# Conclusion d'utilisation de JavaCC et AST d'Eclipse

1. Facilité d'implémentation

   Selon nous, la différence entre les expériences d'utilisation des deux outils est grande. Si on peut considérer les deux outils comme deux langages de programmation, JavaCC est plus comme un langage de bas niveau (peut-être plus facile à comprendre pour les compilateurs), et les AST d'Eclipse se comparent à un langage de haut niveau.

   Quand le JavaParser effectue l'analyse syntaxique (JavaCC), il faut définir les visiteurs pour trouver les nœuds sur lesquels on désire faire les opérations (comme recueillir des statistiques sur les différents attributs, etc.).

   Pour les AST d'Eclipse, c'est un autre cas. Pour accéder au nœud désiré, il existe des fonctions prêtes à être utilisées dans les librairies. C'est facile à utiliser et la principale difficulté est de trouver les bonnes fonctions.

2. Avantages et inconvénients

   JavaCC :

   Avantages : Plus de liberté lors de l'écriture du code, et plus de liberté pour l'implémentation des fonctions.

   Inconvénients : Lorsqu'il y a beaucoup de code (s'il y a des millier lignes de code), la seule façon de trouver la définition d'un méthode est 'ctrl+f'.

   AST d'Eclipse :

   Avantages : L'AST View fournit une structure claire de l'ensemble du programme qui reflète intuitivement la relation entre les classes et les méthodes.

   Inconvénients : On ne peut pas obtenir certains données qui ne peuvent pas être accédées directement par les méthodes dans la librairie.

3. Impact d'une modification du langage Java

   Lorsqu'on veut refactoriser ou optimiser du code, nous devons d'abord l'analyser et le comprendre. Pour se faire, il est préférable d'utiliser une représentation d'arborescence basée sur la structure hiérarchique du code du programme. Avec AST, nous pouvons voir les mots, les phrases et les blocs dans l'arbre de syntaxe abstraite de manière plus pratique et intuitive.