**编译项目报告**

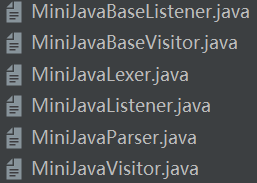
卫艺璇 15307130433

王炜越 15307130349

1. **ANTLR 4与Yacc/Bison工具的对比**

* ANTLR生成LL(k)解析器，而Yacc和Bison都生成LALR解析器。LL(k)语法相对于LALR更加强大，也没有shift-reduce，reduce-reduce类似的语法冲突错误。
* ANTLR生成代码后可读性很高，因其全部封装在对应的class中。在语法的parser上，ANTLR比较清晰的switch/case来匹配token，类似于手动书写一个DFA，而在Yacc中使用的是parser table。并且对于switch的效率问题，因为编译器对于switch是有优化的，所以效率上没有很大影响。
* ANTLR提供了对于tree construction、tree walking和translation的支持，这对于我们话抽象语法树有很大的帮助。
* ANTLR的错误处理机制很灵活，其使用exception-driver实现，exception的最小粒度能得到具体的token，方便我们在语法解析时控制系统的错误处理。
* ANTLR对于不同语言有很好的支持，对于一个rule，我们可以很方便地加入一段代码，同时可以很方便地在规则之外加入类的成员变量、类的成员函数以和全局的变量和函数等。
* ANTLR不仅功能更强、容易扩展和开源，而且ANTLR生成的代码和使用递归下降方法（手工生成分析器的主要方法）生成的代码很相似，易于阅读理解。而基于LR分析法的Yacc分析器生成工具生成的程序就比较晦涩。此外我们可以在文法描述中插入特定的语义动作，告诉ANTLR怎样去创建抽象语法树和怎样输出（对我们PJ的完成很有帮助）。

1. **项目流程**
2. 首先书写一个MiniJava.g4文件，该文件中定义了ANTLR中Parser和Lexer的相关定义。
3. 在命令行执行。。。。。命令来编译MiniJava.g4文件。成功执行之后，可以得到如下几份文件，



1. 根据自己需求更改这些文件，并再新建一个MiniJavaAnalyze.java文件利用这些文件生成抽象语法树和一个ErrorListener.java文件来进行错误处理。
2. 然后利用javac xxx.java编译对应.java文件生成相应.class文件，最后使用java MiniJavaAnalyze运行编译好的MiniJavaAnalyze.class文件。
3. 测试：输入MiniJava格式代码，ctrl+D结束输入，该编译器会输出抽象语法树图，并以log形式在命令行中进行错误显示。
4. **源代码分析**

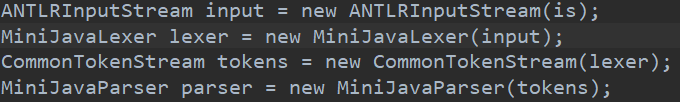
3.1 **源代码结构**

源代码中包含9个文件：

* ErrorListener.java：我们用来错误处理的代码。
* MiniJavaAnalyze.java：核心代码，用来生成抽象语法树。
* MiniJavaBaseListener.java：。。。。。。。
* MiniJavaBaseVisitor.java
* MiniJavaLexer.java：ANTLR中的Lexer处理文件
* MiniJavaListener.java：
* MiniJavaParser.java：ANTLR中的Parser处理文件
* MiniJavaVisitor.java：

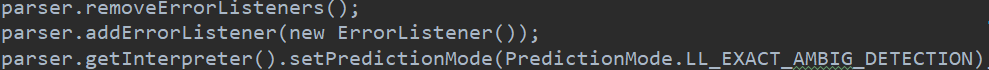
**3.2 核心代码工作原理**

1. 在核心代码中，我们首先使用ANTLR中的lexer和parser处理文件得到一个Parser Tree：



。。。。。。加个图片。。。。

1. 错误处理，通过将原本ANTLR中的错误处理文件更替为我们的错误处理文件：ErrorListener.java；此外并通过parser.getInterpreter().setPredictionMode()函数将文法中所有有二义性的地方都显示出来：



1. 接下来是生成抽象语法树AST：

先利用parser.goal()生成一个ANTLR Parser树的根节点ParserRuleContext，然后调用generateAST画出相应抽象语法树。

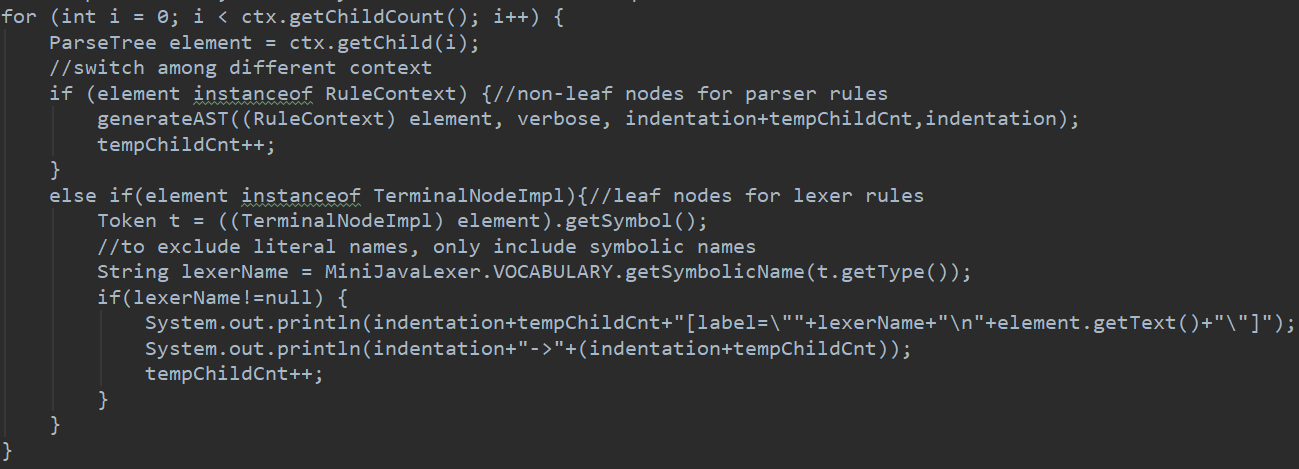


画抽象语法树的主要思想是：

对ANTLR生成的Parser树进行递归遍历。

Parser树有两种节点：叶子节点和非叶子节点。如果遍历到非叶子节点，则继续递归调用generateAST向下遍历；如果遍历到叶子节点，则记录叶子节点的三个信息：Sequence、Type、Text。三者分别为：Sequence：结点的层次序号，例：根节点是0，根节点的直接子孩子是1，下一任子孩子是2，以此类推。一个序号可能有多个结点；Type：记录节点的类型，属于哪一类；Text：记录结点的具体内容。

在画图时，根据节点的Sequence序号进行两点之间的连接，例：根节点goal的序号是0，第一任子孩子main class的序号是1，第二任是2，等等。每次将具有相邻序号的结点进行连接，最后就可以生成一个抽象语法树。



........加个图。。。。。。

1. 接下来进行测试

3.3 工作中遇到的问题及解决

1. **额外功能的说明与项目感想**

项目感想：

通过本次项目，我们更加理解了编译器的工作原理，将ANTLR生成的Parser、Lexer树进一步抽象为机器更好理解的抽象语法树。。。。。