System Analysis - CPSC 499

Assignment 2: Parse Trees vs. AST-Based Frameworks

Team: A2-3

Mandeep Kumar 30162674,

Mann Patel 30182233,

Nicolas Amaya 30206444

1. **How difficult were these evolutions?**

LCA to LCA	Difficulties	Difficulty grade (1-5)
LCA-J1.4→ LCA-JP1.0.0	The main difficulty was to translate our original ANTLR class analysis tool into JP, which has a completely different approach, as we are relying on an AST framework and using a pre-built system. Probably what made this challenging was not being familiar with JP.	4
LCA-J1.4 → LCA-J8	As the grammar expanded, it was restructured. Declarations became more context-dependent, reflecting new constructs like annotations, enums, lambda expressions, and type parameters. For example, the classDeclaration() rule was no longer the only entry point, so you had to account for enumDeclaration() and other forms of type declarations. This meant the Local Class Analyzer had to navigate multiple alternative rules to correctly extract the same information.	3
LCA-JP1.0.0 → LCA-JP3.27.0	The main difficulty was identifying the changes from JP1.0.0 to JP3.27.0. So we could successfully update our JP to the new specifications.	2

1.1 Is there much of a difference?

Yes! Big Time, after migrating the analyzer to JavaParser, maintaining and extending it became much simpler, since we don't have to maintain

the JavaParser. So we no longer need to manually update or regenerate a grammar every time a new Java version introduces syntactic changes.

This means that in the future, any updates to the analyzer tool will be focused on adapting the analyzer logic to new updates to the API, so even in the worst-case scenario where things are changed dramatically, the changes are isolated only on our analyzer tool, which is much easier to maintain.

2. **How do you predict that the difficulties will/will not change moving forward to the modern versions of the grammar and of JavaParser or even into the future, and why?**

   Based on this exercise, I feel like modern versions will have fewer difficulties than older versions, which is the case since newer versions are already working within JP, making them easier to port into a newer version. As times go by, I predict that these transitions should grow progressively smoother because the Java language will mature more and become more polished and more standardized, which means that as time goes by, the changes will be less and less dramatic.

3. **Given the choice, which makes more sense to use: an ANTLR-format grammar or an AST-based framework, and why?**

   AST-based framework, the benefits of not having to maintain the Java grammar and being able to focus more on the analysis side of things are too great, and that is without mentioning potential issues that only exist when you try to maintain your own grammar, like worrying about your parser's correctness or compatibility. In conclusion, AST not only removed the overhead of dealing with the javaparser complexity, but it also enables scalability since it greatly reduced the dependencies of our analysis tools.