浙江大学实验报告

课程名称：JAVA 应用技术　　指导老师：　翁恺　　姓名：　吴雅芳

实验名称：　Calcultor　　实验类型：　编程实现　　学号：　3120104690

一、实验目的和要求

目的:了解 java 基本语法。熟悉表达式解析的方法。熟悉栈的应用和 java 程序的编写。

要求：

Write a program that reads an expression as input and print out the result. Only integers and operators below are allowed in the expression:

+ - * / % ( )

Your program should be able to deal with multiple cases of input.

二、实验内容和原理（必填）

将中序表达式转变成后序表达式(逆波兰表达式)，然后利用栈来实现表达式的计算。

三、主要仪器设备（必填）

Eclipse

四、操作方法和实验步骤

在 Eclipse 中新建 java project,键入代码，编译通过得到 class 文件，再进行解释执行。

首先将中序表达式转化成逆波兰表达式：

1.判断是否为数字，是数字则压入后序表达式栈中；

2.判断是否为符号：

如果为符号，与操作符栈顶的操作符比较，优先级大于栈顶优先符时，压入栈中；否则，将操作符栈中的操作符弹出到后序表达式栈中，知道当前操作符优先级大于操作符栈顶操作符时为止。如果是‘（’直接压入操作符栈，如果是‘）’，则取出栈中元素直到将最近的’(’取出为止。

3.重复 1,2 步，直到中序表达式被检验完；将操作符栈中的元素弹出压入后序表达式栈。

4.将后序表达式栈中元素反转。

接下来求解后序表达式：

1.从后续表达式反转式栈(记做 Stack A)中取出元素，如果是数字，则压入栈 B 中

2.如果是操作数，则取出栈 B 中的两个元素做相应运算，并将结果压入栈 B

3.重复 1,2 步，直到栈 A 中为空；此时栈 B 中只剩一个元素，即为表达式结果。

五、实验数据记录和处理

Sample Input：

3 + 5 * 6

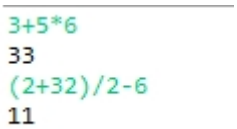
(2+32)/2-6

Sample Output：

33

11

运行结果截图：



六、实验结果与分析（必填）

运行正确。

七、讨论、心得

1.涉及多项 Stack 操作：

新建 Stack:

```
private Stack<String> postfixStack=new Stack<String>();//后缀表达式栈
private Stack<Character> OPStack=new Stack<Character>();//运算符栈
private Stack<String> reverseStack=new Stack<String>();//反转后缀
private Stack<String> operatorStack=new Stack<String>();//操作数后缀
```

Pop 操作：取出栈顶元素

Empty 操作：判断 Stack 是否为空，为空返回 true，不为空返回 false

Peek 操作：读取栈顶元素（不把它从中取出）

Push 操作：把元素压入栈中

```
while(!OPStack.empty()&&checkPriority(exp[i])<=checkPriority(OPStack.peek()))
{
    postfixStack.push(String.valueOf(OPStack.pop()));
}
OPStack.push(exp[i]);
//System.out.println(i+"="+exp[i]);
```

2.涉及多种数据类型的转换：

String->char[]:

String.toCharArray()

```
char[] exp=expression.toCharArray();
```

char[]->String:

String s;

char[] char;

s=String.ValueOf(char)

```
while(!OPStack.empty())
{
    postfixStack.push(String.valueOf(OPStack.pop()));
}
```

String->int:

Int i=12345;

String s="";

第一种方法：s=String.ValueOf(i);

第一种方法：s=i+"";

```
return  String.valueOf(result);
```

int->Sring:

String s=”12345”;

int i=0;

i=Integer.parseInt(s);

```
switch(x)
{
case'+':
    result=Integer.parseInt(first.trim())+Integer.parseInt(second.trim());
    break;
case'-':
    result=Integer.parseInt(first.trim())-Integer.parseInt(second.trim());
    break;
case'*':
    result=Integer.parseInt(first.trim())*Integer.parseInt(second.trim());
    break;
case'/':
    result=Integer.parseInt(first.trim())/Integer.parseInt(second.trim());
    break;
case'%':
    result=Integer.parseInt(first.trim())%Integer.parseInt(second.trim());
    break;
default:
    break;
}
```

3.string.length 在未知字符串长度的表示时十分有用。

装

订

线