MiniJava Compiler Project Report

13307130414 夏亦婧

一、词法/语法生成工具

本实验中，使用ANTLR词法/语法生成工具。

使用ANTLR作为词法/语法生成工具，我们只需要编写ANTLR的语法文件，描述我们要解析的语言的语法，ANTLR就能够自动生成语言的解析器。ANTLR的解析过程分为两个阶段，第一阶段进行词法分析(lexer)，将源文件读入后分解为符号（token），第二阶段进行解析（parser），生成语法树（或称解析树）。

ANTLR生成的解析器为递归下降解析器（recursive-descent parser），是自顶向下解析器（top-down parser）的一种。解析过程是从语法树的根开始向叶子（token）递归。

二、代码结构

1. base/
    1. 给出MiniJava的语法定义文件(MiniJava.g4)，参考BNF for MiniJava
    2. 由ANTLR生成的若干java源文件和tokens文件
2. resources/
    1. 给出用于测试的测试文件
3. scope/
    1. 对scope进行定义，并生成ClassScope、VarScope等类
4. CheckPhase.java
    1. 对生成的解析树进行遍历，重写部分函数，实现对继承类是否存在的判断
5. Errors.java
    1. 对除ANTLR自带的错误处理以外的其他错误进行输出
6. Compiler.java
    1. 含main函数的运行文件，调用ANTLR生成的lexer、parser，生成解析树的gui界面
    2. 调用CheckPhase，实现添加的错误处理功能
    3. 对ANTLR的自带错误输出，添加下划线输出的部分

三、错误处理与修复

1. 词法错误：为ANTLR默认行为，如提示多余符号、词法匹配不正确等
2. 语法错误：为ANTLR默认行为，如
3. 语义错误：实现了对类继承是否存在的检查，如果继承的类不存在，则会产生错误提示。

```
line 18:0 The class extended 'NoClass' does not exists.
class BS extends NoClass{
^
```

4. 错误修复：使用ANTLR默认的错误修复（single-token insertion和single-token deletion）
5. 其他：对错误的显示中，在ANTLR原本给出的错误提示基础上，给出当前错误行的内容，并在错误的地方进行下划线的显示，使错误显示更为详尽。

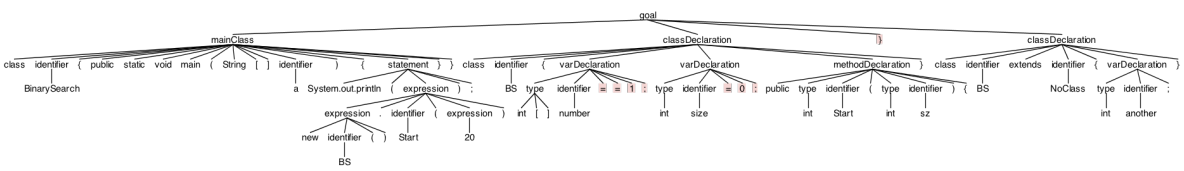
```
line 11:17 mismatched input '=' expecting ';'
    int[] number == 1;
                  ^
```

四、输出示例

输出的文本：

```
[alicexia:src alicexia$ java Compiler test.txt
line 11:17 mismatched input '=' expecting ';'
    int[] number == 1;
                  ^
line 12:13 mismatched input '=' expecting ';'
    int size = 0;
             ^
line 15:4 mismatched input '}' expecting {'{', 'return', 'int', 'boolean', 'if', 'whil
e', 'System.out.println', IDENT}
    }
    ^
line 16:0 extraneous input '}' expecting {<EOF>, 'class'}
}
^
line 18:0 The class extended 'NoClass' does not exists.
class BS extends NoClass{
^
```

输出的抽象语法树：



五、项目感想：

知道了ANTLR提供的两种遍历方式，对ANTLR通过Listener方式进行遍历且自己加入错误处理的部分有了初步的了解。知道了编译器在对代码进行解析的过程中通过的几个阶段，以及使用的错误处理的方式。
在写代码时，从scope开始对各个方面都进行了思考，但是最后没有实现，以后可能会再进行补充和实现，将学习到的知识实践到代码中。