Decaf PA1B 实验报告

计53 王润基 2015011279

实验内容

用自顶向下方法实现语法分析。在新的框架下，补充实现以下内容：

1. 增加错误恢复功能
2. 重新实现新的语法特性

实验过程

错误恢复

根据说明，需要在查询预测集合之后，判断是否返回null。若是，则不断向后扫描符号，直到其位于Begin/End集合中。

```
Pair<Integer, List<Integer>> result = query(symbol, lookahead);
// >>>>>>>>>> 新加入内容
Set<Integer> endSet = new HashSet<>();
endSet.addAll(follow);
endSet.addAll(followSet(symbol));

if(result == null)
{
    error();
    while(true) {
      // 遇到的坑点：不能把 lookahead = lex(); 写在这里
        result = query(symbol, lookahead);
        if(result != null)
            break;
        if(endSet.contains(lookahead))
            return null;
        lookahead = lex();
    }
}
// <<<<<<<<<<
```

由于匹配失败时返回null，为了避免NullPointerException，需要忽略用户处理时出现的异常：

```
try {
  act(actionId, params); // do user-defined action
} catch (Exception ignored) {}
```

新语法特性

把PA1A中parser.y新的部分添加进来。实践上可以diff一下(A)parser.y和(B)parser.spec。

需要修改的是Do和Case的文法，要符合LL(1)规范。具体地：将左递归改为右递归，对应下面的自定义函数也要修改（忘了改这里引发了若干Bug，没留意List元素的插入位置又引发了Bug）

回答问题

1. 处理方法是优先匹配其中一个产生式（本例中是 else stmt）。

   因此对如下语句：

   ```
   if(true) if(false) a = 1; else a = 2;
   ```

   当分析到内部的if时，会优先匹配后面的else，产生下面的解析结果。

   ```
   if
       boolconst true
       if
           boolconst false
           assign
               varref a
               intconst 1
       else
           assign
               varref a
               intconst 2
   ```

2. 例如

   ```
   class Main {
       static void main() {
           int a;
           if;(true) {
       //     ^        ^ 报错
                 a = 1;
           }
       }
   }
   ```

   由于当前策略是：遇到不能匹配的token跳过文法符号但不消耗它。因此若意外提前出现了如`;`之类的终结符号，则正在解析的语句块会直接终结，出现在后面的正常内容则被误判为新的错误。