**实 验 报 告**

|  |  |
| --- | --- |
| **课程名称：** | **编译技术** |
| **学生姓名：** | **石望华** |
| **学生学号：** | **201630676843** |
| **学生专业：** | **软件工程** |
| **开课学期：** | **2017-2018第一学期** |

**软件学院**

**2017年12月**

# 实验三 Decaf语义分析

实验四 中间代码生成

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **地 点：** | | | B7 楼 | | 233 房； | | **实验台号：** | | 003 | |
| **实验日期与时间：** | | | 2017/12/20 | | | | **评 分：** | |  | |
| **预习检查纪录：** | | |  | | | | **实验教师：** | |  | |
| **批改意见：** | | |  | | | | | | | |
| 出勤情况 | 课堂态度 | | 实验报告质量 | | | | | |
| 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

实验三：

（1）实现decaf语言的中间代码生成程序，添加for语句的语义分析和中间代码生成。

（2）实现类型检查。

实验四：

生成三地址代码。

2、实验环境

Eclipse

3、关键数据结构和核心算法

3.1 数据结构

哈希表:用于word的快速匹配（关键字）。

3.2 核心算法

各种递归程序。

各种类的继承结构，组合。

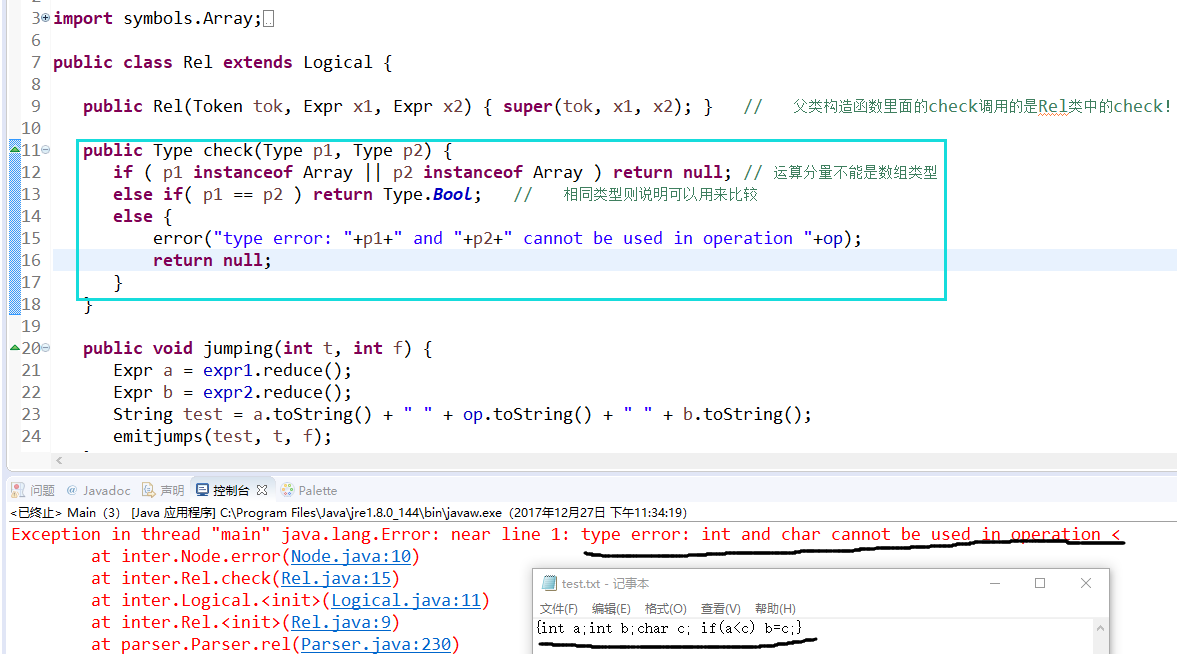
作用域的合理搭配。

**parser核心算法。**

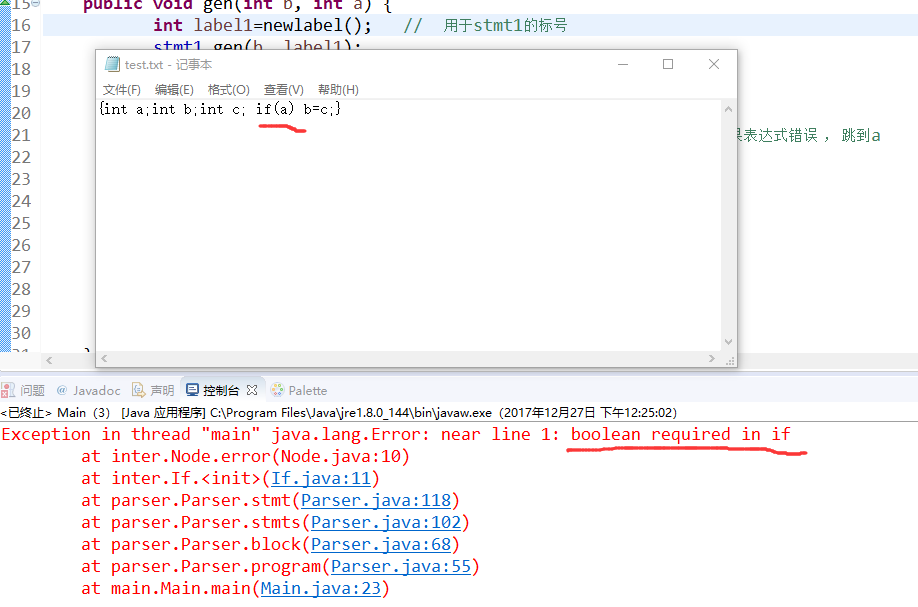
**与bool表达式有关的一连续方法。**

3.3 错误处理

1. 类型检查：操作数需要类型兼容。

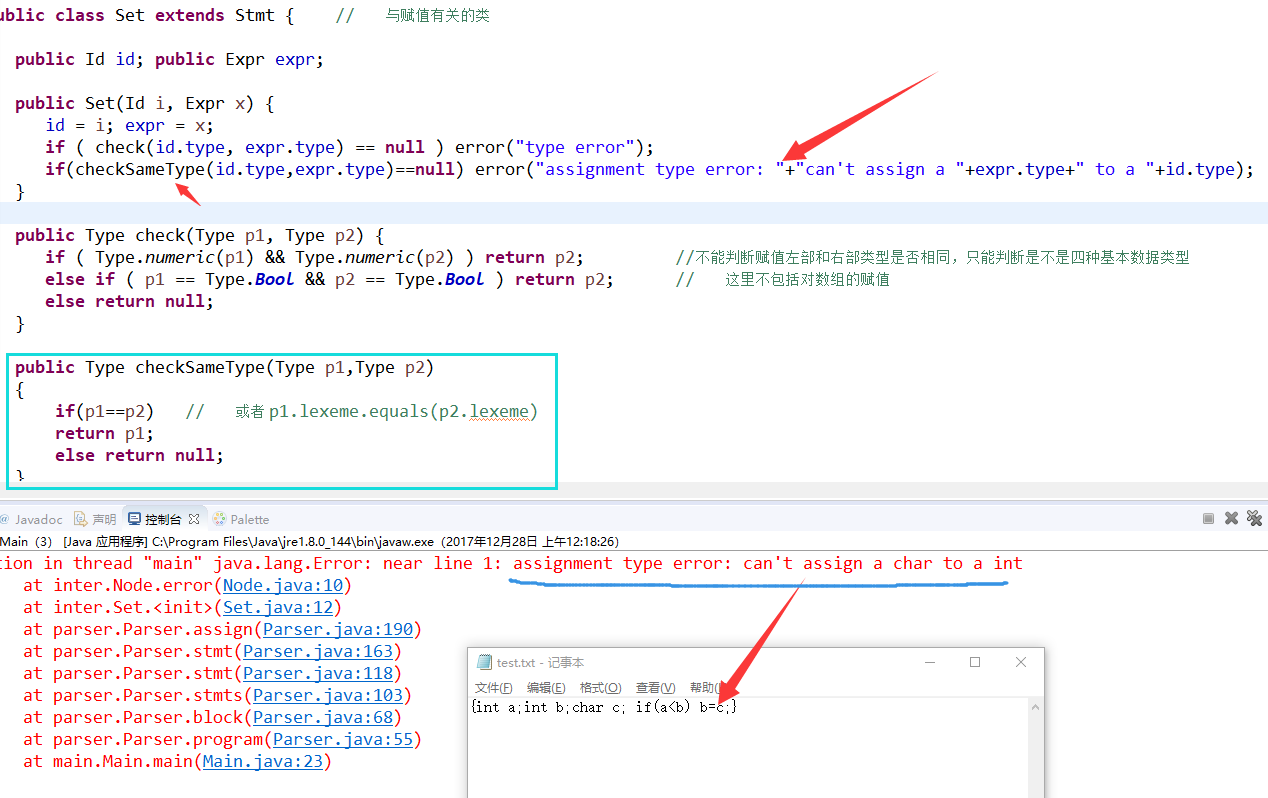


(2)类型检查：条件表达式需要是bool类型。



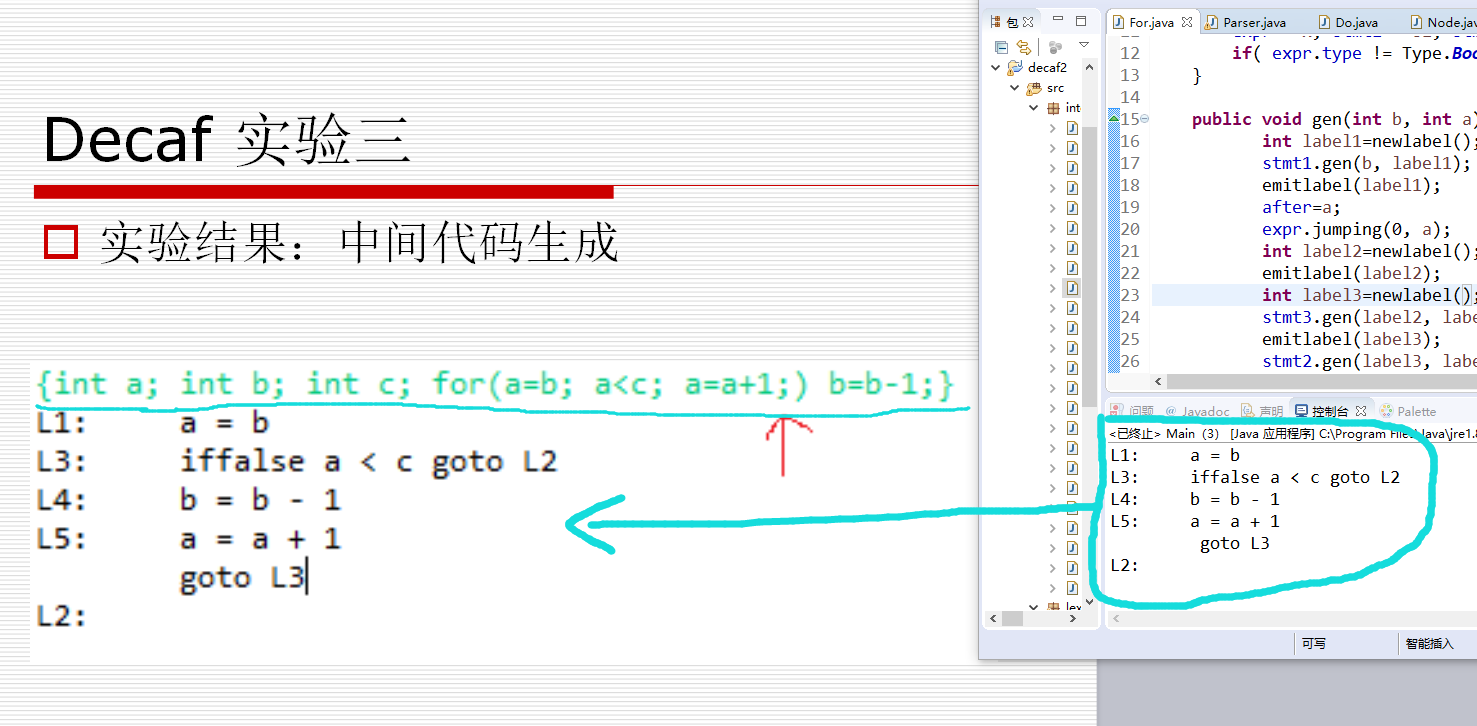
（注：这个检查是在If类中完成的）

（3）类型检查：赋值左部和右部类型相同

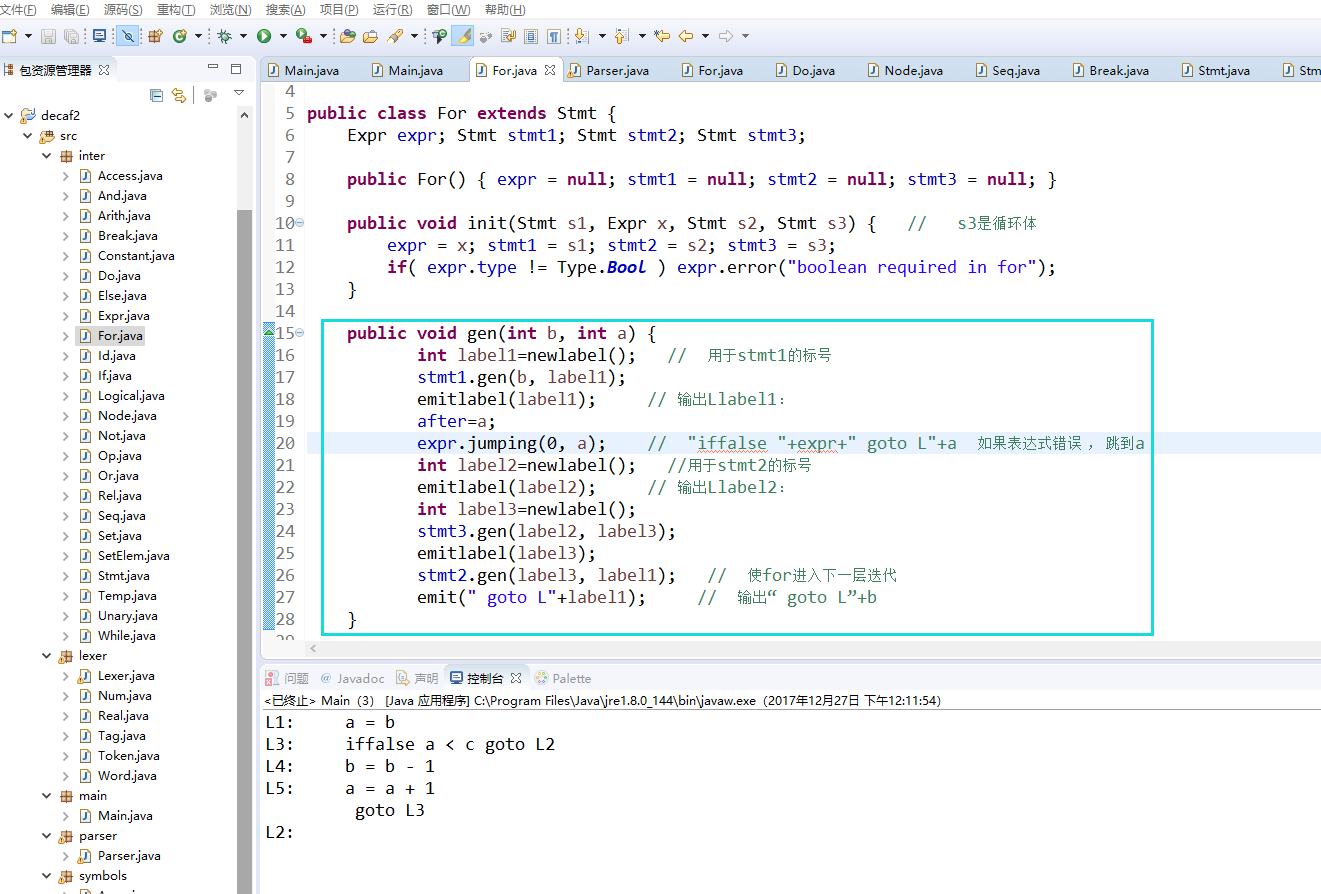


3.4 运行结果

含for语句的中间代码生成：



For类的gen代码：（可以放大来看）



4、总结

通过这两次实验以及课后的学习学到了很多知识，主要是大致看懂了这段程序，对以某些简单赋值、判断作为代码的输出的作了跟踪，了解了它们是怎么在一个个方法之间传递、运行的，特别是Rel类调用父类Logical的构造方法时，父类构造方法里的check调用的不是父类的check而是子类的check，查了好久才查出来。对于for语句的三地址码的生成，gen方法里那些语句的gen里的label并不是那么严格有序，比如两个label相同也可以正确生成等等。由于这次是直接读文件，上次block里｝的错误也就没关系了。我发现举个例子比如简单的“if（a<b）”然后去分析它是怎么通过编译的，这个方法挺不错，化繁为简，容易弄懂每一步的作用进而去深入分析。