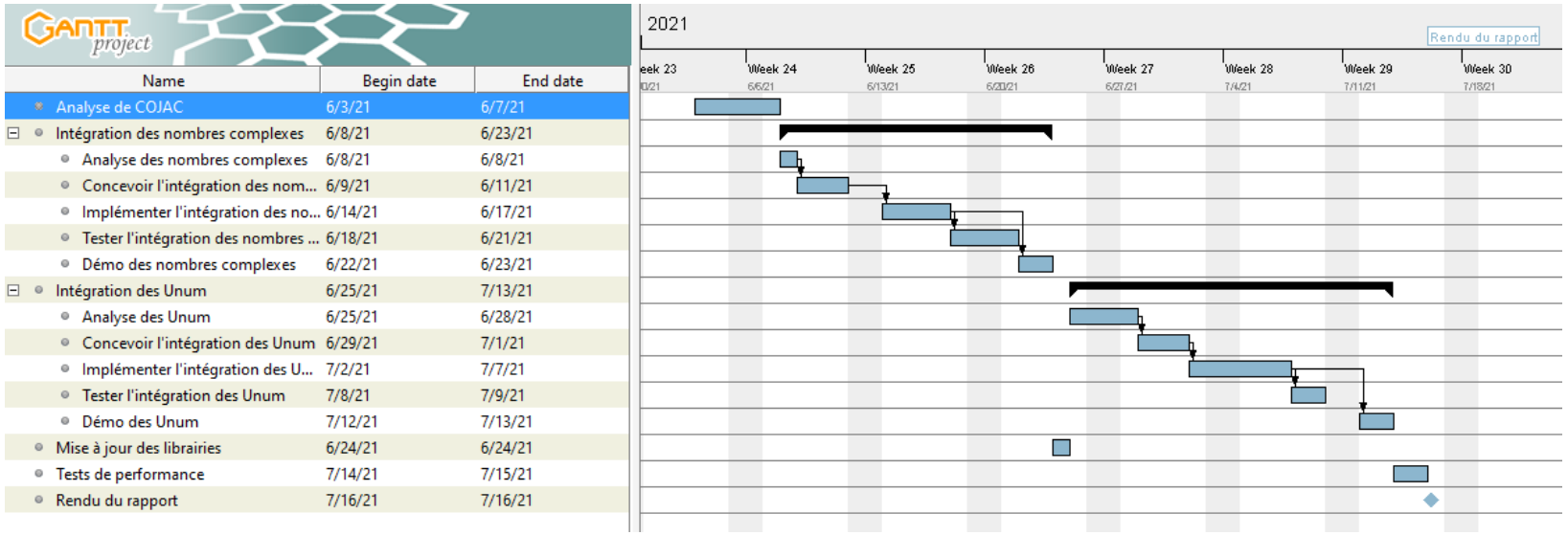

FIGURE 1 – Diagramme de Gantt

7 Références

[1] Cojac - boosting arithmetic capabilities of java numbers. `https://github.com/Cojac/Cojac`, 2019.