PA1-B实验报告

计55 徐韧喆 2015011350

1、实验要求

在本阶段，实验要求手工实现自顶向下的语法分析，并支持一定程度的错误恢复。

2、实验实现

本次实验分为两部分。

增加错误恢复功能

仿照文档提供的方法，改写parse函数如下：

```
读取Begin(A)和Follow(A)
计算End(A)
if symbol在Begin(A)中 then
    继续分析
else
    while lookAhead不在Begin(A)和End(A)中，且lookAhead未到结束符 do
        lookAhead取下一个元素
    end while
    if lookAhead在Begin(A)中 then
        继续分析
    else
        返回null
    end if
end if
```

并且在分析过程中，如果一个成分的某个子部分分析失败，则整体分析失败。即循环判定 `params[i]` 是否为空，如果有一个为空则返回空。

增加新特性对应的LL(1) 文法

将 `lexer.java` 拷入，对 `Tree.java` 和 `Parser.spec` 仿照上次代码进行更改，整复数、super、对象复制的做法与上次基本一致，故在这里不再做介绍，在此详细介绍一下另外两部分的写法：

Case表达式

为了消除左递归，做法如下：

```
Expr -> CASE '(' Expr ')' '{' CaseList DefaultExpr '}'
CaseList -> ACaseExpr CaseList
ACaseExpr -> Constant ':' Expr ';'
DefaultExpr -> DEFAULT ':' Expr ';'
```

主要改动在 `CaseList` 上，把右侧式子两部分颠倒一下即可。

循环卫士

为了消除左递归，做法如下：

```
DoStmt -> DO DoSubStmt DoBranchList OD
DoSubStmt -> Expr ':' Stmt
DoBranchList -> DoBranch DoBranchList
DoBranch -> DOOD Expr ':' Stmt
```

其中 `DO` 指 `do` ， `OD` 指 `od` ， `DOOD` 指 `|||` 。主要改动在 `Dostmt` 、 `DoBranchList` 和 `DoBranchList` 上。

3、附加题

问题1

添加优先级，在非严格模式下，如果多个产生式的PS集包含一个token，那么就自动对这些产生式设置优先级，最终在解析的时候，选择优先级最高的解析即可。

举例如下：

```
if (1)
    if (1)
        n = 2;
    else
        n = 3;
```

对于如上 `decaf` 代码，实际解析结果如下：

```
if
    intconst 1
    if
        intconst 1
        assign
            varref n
            intconst 2
    else
        assign
            varref n
            intconst 3
```

因为在冲突时优先解析 `else` 而不是空，所以在以上代码会优先对应内侧的 `if` ，因此就会解析出上述结果。

问题2

例子如下：

```
class Main {
    static void main() {
        return case (n) {
            1: 1;
            2: 2;;
            default: 3;
        };
    }
}
```

得到报错信息：

```
*** Error at (5,18): syntax error
*** Error at (6,13): syntax error
*** Error at (7,10): syntax error
*** Error at (9,1): syntax error
```

理由如下：因为 `;` 在 `Follow(return)` 中，所以传入到 `CaseList` 时， `;` 在 `End()` 集合却不在 `Begin()` 集合中，根据算法，它放弃 `CaseList` 的解析，来到上层解析，同时 `;` 也不在 `Case` 的 `Begin()` 集合中，所以再次回到上级，从而后续操作在 `return` 中解析，所以多次出错。