Decaf\_PA2实验报告

计31班 刘智峰 2013011427

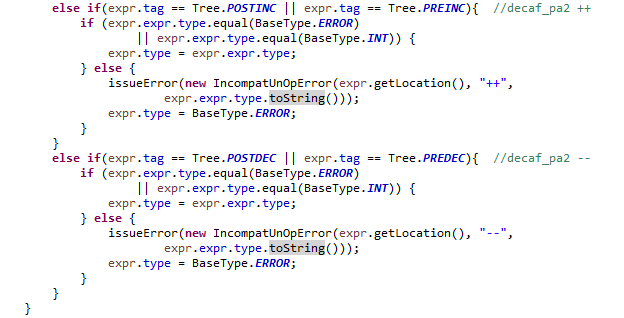
【实验内容】

本次实验将给出decaf基本框架，其中已经完成了《decaf语言规范》中所描述语言特征的词法、语法及语义分析。现在，你需要在前一阶段的词法语法分析（同时生成抽象语法树）的基础上，继续针对decaf语言新增的语言特性进行语义分析。开始时，你需要将前一阶段的工作复制到本次实验的框架中，替换掉框架中的词法语法分析（同时生成抽象语法树）的功能。建议你首先要充分理解基本框架中语义分析的代码结构以及功能，参考框架中对相似语言特征的语义处理过程，根据自己对语言的理解完成新增语言特性的语义分析。

【实验一】

1. 实验描述：实现自增、自减单操作算子的语义分析。自增、自减单操作算子形如i++,++i,i--,--i，其语义解释与C语言中一致。
2. 实现思路：通过观察TestCase中给的test\_q1\_decaf和test\_q1\_error.def以及相应的结果，可以发现，形如r=a++这样的运算，如果r和a的类型不一样，会报“incompatible operand”的错误。由于在PA1的实验中，我将++、--归入了Unary操作符，在观察TypeCheck.java文件中的visitUnary函数后，我也将++、--的实现添加到了visitUnary函数中。
3. 实现说明：

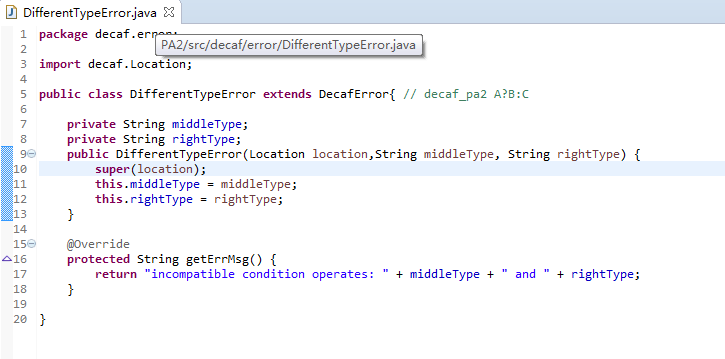
++、--没有内部内部作用域，所以不用在BuildSym.java中添加，直接在TypeCheck.java中进行检查即可。具体做法直接参照visitUnary函数即可。

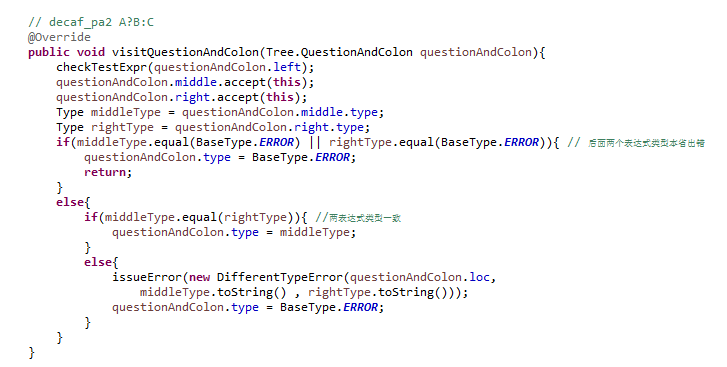


【实验二】

1. 问题描述：实现三操作数算子？：的语义分析。三操作数算子形如A？B：C。
2. 实现思路：“A？B：C”对应规则Expr -> Expr ? Expr : Expr为。正确的三元运算符需要满足“？”前的数据类型为bool，并且B与C的数据类型要一致。所以首先应对A进行类型检查。如果B与C中任何一个的数据类型已经为Error，则整个三元运算符的数据类型应赋为Error。当两者具有非Error的数据类型后再判断两者数据类型是否一致，不一致则报错，且将整个三元运算符的数据类型赋为Error。
3. 实现说明：

?:操作符没有内部内部作用域，所以不用在BuildSym.java中添加，直接在TypeCheck.java中进行检查即可。通过观察test\_q2\_error.def以及相应的结果后，我发现中间和后面这两个Expr不同时，输出的语句在已有的报错文件中未曾出现，所以我**新加了一个报错处理的java文件DifferentTypeError**来处理这类错误。





【实验三】

1. 问题描述：实现反射运算numinstances 的语义分析。反射运算numinstances形如 numinstances（A），其语义解释为：计算结果返回类A当前实例对象的个数。
2. 实现思路：在PA1中，numinstances()的实现是对照instanceof()函数来做的。所以在进行语义分析时，也可以直接参照着instanceof()的分析函数来做。通过观察test\_q3\_decaf和test\_q3\_error.def以及相应的结果可知，对于numinstances(A)所做的报错处理，就是判断A是否为一个Class。若不是，则报错。
3. 实现说明：

Numinstances()没有内部内部作用域，所以不用在BuildSym.java中添加，直接在TypeCheck.java中进行检查即可。



蓝色部分为参考的instanceof()的报错处理，下面有注释的为自己实现的numinstanceof()的出错处理。刚开始我没有设置type，老出空指针的错误，最后请教了计33的郭志芃同学。

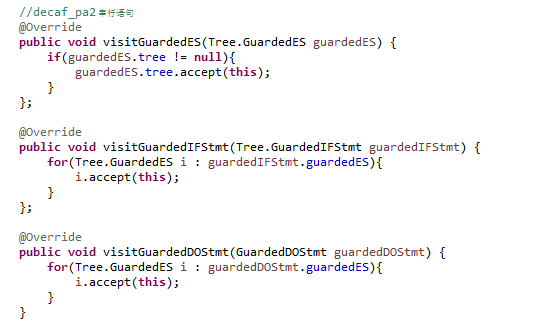
【实验四、五】

1. 问题描述：实现串行条件卫士语句和串行循环卫士语句的语义分析。语意见实验说明。

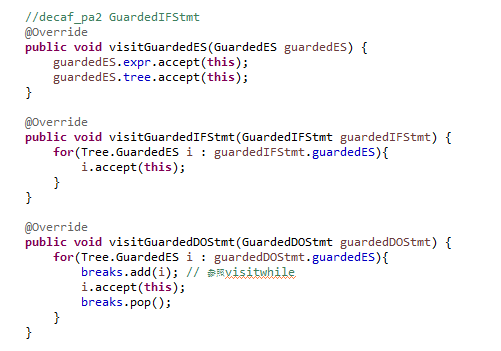
实现思路：串行条件卫士语句和串行循环卫士语句有内部域，所以需要在BuildSym.java中构造符号表。刚开始不知道怎么下手，但看到实验说明中老师提示的“参照If、While语句”后，依葫芦画瓢地构造了符号表，并在TypeCheck.java中实现了出错处理。由于在PA1的实验中，我为E：S定义了一个类型GuardedES，然后串行卫士和循环卫士都维护一个GuardedES类型的列表。所以我需要在BuildSym.java和TypeCheck.java中为GuardedES、GuardedIfStmt、GuardedDoStmt这三者建⽴立对应的语法。然后进行判断E是不是一个Bool类型的表达式、表达式的正确性等。而这些判断仿照If、While就能实现。同时，循环卫士和条件卫士有一点不同。循环卫士中允许出现break语句，而条件卫士中没有。break的实现直接照搬while的例子即可。

1. 实现说明：

BuildSym.java：

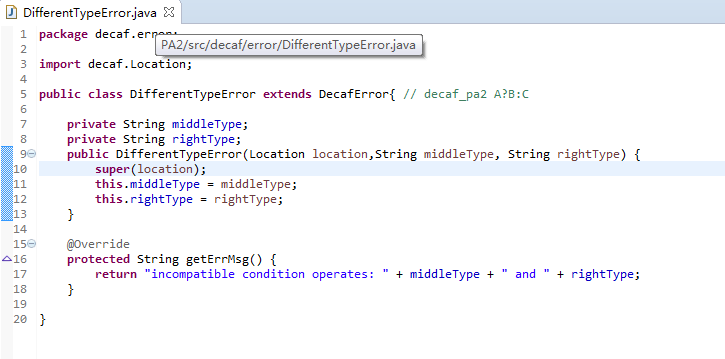


TypeCheck.java:



【增加的处理编译错误的类】

在实验二中已经提到过了。形如A?B:C的表达式，需要保证B和C为同一类型。为了在B、C类型不同时进行报错，参照了TestCase中的结果后，我自己实现了一个DifferentTypeError.java文件用于此类错误的报错。



【实验总结】

本次实验遇到的最大的bug就是导入没导入清楚，导致做第3个小实验的时候一直出现空指针的问题，调了一个多小时，最后重新导入一遍就过了……

相比于PA1，本次实验我明显对decaf实验更加熟练。通过进一步的联系，我更加深入地学习了解了decaf语言语义分析的过程和操作。

本次实验的难度不算太大，但小bug也不少。感谢老师在实验说明中提示的可以参考If、while的实现方式。如果不参考，自己摸索，要花更多的时间。在实验过程中，我还请教了计33班的郭志芃同学。感谢老师、助教和郭志芃同学给我的帮助！