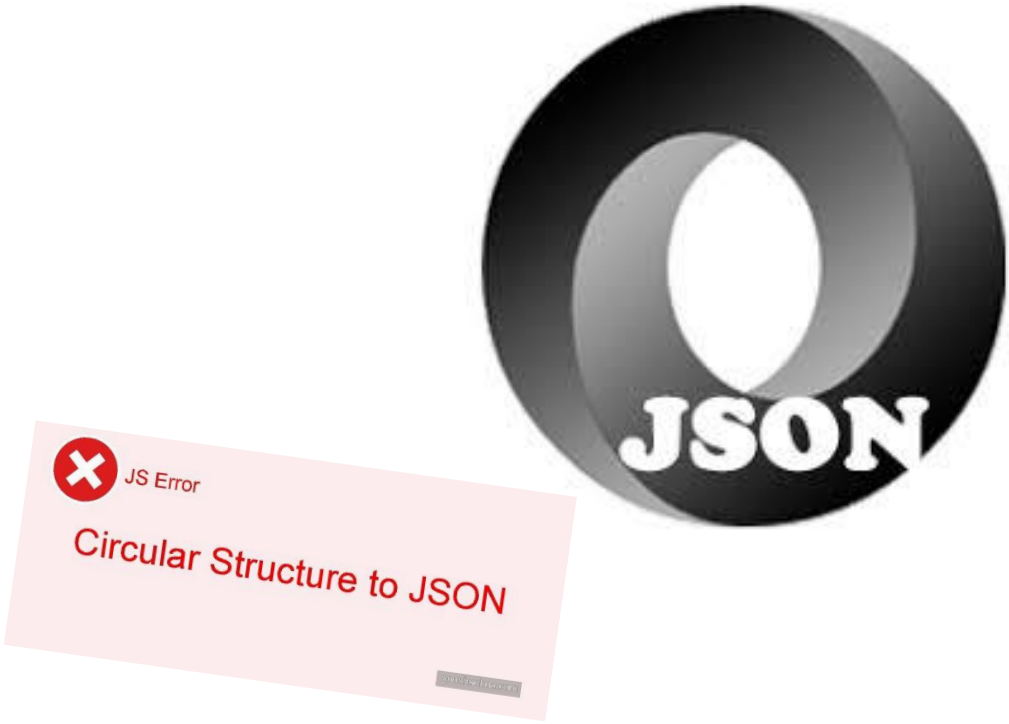
Rapport de Projet : SAE32_2022

Réalisation d'un Inspecteur de JSON



Réalisé par :

Julien CHARBONNEL
Claire GOBERT
Félix BRINET

Encadré par :

Monsieur HERNANDEZ Luc

Table des matières

Introduction :

Le projet consiste à créer un programme en Java pour faciliter la lecture de fichiers JSON en utilisant un affichage clair et de la coloration syntaxique. La première phase consiste à afficher de manière claire le contenu JSON à la console. Tandis que la seconde phase est d'afficher dans une interface le fichier JSON avec une coloration. Quant à la troisième, il faut faciliter l'exploration des contenus JSON en dépliant ou repliant les tableaux et objets. Pour finir la 4e phase, il faut ajouter à l'interface, la possibilité de basculer d'un affichage JSON à un affichage PHP.

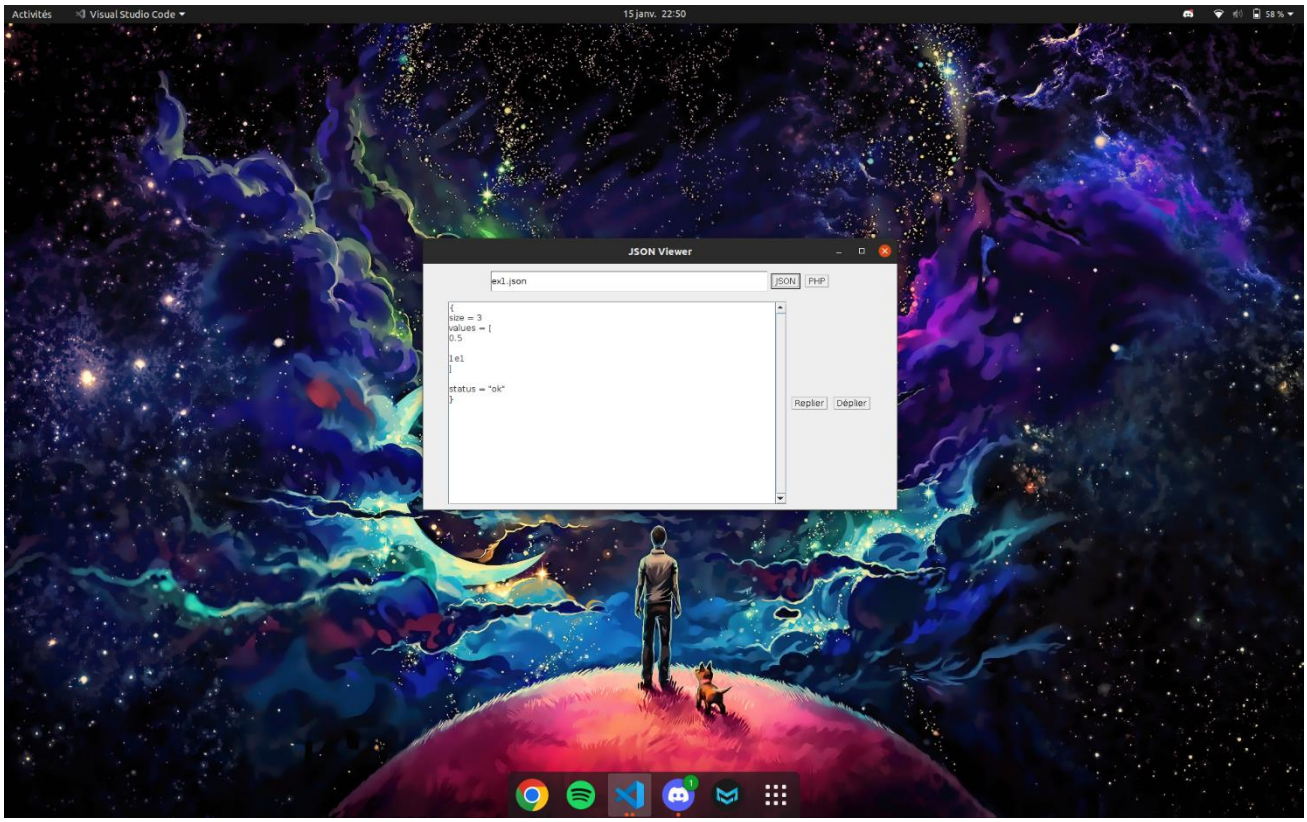
Organisation de l'équipe :

Nous avons tout d'abord créé un Trello pour définir toutes les tâches à faire et un groupe discord pour faciliter la communication. Julien a commencé la première phase du projet avec l'arbre de syntaxe abstraite pendant que Claire et Félix commençaient la partie graphique. Julien a rencontré des difficultés durant la phase 1, il s'y est repris à multiples reprises pour créer l'AST, donc Claire et Félix ont essayé de l'aider. Nous avons eu du mal à comprendre le fonctionnement de l'arbre de syntaxe malgré nos demandes d'aide à Monsieur Hernandez et nos camarades.

Cependant, pas loin de la date finale du rendu du projet, nous avons réussi à créer un arbre de syntaxe abstraite afin de lire un contenu JSON. Comme quoi il ne faut jamais abandonner et persévérer en programmation.

Fonctionnalités :

- Pour exécuter la phase 1 du projet (uniquement l'affichage en console) vous pouvez exécuter la commande suivante : **make phase1, ensuite pour lancer la phase 1 faire java –jar phase 1 ex1.json**
- Pour exécuter le programme avec l'interface, vous pouvez utiliser la commande : « **make phase2 puis pour lancer l'interface graphique : java –jar phase2** »
- Si l'utilisateur met en argument quelque chose, cela lui envoie un message d'erreur en lui prévenant de ne pas mettre d'argument lors de l'exécution. (Sauf pour la phase 1)
- Une fois le programme exécuté avec la commande java –jar phase2, la fenêtre suivante apparait :



- Sur la fenêtre, il y a un **champ de texte** pour que l'utilisateur rentre l'url ou le chemin local du fichier JSON.

- A côté, il y a un bouton pour **« valider** » le fichier que l'on veut rendre l'affichage plus agréable.
- Il y a également un bouton pour **« recharger »** pour réafficher le fichier JSON en cas de changement du contenue.
- Nous pouvoir remarque un bouton **« PHP »** pour transformer le fichier JSON en affichage PHP.
- Après avoir rentré le fichier JSON dans le champ et avoir cliqué sur le bouton « valider », un champ de texte apparaît avec le fichier JSON. L'utilisateur ne peut pas modifier le champ de texte car cela a été désactivé.

Structure du programme :

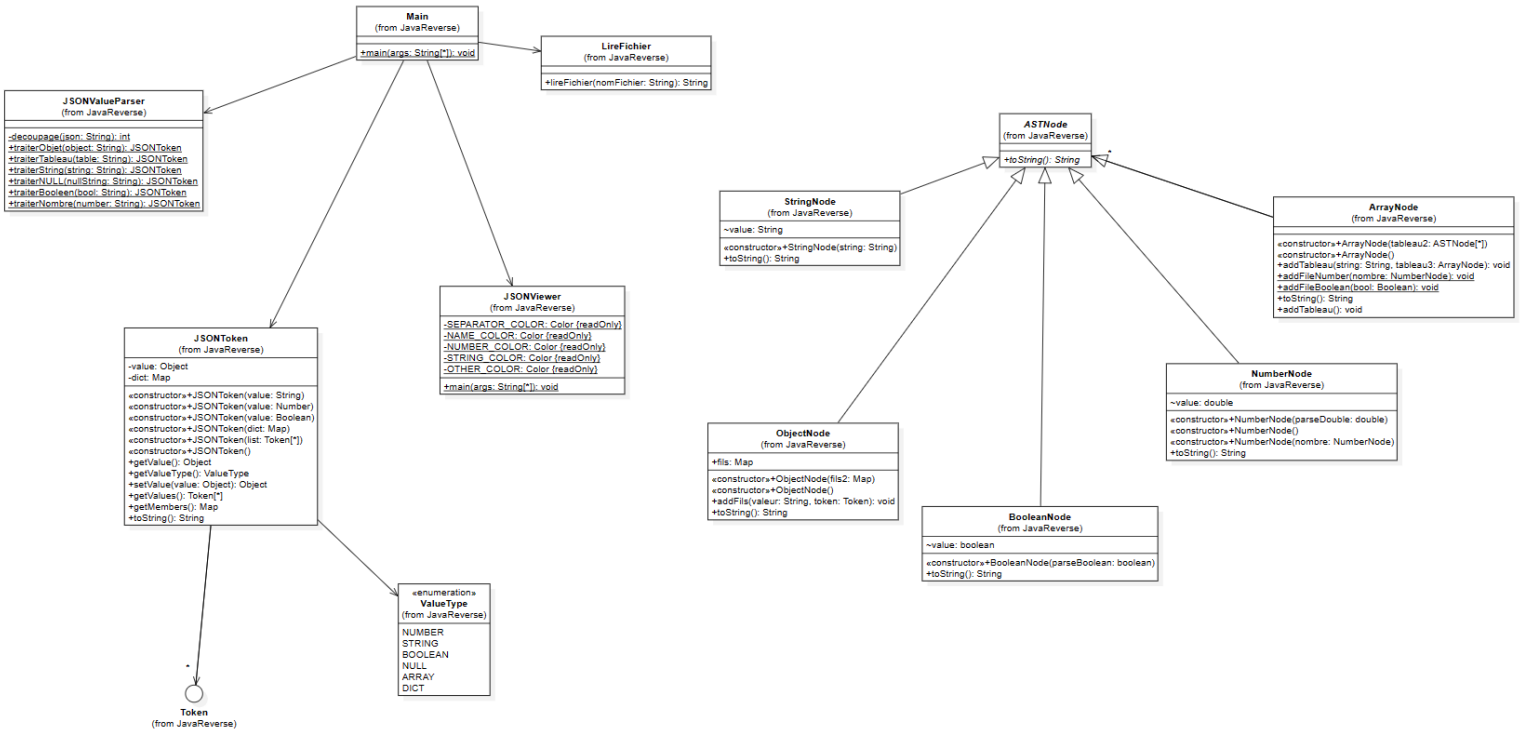
Au début, nous avons créé plusieurs classes que nous avons laissées de côté, mais qui nous ont permis de mieux comprendre la structure de notre programme. Donc nous avons préféré les garder dans le git, afin que vous ayez une vision de nos anciennes tentatives. Et on préférait les garder si par la suite, on voudrait améliorer ou changer notre programme.

A ce jour le programme est divisé en plusieurs classes qui forment l'arbre de syntaxe pour la partie modèle, les fichiers back-end sont stockés dans modèles et compilés dans build. Nous avons une classe JSONValueParser qui s'occupe de découper le fichier caractères par caractères, comme vous nous l'avez conseillé afin d'éviter de lire le fichier ligne par ligne ce qui serait plutôt compliquer à exploiter. L'interface Token qui représente un token JSON. Nous avons une autre classe JsonToken qui implémente Token, cette classe permet de créer un token JSON à partir d'une chaîne de caractères.

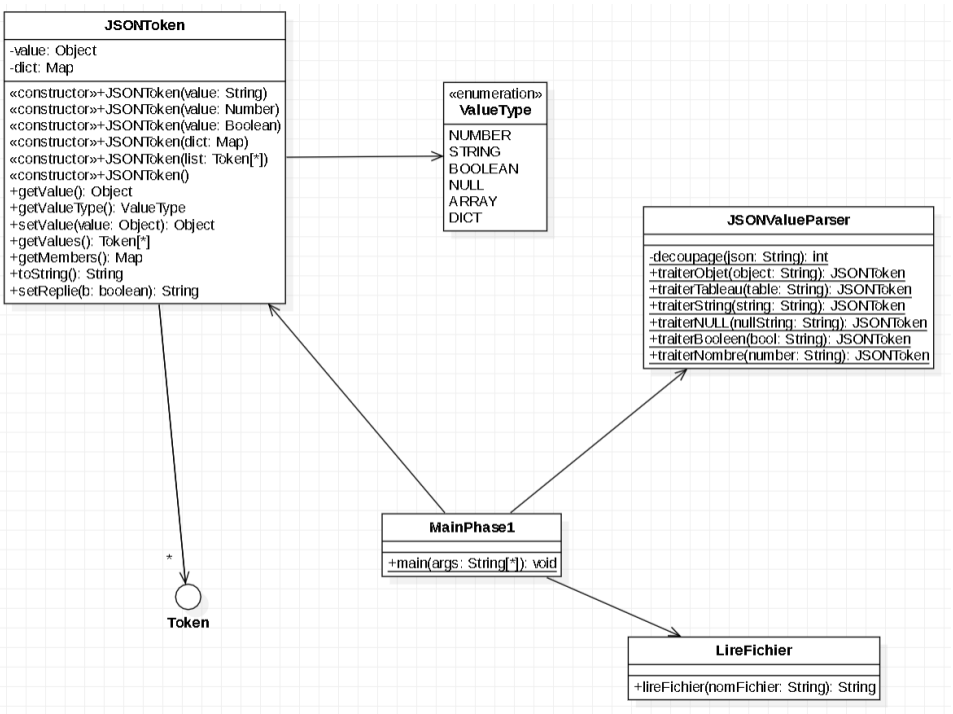
Une classe LireFichier qui possède une méthode lireFichier() ce qui nous permet de récupérer une chaine de caractères du fichier json lu.

Une classe ValueType qui permet une énumération des types de valeurs. Par exemple, si la valeur lue est un nombre, un string, un booléen, etc.

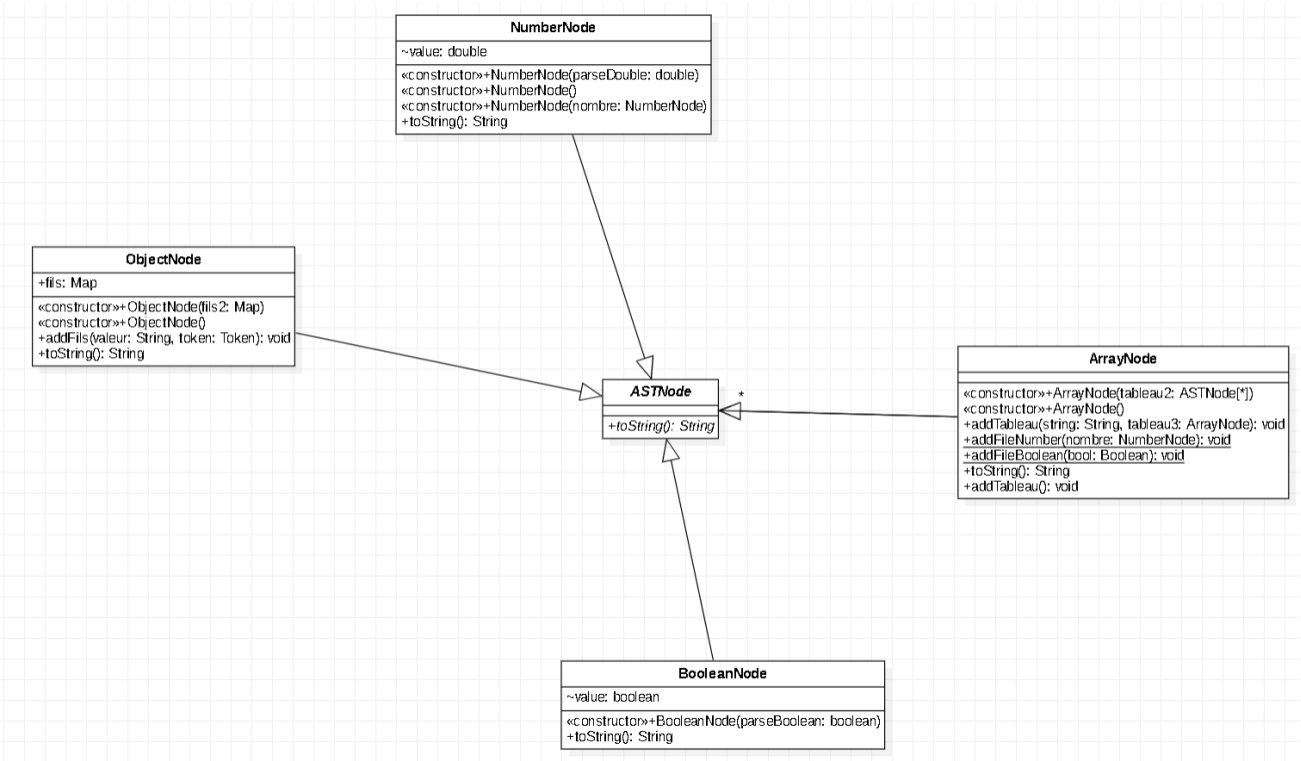
Diagramme de classe général du projet



DC de la phase 1 :



DC d'une ancienne version de l'arbre de syntaxe abstraite



Explication arbre syntaxe abstraite :

Concernant l'arbre de syntaxe abstraite, Julien l'a recommencé 6 fois avant d'avoir quelque chose de propre et raisonnable, il a essayé de séparer au maximum le code en plusieurs classes afin d'avoir du code pour de la Programmation Orienté Objet.

Nous avons une classe pour l'arbre en lui-même ; une autre pour les nœuds des objets, car une accolade ouvrante dans du JSON, est considéré dans notre code comme la création d'un nouvel objet ; puis une autre classe pour les tableaux afin de pouvoir y stocker toutes les valeurs qu'on veut dedans ; dans une file afin de garder l'ordre d'entrer des valeurs ; puis une classe pour toutes les différentes valeurs des nœuds : par exemple une classe pour le nœud des chaînes de caractères, une autre pour les nœuds des nombres, une pour les booléens, etc.

Enumération des structures de données abstraites :

- Classe abstraite : ASTNode.java
- Interface : Token.java

Nous avons utilisé des dictionnaires afin d'avoir des pairs clé valeur, un String associé à un token.

Une liste afin d'avoir une liste entières des éléments du token.

Synthèse du projet

Nous avons donc un affichage d'un contenu JSON de meilleure qualité, plus simple et plus intuitif. L'interface est simple mais fonctionnel. Le projet Agile précédent nous a permis de mieux nous connaître, tandis que ce projet a approfondi notre cohésion de groupe. Si nous devions le refaire, nous aurions commencé tous ensemble sur l'arbre de syntaxe abstraite. Nous sommes satisfaits du résultat obtenu et nous avons apprécié faire ce projet.

Conclusions personnelles

Conclusion de Félix :

Ce projet a été une expérience très enrichissante pour ma part. J'ai pu m'améliorer en programmation Java mais aussi sur git en utilisant les branches. La première difficulté a été de comprendre l'arbre de syntaxe abstraite et puis je regrette de ne pas avoir pu plus aider Julien. Pour autant, grâce à mes compétences j'ai pu me rendre utile sur l'affichage graphique, sur la rédaction du rapport et le diagramme de classe. Pour finir, je suis content d'avoir pu travailler avec Julien et Claire, nous nous sommes très bien entendus.

Conclusion de Julien :

Cela a été un véritable défi pour moi concernant le back-end de notre application, j'ai pu découvrir comment créer une application d'inspecteur de JSON de A à Z. Cela m'a permis de découvrir comment peut-être construit une bibliothèque déjà existante pour le JSON en JAVA ou dans d'autres langages de programmation. J'ai adoré travailler en équipe avec mon groupe, on a continué à adopter l'approche agile au sein d'un projet, que nous avons découvert au sein de nos différents cours sur ce sujet. Ce qui m'a beaucoup plus, je trouve ça par la même occasion plutôt efficace pour un projet peut importes la taille de celui-ci. Une équipe très agréable, c'est plaisant d'étudier dans des conditions comme celle-ci.

Conclusion de Claire :

Ce projet était très intéressant, le fait de retravailler des personnes que je connaissais a permis de faciliter le partage des tâches. Une bonne cohésion d'équipe permet un travail efficace. Au début du projet, j'ai eu du mal à comprendre l'arbre de syntaxe abstraite mais au fur et à mesure du projet j'ai pu aider Julien. En attendant que l'arbre fonctionne j'ai travaillé sans et le fait de passer à l'AST a été un peu difficile. J'ai vraiment aimé travailler avec mes camarades Julien et Félix. Je pense qu'avec une meilleure gestion du temps nous aurions pu faire un travail plus abouti.