后缀表达式*Postfix*

> 学号：21307347
>
> 姓名：陈欣宇

步骤一 静态成员与非静态成员

静态成员与非静态成员变量的区别：静态变量由所有该类的对象共享，仅在类初次加载时初始化；而非静态变量在每个类对象创建时都会初始化，且各个类对象的非静态变量互不影响。静态成员和非静态成员使用需要看具体情况，当变量需要初始化加载或经常调用时建议加上`static`,其他情况则使用非静态成员。

类`Parser`中`int lookahead`定义为静态(**static**)，将`lookahead`定义为非静态进行测试，测试结果与静态情况一致，对正确性没有影响。因为程序中Parser类只有一个实例，使用静态成员还是非静态成员的结果是一样的。这里`lookahead`用作数据流的读取变量，需要保持数据的一致性，因此使用static更合适。

步骤二 消除程序中的尾递归

将`rest()`改为不带尾递归的等价程序（递归→循环）,如下编写`rest_cycle()`等价函数

```
void rest_cycle() throws IOException{
        while(lookahead == '+' || lookahead == '-'){
            if(lookahead == '+'){
                match('+');
                term();
                System.out.write('+');
            }else {
                match('-');
                term();
                System.out.write('-');
            }
        }
    }
```

用python生成不同1数量的`1+1+...+1`的中缀表达式，见`testcases/tc-1001~1005.infix`，使用脚本`testcase-1000.bat`一次性运行。用于比较带不带尾递归的性能，不断增加表达式长度，知道数量为100000时，二者展现明显不同性能。

```
Running Testcase-1001 to Testcase-1005.
============================================
The tc-1001.infix input...
Input an infix expression and output its postfix notation:

Time taking: 0.011000 ms


End of program.
--------------------------------------------
The tc-1002.infix input...
Input an infix expression and output its postfix notation:

Time taking: 0.343100 ms


End of program.
--------------------------------------------
The tc-1003.infix input...
Input an infix expression and output its postfix notation:

Time taking: 1.204300 ms


End of program.
--------------------------------------------
```

数量	rest()	rest_cycle()
10	0.0123 ms	0.0125 ms
100	0.1372 ms	0.1320 ms
1000	1.3576 ms	1.2122 ms
10000	4.4356 ms	3.4104 ms
100000	22.3207 ms	11.0778 ms

经分析，使用尾递归和使用循环的时间复杂度均为O(n)，影响到性能的因素是递归的消耗，递归调用需要进行多次参数传递，地址保留，保存多个形参，从而影响运行效率，导致在计算数量上升时，进行更多递归，运行耗时要长于循环调用。

步骤三 为程序扩展错误处理功能

- 将除了`0~9`和`+-`以外的其他字符包括空格都作为非法字符输入，也就是词法错误
- 缺少运算符或操作数则判定为语法错误，并给出具体错误提示
- 给出正确表达式的提示，其中`o`表示该位置缺少运算符，`d`表示该位置缺少数字
- 输出所有的错误及其位置

`testcase-2000.bat`打开以下图片中用例：

```
Running Testcase 2000: a correct input from DBv2.
==========================================
The input is:
9+2- 9* +2-+3+5-9k+6-kj899+*0
------------------------------------------
Input an infix expression and output its postfix notation:
92+ (error)
recover infix: 9+2-9+2-d+3+5-9+6-8o9o9+0

Lexical error: at place 4, illegal character.
Lexical error: at place 6, illegal character.
Lexical error: at place 7, illegal character.
Syntax error: at place 11, absence of digit.
Lexical error: at place 17, illegal character.
Lexical error: at place 21, illegal character.
Lexical error: at place 22, illegal character.
Syntax error: at place 24, absence of operator.
Syntax error: at place 25, absence of operator.
Lexical error: at place 27, illegal character.

Time taking: 10.471600 ms

End of program.
==========================================
```

步骤四 为程序增加文档化注释

运行`doc.bat`生成javadoc文档，存放在doc目录下。

步骤五 程序的单元测试（可选）

Junit对程序进行单元测试，测试代码位于`test/RestTest`中

项目中不同的功能函数互相调用，难以单独测试，故直接使用测试用例，面向不同功能进行测试，即分别测试词法错误和语法错误的处理，查看输出是否符合预期。

```
Test for lexical.
input:5+93+-+98
59+ (error)
recover infix: 5+9o3+d-d+9o8

Syntax error: at place 3, absence of operator.
Syntax error: at place 5, absence of digit.
Syntax error: at place 6, absence of digit.
Syntax error: at place 8, absence of operator.

Time taking: 4.918101 ms

Test for syntax.
input:5+ 3*+ 9
5 (error)
recover infix: 5+3+9

Lexical error: at place 2, illegal character.
Lexical error: at place 4, illegal character.
Lexical error: at place 6, illegal character.
```