**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **编译技术** |
| **学生姓名：** | **陈卓文** |
| **学生学号：** | **201936380215** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2021-2022第一学期** |

**软件学院**

**2021年11月**

# 语法分析

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7 楼 | | 233 房； | | **实验台号：** | |  | |
| **实验日期与时间：** | | | 2021/11/25 | | | | **评 分：** | |  | |
| **预习检查纪录：** | | |  | | | | **实验教师：** | | 应伟勤 | |
| **电子文档存放位置：** | | |  | | | | | | | |
| **电子文档文件名：** | | |  | | | | | | | |
| **批改意见：** | | |  | | | | | | | |
| 出勤情况（20%） | 课堂态度（20%） | | 实验报告质量（60%） | | | | | |
| 任务完成情况 | | 源代码注释 | | 报告内容 | |
| □ 出勤  □ 缺勤  □ 早退  □ 请假 | □ 认真  □ 不认真 | | □ 完成  □ 未完成 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | | □ 优秀  □ 良好  □ 通过  □ 未通过 | |

# 报告内容

1、实验目标

实现一个decaf parser

2、实验环境

Ubuntu 20.04 , OpenJDK 11

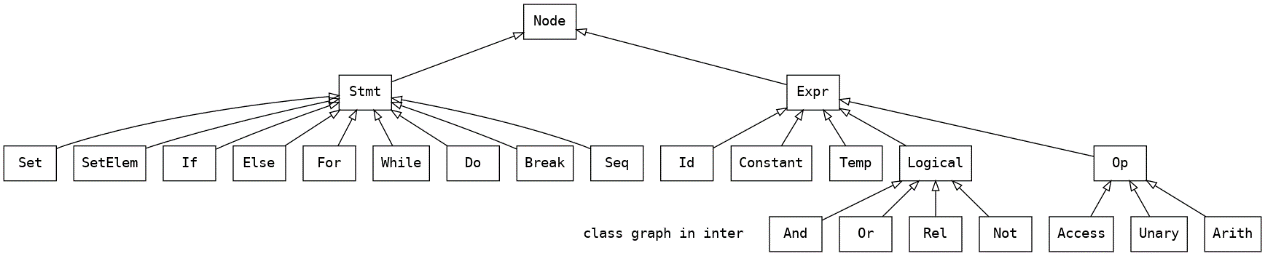
3、关键数据结构和核心算法

3.1 数据结构

源代码核心类有两个，Lexer及Parser

其中Lexer负责处理Token分解，Parser负责构建语法树

Inter中储存了语法树中所需要的Node，其中继承关系如下



3. 2 核心算法

第一部分是Token分解，所用代码及实现原理属于第一次实验内容。

第二部分是语法树构建，由于Decaf的语法，构造时只需要从头到尾线性扫描一次，且后面的Token不会影响前面已生成的节点。

Parser具有多个成员函数用于生成节点，采取递归下降的方式进行语法树生成。这些函数从Token串提取数据，并返回与函数名所对应的节点，有需要的基于语法递归调用其他节点函数。例如block函数处理代码块；stmt处理单条语句；bool处理表达式。

最后Main通过调用Parser的program函数得到整个代码的语法树。再通过语法树的display成员函数打印出来。

本次实验在原框架上增加的功能有添加语法树节点类型Inter.For ，类型定义：

public class For extends Stmt {

Stmt first;

Expr second;

Stmt third;

Stmt stmt;

public For() {}

public void init(Stmt fir, Expr sec, Stmt thi, Stmt st) {}

public void display() {}

}

还增加了Parser中的stmt中的for分支判断：

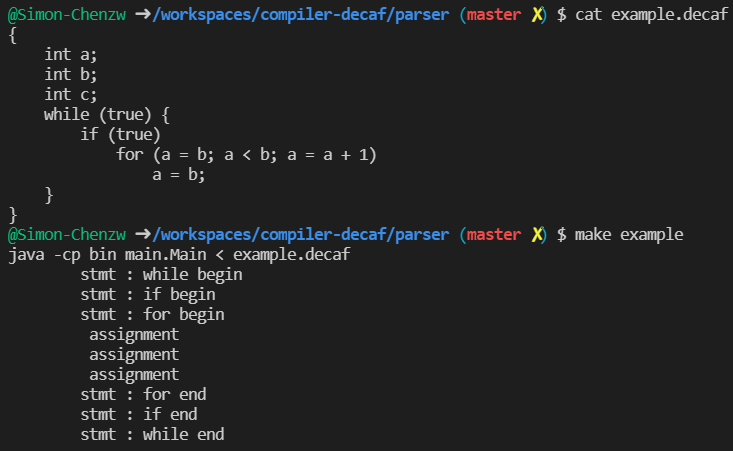
case Tag.FOR:// stmt -> for (assign;bool;assign) stmt

3. 3 错误处理

主要的错误抛出地点有两个

1. Parser的成员函数error：负责处理语法树构建时的错误，例如：
   1. 依据语法规则必须出现的括号、分号等字符没有出现
   2. 赋值语句没有变量名
2. Inter.Node的成员函数error：负责处理语法树节点的非法结构，例如：
   1. If、For、While、Do-while等循环中的判断语句不是布尔表达式
   2. 将两个不可运算的类型进行运算时

3.4 运行结果





4、总结

从这次实验中，我掌握了decaf的语法，并且学会了从Token串生成语法树的方法。通过阅读源码掌握了如何搭建一个lexer及parser的框架。

有个建议或者说疑惑，为什么实验内容中有一项是对代码添加注释。我觉得代码的理解程度与注释的多少不是那么的相关，注释是为了帮助他人理解代码中的难点，而不是表达自己的理解。同时也很难掌握什么是“难点”，很难判断这段代码算不算难点。