PA1-B Report

大致思路

ABSTRACT

这个比较简单，做法和 PA1-A 一样。

VARTYPE

这个也比较简单，做法和 PA1-A 一样。

FUNCTYPE

这个就比较麻烦了，在 TYPE 和 NEW 两个地方参照 ARRAY 的处理方法写就好了。

CALL

这个也比较简单，做法和 PA1-A 一样。

LAMBDA

这个也比较简单，做法和 PA1-A 类似。

遇到的挑战

天天打错变量名调几个小时。

Eclipse uninstall

IDEA install

要求回答的问题

Q1. 本阶段框架是如何解决空悬 else (dangling-else) 问题的?

手动设置了优先级，当两个产生式的 PS 集合冲突时，会优先选择带有 ELSE Stmt 的产生式，这样会把每个 ELSE 与最近的 IF 匹配。

Q2. 使用 LL(1) 文法如何描述二元运算符的优先级与结合性？请结合框架中的文法，举例说明。

框架中的文法如下：

```
Op1             :   OR
                ;
Op2             :   AND
                ;
Expr1           :   Expr2 ExprT1
                ;
ExprT1          :   Op1 Expr2 ExprT1
```

```
                |   /* empty */
                ;
Expr2           :   Expr3 ExprT2
                ;
ExprT2          :   Op2 Expr3 ExprT2
                |   /* empty */
                ;
```

优先级：在解析 `a OR b AND c` 的时候，会先把 `a` 解析成 Expr2，再把 `b AND c` 解析成 Expr2，再把整个输入解析成 Expr1。

在解析 `a AND b OR c` 的时候，会先把 `a AND b` 解析成 Expr2，再把 `c` 解析成 Expr2，再把整个输入解析成 Expr1。

结合性：实际上的文法并不是这样子的，框架中的文法是消完左递归之后得到的文法。框架中结合性的实现方式是通过实现 `buildBinaryExpr` 得到的。本来的文法应该是

```
Expr1   :   Expr1 Op1 Expr2
        |   Expr2
        ;
```

容易看出这个产生式有左递归，并且是左结合的。

这个运算符是右结合的：

```
Expr1   :   Expr2 Op1 Expr1
        |   Expr2
        ;
```

Q3. 无论何种错误恢复方法，都无法完全避免误报的问题。 请举出一个具体的 Decaf 程序（显然它要有语法错误），用你实现的错误恢复算法进行语法分析时会带来误报。 并说明该算法为什么无法避免这种误报。

```
class Main
{
    static void main()
    {
        int a = 5);
    }
}
```

对于上面这个 Decaf 程序，我实现的错误恢复算法报了两个错误：

```
*** Error at (5,18): syntax error
*** Error at (7,1): syntax error
```

但是实际上只有 `(5,18)` 处多了一个右括号。把这个右括号忽略即可正确地完成分析。

Parser 的处理过程为：

在 `(5,18)` 处多了一个 `(` ，因为 `)` 属于 Block 的 Follow 集合，于是就会一直往外跳到 FieldList 处（跳过第四行的 `{`，但是 `)` 不属于语法树中 FieldList 的 Begin 集合以及其祖先的 Follow 集合，就又会报一个错（和上个错在同一处），并被忽略。接着第六行的 `}` 就会和第二行的 `{` 匹配。最后第七行的 `}` 就无法匹配，又会报一个错。

`)` 属于 Block 的 Follow 集合的原因是 可能会出现 `(fun(){})` 这样的情况，但是这在我们的代码中没有出现。

这个算法无法避免这种误报的原因是：Follow 集合中的符号实际上并不一定能接在该非终结符后，所以会在可恢复分析的时候终止分析，导致误报。