

编译器构造实验

Lab3

后缀表达式 Postfix

姓名：郝裕玮
班级：计科 1 班
学号：18329015

目录

1 实验环境.....	3
1.1 JDK 版本.....	3
1.2 开发环境.....	3
2 实验过程&结果展示	3
2.1 Step 1 静态成员与非静态成员	3
2.2 Step 2 消除程序中的尾递归	4
2.3 Step 3 扩展错误处理功能	5
3 实验心得.....	8

1 实验环境

1.1 JDK 版本

JDK 版本: 11.0.14

1.2 开发环境

开发工具: Eclipse

2 实验过程&结果展示

2.1 Step 1 静态成员与非静态成员

在 Java 语言中, 被 `static` 所修饰的变量或者方法会储存在数据共享区, 被所有对象共享。即当属于同一个类的所有对象出现共享数据时, 就需要将存储这个共享数据的成员用 `static` 修饰。

但是由于该程序中只创建了一个 `Parser` 类对象, 因此不存在不同对象共享数据的情况。所以 `lookahead` 声明为 `static` 或非 `static` 对程序正确性无影响 (即运行结果不变)。

根据上述对 `static` 变量的作用介绍, 由于该程序只有一个数据输入流, 所以很显然 `lookahead` 的值只有唯一一份, 那么设置为 `static` 变量放入数据共享区是很有必要的。

2.2 Step 2 消除程序中的尾递归

(1) 源代码中的 rest 函数如下所示：

```
void rest() throws IOException {
    if (lookahead == '+') {
        match('+');
        term();
        System.out.write('+');
        rest();
    } else if (lookahead == '-') {
        match('-');
        term();
        System.out.write('-');
        rest();
    } else {
        // do nothing with the input
    }
}
```

(2) 将尾递归转化为循环的代码如下所示：

```
void rest() throws IOException {
    while(1){
        if (lookahead == '+') {
            match('+');
            term();
            System.out.write('+');
        } else if (lookahead == '-') {
            match('-');
            term();
            System.out.write('-');
        } else {
            return;
        }
    }
}
```

接下来我们从理论上进行性能分析：

	时间复杂度	空间复杂度
尾递归	$O(n)$	$O(n)$
循环	$O(n)$	$O(1)$

对于时间复杂度，二者相同。

对于空间复杂度，由于尾递归需要递归调用栈空间，所以有额外开销，而循环则没有这样的空间开销。

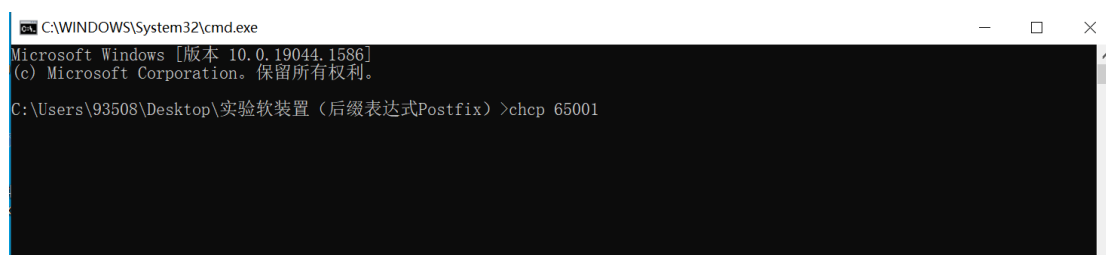
2.3 Step 3 扩展错误处理功能

具体的代码补充内容和代码思路分析可详见 Postfix.java 文件中的代码注释或参考 javadoc 文档，这里不再赘述。以下将直接展示各种测试样例的结果：

补充说明，由于代码的输出内容中有中文，直接点击脚本文件运行会产生乱码（该问题在网上查阅很久的资料后没有解决）。最终我的解决办法为，在当前路径下打开 cmd 窗口，输入 chcp 65001，之后再手动输入需要运行的脚本文件名即可运行（具体步骤可参考 testcase-001）。

（1）testcase-001

输入表达式为：9-5+2，很显然是正确的表达式。



```
C:\WINDOWS\System32\cmd.exe
Microsoft Windows [版本 10.0.19044.1586]
(c) Microsoft Corporation。保留所有权利。
C:\Users\93508\Desktop\实验软装置（后缀表达式Postfix）>chcp 65001
```

```
C:\WINDOWS\System32\cmd.exe - testcase-001.bat
Active code page: 65001

C:\Users\93508\Desktop\实验软装置（后缀表达式Postfix）>testcase-001.bat
Running Testcase 001: a correct input from DBv2.
=====
The input is:
9-5+2
=====
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
95-2+
表达式无错误!
=====
End of program.

The output should be:
95-2+
=====
Press any key to continue . . .
```

(2) testcase-002

输入表达式为：1-2+3-4+5-6+7-8+9-0，很显然是正确的表达式。

```
C:\WINDOWS\System32\cmd.exe - testcase-002.bat
Active code page: 65001

C:\Users\93508\Desktop\实验软装置（后缀表达式Postfix）>testcase-002.bat
Running Testcase 002: a correct long input.
=====
The input is:
1-2+3-4+5-6+7-8+9-0
=====
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
12-3+4-5+6-7+8-9+0-
表达式无错误!
=====
End of program.

The output should be:
12-3+4-5+6-7+8-9+0-
=====
Press any key to continue . . .
```

(3) testcase-003（两个运算量之间缺少运算符）

输入表达式为：95+2，很显然 9 和 5 之间缺少一个运算符，图见下页。

```
C:\WINDOWS\System32\cmd.exe - testcase-003.bat
Active code page: 65001
C:\Users\93508\Desktop\实验软装置 (后缀表达式Postfix) >testcase-003.bat
Running Testcase 003: missing an operator.
=====
The input is:
95+2
=====
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
92+
表达式错误总数为: 1
在位置1和位置2之间出现语法错误: 缺少运算符
```

(4) testcase-004 (运算符缺少左或右运算量)

输入表达式为 $9-5+-2$ ，很显然 5 后面的 $+$ 符号中间缺少了一个运算量。

```
C:\WINDOWS\System32\cmd.exe - testcase-004.bat
Active code page: 65001
C:\Users\93508\Desktop\实验软装置 (后缀表达式Postfix) >testcase-004.bat
Running Testcase 004: missing an operand.
=====
The input is:
9-5+-2
=====
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
95-2+
表达式错误总数为: 1
在位置4和位置5之间出现语法错误: 缺少运算量
```

(5) 表达式含空格

输入表达式为: $1 + 2 - 3 + 4$ ，程序会自动忽略空格来判断表达式的正确性。

```
C:\WINDOWS\System32\cmd.exe - run.bat
Active code page: 65001
C:\Users\93508\Desktop\实验软装置 (后缀表达式Postfix) >run.bat
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
1 + 2 - 3 + 4
12+3-4+
表达式无错误!
End of program.
Press any key to continue . . .
```

(6) 词法错误 ((1) — (5) 均为语法错误)



```
C:\WINDOWS\System32\cmd.exe - run.bat
Active code page: 65001
C:\Users\93508\Desktop\实验软装置 (后缀表达式Postfix) >run.bat
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
1+2+3p+4
12+3+4+
表达式错误总数为: 1
在位置6处出现词法错误: 出现未定义字符
Press any key to continue . . .
```

(7) 出错恢复



```
C:\WINDOWS\System32\cmd.exe - run.bat
Active code page: 65001
C:\Users\93508\Desktop\实验软装置 (后缀表达式Postfix) >run.bat
Picked up JAVA_TOOL_OPTIONS: -Dfile.encoding=UTF-8
Input an infix expression and output its postfix notation:
1+2+-3p+4
12+3+4+
表达式错误总数为: 2
在位置4和位置5之间出现语法错误: 缺少运算量
在位置7处出现词法错误: 出现未定义字符
Press any key to continue . . .
```

可见当程序发现一个错误时不是立马停下来，而是能够从跌倒的地方爬起来，继续分析下去，通过一次运行即可输出该表达式中所有错误

3 实验心得

本次实验通过代码实现使得我对语法分析的步骤有了更深一步的理解，同时也通过代码实现错误恢复这一概念让我对这次实验的代码内部逻辑掌握的更加深刻。