# Decaf PA2 实验报告

**计 42 王栋 2014011290**

1. 用阶段一中 Lexer.l、Parser.y 文件覆盖本次实验的文件，将实验一中 Tree.java 中添加的内容复制过来。

2. ①continue 仿照 break 在 error 中添加 ContinuerOutOfLoopError；TypeCheck 中新建 continues 的栈并仿照 break 的栈来进行维护，并重写 visitContinue 的函数。

②switch-case：仿照 checkTestExpr 构造 checkSwitchVar 和 checkCaseLabel 分别用于检查 Switch 和 Case 后 Expr 的类型，error 中添加 BadSwitchVarError 和 BadCaseLabelError，Case 和 Default 需要做好 break 的入栈和出栈，之后做好各部分 accept 来重写 visit 即可。需要注意的是在 BuildSym 中同样需要重写 visit 函数，如果没有写会报空指针的错误。

③repeat-until：仿照 while 在 TypeCheck 中重写 visitRepeat，将 checkTestExpr 调整至末尾，在 BuildSym 中重写 visitRepeat。

④pclone ： 在二元对象运算表达式中添加 pclone，若左右两个对象之一为 Error，则 pclone 直接返回 Error，若两个对象皆为 Class 类型，通过 ClassType 的函数 compatible 和 getParentType 来找到两者的最小公共父类作为返回类型，否则按照二元运算符的类型错误进行处理。需要注意的是在 BaseParser 的 opStr 中添加 PCLONE，否则在类型错误时会无法识别”<<”。

⑤”?:”：三元运算符可以仿照二元运算符来构造。在 error 中添加 IncompatTerOpError。三元运算符首先需要用 checkTestExpr 检查布尔表达式，之后若后两个表达式类型相同则返回类型为表达式类型，否则返回类型为 Error，在后两个表达式没有错误且类型不等的情况下才会报 IncompatTerOpError。

3. 实验中遇到的问题：①开始时用实验一的 Tree.java 的覆盖此次实验的 Tree.java 会出现错误，仔细查看发现此次 Tree.java 中新添加了一种属性 type 和函数 setType，之后将实验一中新添加的内容复制至此次实验的 Tree.java，发现在已定义的常量（尤其是常量 case）后添加常量会报错，在常量定义末尾添加则没有这个错误。②开始时并没有注意需要在 BuildSym 中重写 visit 函数，故在 Switch-case 这一部分总是报空指针的错误。③”<<”的构造过程中没有在 BaseParser 的 opStr 中添加 PCLONE，故错误识别中会输出 unknow。④构造”<<”和三元运算符时，返回类型需要注意，需要根据 output 和 result 的比较来判断并逐渐调整。