MiniJava Compiler

13307130265 杨越

1.环境配置与依赖

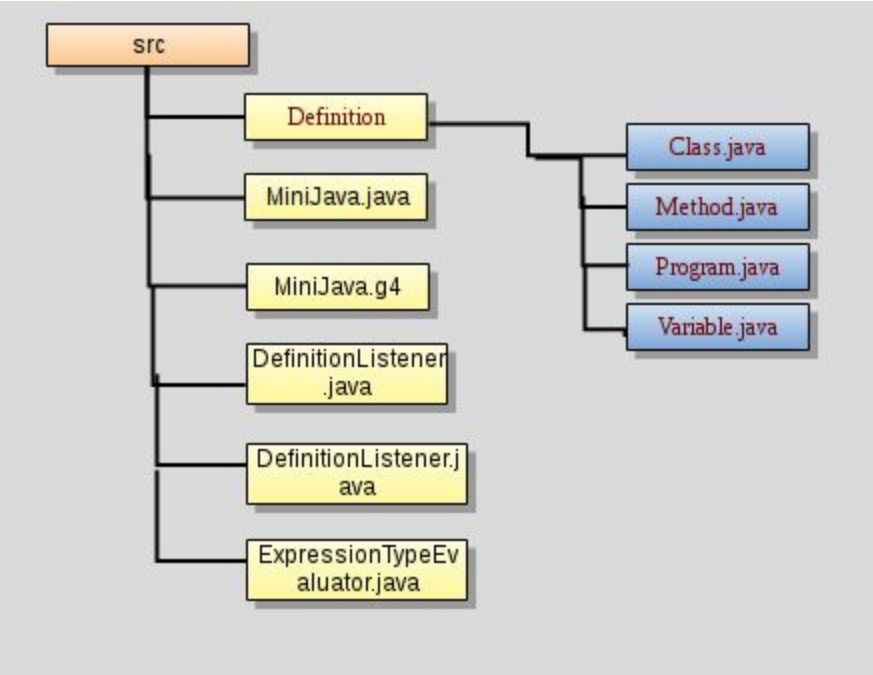
平台: Linux Debian 3.16.36-1+deb8u2.
Java: openjdk version "1.8.0_111"
./build.sh 编译整个项目
./run.sh correct_minijava （跑correct_minijava里所有正确的minijava程序）
./run.sh incorrect_minijava （跑incorrect_minijava里所有有错误的minijava示例程序）
若需要查看语法树，则需要在run.sh里加上-gui的命令.

2.架构设计与工具选择

为了实现MiniJava编译器，我选择了ANTLR4作为我的Parser工具。这个工具能有效、简洁的算出所有的tokens。可以通过对ANTLR生成的Listener & Visitor Override, 简单有效的完成我想要的操作。ANTLR对于每个Token都相当于树上一个节点，可以通过对语法的设计在程序中得知当前的节点类型是什么。

为了代码的简洁美观，我决定对于MiniJava里的所有item进行封装，对Class, Method, Program, Variable进行封装。好处是可以简洁代码的实现，例如对Class进行定义时，会包含Method和Variable, 这样可以有效的解决模块之间的依赖关系，保证代码的可读性。

3.代码结构



Definition里为所有MiniJava语法中各个模块的实现

1. Program.java为对整个MiniJava程序的描述, 包含了主类是谁，有哪些class被定义在程序中，且记录他们的名字。
2. Class.java为对MiniJava里的类的描述。包含了Class scope下的Varibales有哪些，有哪些methods定义在class中， 继承了哪个class
3. Method.java为对MiniJava里的方法的描述。包含了Method这个scope下的所有Variables, 以及返回类型，参数的类型是什么。
4. Variable.java为对MiniJava里的变量的描述。包含了变量名，变量类型。

检查MiniJava程序的正确性，我们需要判断如下

1. 程序里是否有主类。
2. 程序里的类名是否有冲突。
3. 程序里的类是否会出现循环extends的情况。
4. 每个类里的变量名是否会有冲突。
5. 每个方法内的变量名与参数名会不会有冲突。
6. 赋值时候左右的兼容性
7. 执行条件语句时，结果需要是boolean
8. 数组取地址时，需要的int.

为了实现上述过程，我的实现是对PaserTree进行两次遍历。第一次遍历将Program.java模块里的实例初始化好。DefinitionListener.java 是我对此的实现。它通过继承了由ANTLR生成的MiniJavaBaseListener这个类，重写了我需要的方法，这样在我进入某些节点时，可以对我想要

的东西例如program进行更改。MiniJavaBaseListener为实现了MiniJavaListener这个接口的类，这样我们只需要在对树遍历时，注入我们生成好的definitionListener就可以通过接口达成我们对程序进行初始话的目的。我重写了nterClassDeclaration, enterVarDeclaration, enterMethodDeclaration这三个method，于是利用definitionLIstener.java对程序遍历后我们得知了不考虑Statement时，程序是可以正确编译。

考虑到Statement时，最主要就是变量类型检查。包括调用的变量是否有定义，变量赋值时是否左右兼容。为此，我实现了StatementListener.java, 也是extends了MiniJavaBaseListener. 我在MiniJava.g4中，对statement进行了分类，利用一些技巧可以得出当前是什么类型的statement.

对于statement里的一些expression需要用ExpreesionTypeEvaluator进行表达式类型的求解。ExpressionTypeEvaluator是extends了MiniJavaBaseVisitor的类，实现了visitExpression这一特殊的method。里面使用了复杂的逻辑判断当前的表达式的类型是什么。

4. 流程

主要流程在MiniJava.java里. 首先使用ANTLR里的inputStream接收stdin的内容。利用ANTLR得到Tokens, 并且得到Parser和ParserTree.
之后在ParserTree上用DefinitionLIstener进行一次遍历。
在之后就是用statementListener再进行一次遍历。

5. 错误报告

在MiniJava类设置了一个静态的方法，叫做printError. 传入的参数为Token，还有ErrorMessage. ANTLR提供的Token包含了Token所在的行与列这个信息。每次碰到错误时，将当前的Token与ErrorMessage一起用MiniJava内的PrintError调用。

错误的信息处理会明确告诉用户，哪一行那一列，为什么会发生错误。

例如

```
1  class Factorial{
2      public static void main(String[] a){
3          System.out.println(new Fac().ComputeFac(10));
4      }// just a comment ignore it
5  }
6
7  class Fac {
8      public int ComputeFac(int num) {
9          int num_aux;
10         if (num < 1)
11             num_aux = false;
12         else
13             num_aux = num * (this.ComputeFac(num - 1)) ;
14         if (num + num && 3)
15             num = 3;
16         else
17             num = 4;
18         return true ;
19     }
20 }
```

在14行时，会出现 && 左右符号不兼容。
错误监听器就会输出 Error, line 14:16 left and right must both be boolean for &&.

6.感想与不足

写了MiniJava的编译器后，对编译整个流程的理解更为深入了。比较可惜的是MiniJava过于简单，比如不能在某个scope内新定义一个变量。这样就少了可持久化的数据结构的应用

在设计的时候，没有考虑清楚如何输出错误信息。导致所有错误信息的判断全部耦合在Listener里，代码显得像一坨屎。妥善的方法是写一个ErrorListener, 用于监听当前的错误，识别错误的信息。README.md