# Appréciations

Pour donner suite à votre utilisation de JavaCC et des AST d’Eclipse, donnez votre avis sur les deux implémentations en fonction de critères tels que : facilité d’implémentation, avantages et inconvénients, impact d’une modification du langage Java sur ce que vous avez réalisé, etc. (maximum une page).

## Miriam Davydov

**Avantages :** personnalisation

**Inconvénients :** répétition, difficile à maintenir, nouvel apprentissage

En mon sens, l’avantage primaire de JavaCC et AST c’est la personnalisation d’un *parser* qui répond mieux à nos besoins, mais tout en étant un avantage, celui-ci se transforme en inconvénient quand on conçoit que nos besoins ne sont pas si différents des autres programmeurs. Leur utilisation implique de recréer la roue et d’apprendre une syntaxe additionnelle rendant son usage compliqué.

Ces outils sont connus pour être difficile à maintenir aujourd’hui remplacé par des plugins maintenus par une équipe ou une communauté.

## Maxime Desmarais

Pour ma part, j’ai trouvé JavaCC très lourd à utiliser pour une grammaire qui fonctionne pour n’importe quel programme valide Java. Quoi qu’il puisse s’avérer très utile pour façonner de petits langages personnalisés, la taille finale du projet est un indice de sa complexité pour Java. Pour ce qui est des AST, je trouve que c’est une idée très intéressante mais qui nécessiterait un peu de peaufinage quant à son implémentation, la majorité des problèmes provenant de son initialisation. Cependant, une fois partis, AST s’avère beaucoup plus comprenable et facile d’utilisation comparé à JavaCC, pour des programmes Java évidemment.

## Médoune YADE

POINTS POSITIFS : Javacc est un outil open source pour générer des analyseurs à partir de spécifications de grammaire écrites dans la langue de grammaire Javacc. AST est un cadre pour représenter et manipuler les arbres de syntaxe abstraits (AST) des programmes Java. Javacc fournit une interface facile à utiliser et intuitive pour développer des analyseurs pour les langages de programmation. AST fournit un moyen pratique d'analyser et de transformer les programmes.

Javacc offre un ensemble complet de fonctionnalités pour créer des générateurs d'analyseurs puissants. AST est utilisé pour développer des extensions de langage et des refactorisations, ainsi que pour analyser et visualiser le code. Javacc utilise une approche orientée objet pour créer des analyseurs, ce qui facilite le maintien et l'extension. AST est bien adapté pour les programmes à grande échelle et peut être intégré à d'autres outils comme IDE.

POINTS NÉGATIFS : Javacc est difficile à maintenir et à déboguer, car il n'est pas conçu pour fonctionner avec d'autres bibliothèques Java. AST n'est pas toujours aussi à jour que Javacc, et est également difficile à utiliser lorsqu'il s'agit de fonctionnalités linguistiques complexes. JAVACC ne fournit aucun support intégré pour les fonctionnalités linguistiques telles que les génériques ou les annotations, ce qui peut être important pour certaines applications. AST n'est pas bien adapté à la programmation générique, ce qui peut être une limitation majeure dans certaines applications. Javacc ne prend pas en charge l'analyse incrémentielle, ce qui peut être une limitation majeure lorsqu'il s'agit de grandes bases de code. AST n'est pas bien adapté pour les optimisations spécifiques au langage telles que l'analyse du flux de données ou l'inférence du type.

## Marlyn Momo Fopossi

Avantages : Adaptation

Inconvénients : Difficile à apprendre, et à configurer Contrairement à java cc qui est quand même lisible, facile à comprendre et commun. AST avec sa manipulation de nœuds est un peu complexe. Chaque nœud AST appartient à une instance AST unique, appelée AST propriétaire. Les enfants d'un nœud AST ont toujours le même propriétaire que leurs nœuds parents. Si un nœud d'un AST doit être ajouté à un AST différent, le sous arbre doit d'abord être cloné pour garantir que les nœuds ajoutés ont le bon AST propriétaire. Ce sont des propriétés complexes à manipuler.