System Analysis - CPSC 499

Assignment 1: Create a Grammar Report

Team: A1-1

Mandeep Kumar (??),

Mann Patel 30182233,

Nicolas Amaya 30206444

In the process of creating this grammar, this was truly an interesting experience. We created a Lexer, Grammar, and two test files in Java. Let's begin by sharing some experiences after working on this project:

Bad:

1. Initially, when I started working on this project, I thought it would be amazing to build rules from my head, highly overestimating the complexity of this project, which helped me realize how little I know about the inner workings of a language.
2. When I started reading the jls2 and was trying to implement the grammar from each section of the book, which went badly really fast since it's not presented in a way that you can directly translate, and to be completely honest, I didn't notice Chapter 18th yet.
3. One of the sweetest and sour moments was finding ‘:’ which was supposed to be a ‘;’ breaking everything since the test code that I was running would hit that line pretty fast, pretty much giving the impression that everything was broken.
4. Translating the JLS2 to ANTRL was a battle; it appeared to be very straightforward, but some lines were unnecessarily unclear, and it did force me to really understand what was going on and why it made sense.
5. Understanding how the order of rules in the parser file influences the grammar's behavior was one of the most difficult things I had to deal with when using ANTLR v4. I initially misjudged the extent to which structure and placement could affect parsing results. This was especially challenging for me because many mistakes felt unpredictable in the absence of a clear mental model. The pivotal moment occurred following a chalkboard demonstration in which the instructor described in detail how ANTLR assesses grammar rules. I filled in the "missing link" for myself by seeking clarification both during and after that conversation.

Good:

1. After understanding how rules are built, it was pretty easy and satisfying to see your grammar take shape as you fix each problem one at a time, and your tree grows and grows.
2. Best moment hands down was when the grammar stopped crashing IntelliJ 🙂

The Things We Do

We generated the parser using Java as the target language with the antlr-4.13.2-complete.jar as well as inteliJ. Following instructions from SJSU's ANTLR Installation Guide, we set up ANTLR successfully. which made generating the parser significantly easier.

Translating Java grammar into ANTLRv4 required careful attention to left-recursion issues. Initially, the grammar caused frequent recursion errors. To resolve this, we broke down the problematic sections into smaller sub-grammars, which allowed ANTLR to process each component cleanly. In chapter 6 of JLS 2 was extremely helpful in understanding the syntactic constructs for local classes. We added the assert keyword manually to the grammar, ensuring compatibility with Java1.4.

The analysis tool was written in Java code , builds the parse tree for each input file and traverses it to identify local classes. It successfully:

- Printed the correct count of local classes.
- Reported each class with its file, line, and column.
- Handled multiple files as input.
- Rejected files with syntax errors.

How To Run Code

Generate Parser and Lexer

```
java -jar antlr-4.13.2-complete.jar -Dlanguage=Java ExprLexer.g4 ExprParser.g4
```

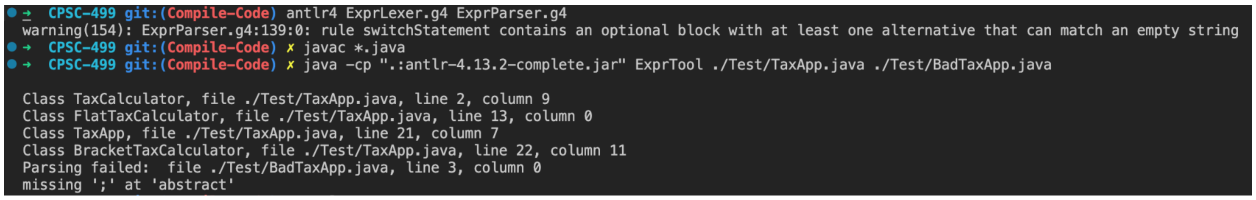
Compile All Java Files

```
javac -cp ".:antlr-4.13.2-complete.jar" *.java   # use ; instead of : on Windows
```

Run The Analysis Tool (ExpTool)

```
java -cp ".:antlr-4.13.2-complete.jar" ExprTool ./Test/TaxApp.java ./Test/BadTaxApp.java
```

The Output Should be as follows



```
● → CPSC-499 git:(Compile-Code) antlr4 ExprLexer.g4 ExprParser.g4
  warning(154): ExprParser.g4:139:0: rule switchStatement contains an optional block with at least one alternative that can match an empty string
● → CPSC-499 git:(Compile-Code) ✗ javac *.java
● → CPSC-499 git:(Compile-Code) ✗ java -cp ".:antlr-4.13.2-complete.jar" ExprTool ./Test/TaxApp.java ./Test/BadTaxApp.java

  Class TaxCalculator, file ./Test/TaxApp.java, line 2, column 9
  Class FlatTaxCalculator, file ./Test/TaxApp.java, line 13, column 0
  Class TaxApp, file ./Test/TaxApp.java, line 21, column 7
  Class BracketTaxCalculator, file ./Test/TaxApp.java, line 22, column 11
  Parsing failed:  file ./Test/BadTaxApp.java, line 3, column 0
  missing ';' at 'abstract'
```

Resources

- ANTLR official documentation: https://www.antlr.org/
- SJSU ANTLR Installation Guide: https://www.cs.sjsu.edu/~mak/tutorials/InstallANTLR4.pdf
- Java Chapter 6 - Crafting Interpreters Language Specification, Second Edition (JLS 2), especially Chapter 6
- UOIT Compiler class CSCI 4020U - Building a Lexer in Antlr Building lexer in Antlr