天津大学

程序设计实践 3

学　　院　智能与计算学部

班　　级　软件工程 5 班

年　　级　2017 级

姓　　名　张　洛

学　　号　3017232148

一.实验目的

能够掌握线性结构的概念、基本操作，选择合理的结构存储数据，选择适当的工具编写程序，掌握多项式运算算法，编写程序实现题目要求。

二.实验内容

编写一个计算器工具，能够实现 windows 附件中科学型计算器的功能。

三.实验步骤

1.编程选择

经过考虑后决定选用 JAVA 作为科学计算器的编程语言。

2.前期准备

通过网上资料查询及同学交流，初步了解科学计算器的实现方法。

3.功能实现

通过与 windows 科学计算器进行比对，本程序提供以下功能的实现：

1.角度弧度制的转换

2.双曲正切与反双曲正切和正切与反正切的相互切换，通过 HYP 按键实现

3.科学计数法转换，通过 F_E 按键实现

4.三角函数 tan、sin、cos 与相应的 arctan、arcsin、arccos

5.双曲函数 tanh、sinh、cosh 与相应的 arctanh、arcsinh、arccosh

6.Log、ln 对数运算

7.Mod 取余运算

8. √平方根运算

9.n!阶乘运算（不支持广义阶乘）

10.^幂运算

11.开 n 次方根计算，这里用 yroot 表示

12.快捷入口 10 的 x 次方，e 的 x 次方，1/x 倒数，x 的 2 次方，x 的 3 次方

13.Exp 科学计数法转换

14.|x|取绝对值

15.基本的清空 C，清空当前 CE，←退格键，+-×÷加减乘除，=等于，小数运算，pi 和 e 的运算，双括号，±正负快捷转换

16.存储功能：MS、M-、M+、MR、MC

17.标准型计算器与科学型计算器的转换。

4.代码部分

ScientificCalculator.java：

```
package scientificCalculator;

import java.util.Scanner;
import java.util.Stack;
import java.util.Vector;
import java.math.*;
import java.util.regex.*;



public class ScientificCalculator {
```

```
    public static String exchange(String temp) {
        temp = temp.replace("arctanh", "b");
        temp = temp.replace("arccosh", "n");
        temp = temp.replace("arcsinh", "m");
        temp = temp.replace("tanh", "g");
        temp = temp.replace("cosh", "h");
        temp = temp.replace("sinh", "j");
        temp = temp.replace("arctan", "q");
        temp = temp.replace("arccos", "w");
        temp = temp.replace("arcsin", "r");
        temp = temp.replace("Mod", "M");
        //temp = temp.replace("abs", "a");
        temp = temp.replace("π", Math.PI + "");
        temp = temp.replace("e", Math.E + "");

        String regex = "[0-9]{1,}!"; //正则表达式
        Pattern pattern = Pattern.compile(regex);
        Matcher m = pattern.matcher(temp);
        Vector<String> matchRegexList = new Vector<String>();
        while(m.find()){
            matchRegexList.add(m.group());
        }

        for(int i = 0;i<matchRegexList.size();i++) {
            temp = temp.replace(matchRegexList.get(i), "f(" +
matchRegexList.get(i).substring(0, matchRegexList.get(i).length() - 1) +
")");
        }


        return temp;
    }


    public static void main(String[] args) {
        String expression;
        System.out.print("Enter an expression: ");
        Scanner input = new Scanner(System.in);
        expression = input.nextLine();
        System.out.println(expression + " = " +
Expression.evaluateExpression(expression));
    }
```

```
}
```

Split.java:

```
package scientificCalculator;

import java.util.Vector;

public class Split {
    public static Vector<String> split(String expression) {
        Vector<String> v = new Vector<>();
        StringBuffer numberString = new StringBuffer();

        for (int i = 0; i < expression.length(); i++) {
            if (Character.isDigit(expression.charAt(i)) ||
expression.charAt(i) == '.' || expression.charAt(i) == 'E')
                numberString.append(expression.charAt(i));
            else if((expression.charAt(i) +
"").matches("t{0,1}c{0,1}s{0,1}L{0,1}a{0,1}")) {//tan cos sin Log
                v.add(expression.charAt(i) + "");
                i += 2;
            }
            else if((expression.charAt(i) + "").matches("l{0,1}")) {//ln
                v.add(expression.charAt(i) + "");
                i += 1;
            }
            else if((expression.charAt(i) +
"").matches("q{0,1}w{0,1}r{0,1}f{0,1}√{0,1}b{0,1}n{0,1}m{0,1}g{0,1}h{0,1}j{
0,1}")) {//arctan arcsin arccos n! √
                v.add(expression.charAt(i) + "");
                i += 0;
            }
            else {
                if (numberString.length() > 0) {
                    v.add(numberString.toString());
                    numberString.setLength(0);
                }

                if (!Character.isSpaceChar(expression.charAt(i))) {
                    v.add(expression.charAt(i) + "");
                }
            }
        }
```

```
        if (numberString.length() > 0)
            v.add(numberString.toString());

        return v;
    }


}
```

Operator.java:

```
package scientificCalculator;

import java.util.Stack;

public class Operator {
    public static boolean DEG = true;

    public static double fact(long op) {
        if(op == 1) {
            return 1;
        }
        else if(op == 0) {
            return 0;
        }
        else {
            return op * fact(op - 1);
        }
    }

    public static void processAnOperator(Stack<Double> operandStack,
Stack<Character> operatorStack) {
        char op = operatorStack.pop();
        double op1 = 0, op2 = 0;
        if(op == '+' || op == '-' || op == '*' || op == '/' || op == 'M' ||
op == '^' || op == 'y' || op == '%') {
            op1 = operandStack.pop();
            op2 = operandStack.pop();
        }
        else if((op +
"").matches("t{0,1}c{0,1}s{0,1}L{0,1}l{0,1}q{0,1}w{0,1}r{0,1}f{0,1}√{0,1}b{
0,1}n{0,1}m{0,1}g{0,1}h{0,1}j{0,1}a{0,1}")) {
```

```
            op1 = operandStack.pop();
        }


        if (op == '+')
            operandStack.push(op2 + op1);
        else if (op == '-')
            operandStack.push(op2 - op1);
        else if (op == '/')
            operandStack.push(op2 / op1);
        else if (op == '*')
            operandStack.push(op2 * op1);
        else if(op == 'M' || op == '%')
            operandStack.push(op2 % op1);
        else if(op == '^')
            operandStack.push(Math.pow(op2, op1));
        else if(op == 'y')
            operandStack.push(Math.pow(op2, 1 / op1));
        else if(op == 't')
            operandStack.push(!DEG ? Math.tan(op1) : Math.tan(Math.PI * op1
/ 180));
        else if(op == 's')
            operandStack.push(!DEG ? Math.sin(op1) : Math.sin(Math.PI * op1
/ 180));
        else if(op == 'c')
            operandStack.push(!DEG ? Math.cos(op1) : Math.cos(Math.PI * op1
/ 180));
        else if(op == 'L')
            operandStack.push(Math.log10(op1));
        else if(op == 'l')
            operandStack.push(Math.log(op1));
        else if(op == 'q')
            operandStack.push(!DEG ? Math.atan(op1) : Math.atan(op1) * 180 /
Math.PI);
        else if(op == 'w')
            operandStack.push(!DEG ? Math.acos(op1) : Math.acos(op1) * 180 /
Math.PI);
        else if(op == 'r')
            operandStack.push(!DEG ? Math.asin(op1) : Math.asin(op1) * 180 /
Math.PI);
        else if(op == 'f')
            operandStack.push(fact(Math.round(op1)));
        else if(op == '√')
            operandStack.push(Math.sqrt(op1));
```

```
        else if(op == 'b')
            operandStack.push(0.5 * Math.log((1 + op1) / (1 - op1)));
        else if(op == 'n')
            operandStack.push(Math.log(op1 + Math.sqrt(op1 * op1 - 1)));
        else if(op == 'm')
            operandStack.push(Math.log(op1 + Math.sqrt(op1 * op1 + 1)));
        else if(op == 'g')
            operandStack.push(Math.tanh(op1));
        else if(op == 'h')
            operandStack.push(Math.cosh(op1));
        else if(op == 'j')
            operandStack.push(Math.sinh(op1));
        else if(op == 'a')
            operandStack.push(Math.abs(op1));
        else {

        }
    }

}
```

Expression.java:

```
package scientificCalculator;

import java.util.Stack;
import java.util.Vector;

public class Expression {
    public static double evaluateExpression(String expression) {
        expression = ScientificCalculator.exchange(expression);
        Stack<Double> operandStack = new Stack<>();
        Stack<Character> operatorStack = new Stack<>();
        Vector<String> tokens = Split.split(expression);

        for (int i = 0; i < tokens.size(); i++) {
            if (tokens.get(i).charAt(0) == '+' || tokens.get(i).charAt(0) ==
'-') {
                while (!operatorStack.empty() && (operatorStack.peek() ==
'+' || operatorStack.peek() == '-' || operatorStack.peek() == '/' ||
operatorStack.peek() == '*'
                        || operatorStack.peek() == 'c' ||
operatorStack.peek() == 's' || operatorStack.peek() == 't' ||
```

```
operatorStack.peek() == 'L' || operatorStack.peek() == '%'
                        || operatorStack.peek() == 'l' ||
operatorStack.peek() == 'q' || operatorStack.peek() == 'w' ||
operatorStack.peek() == 'r' || operatorStack.peek() == 'M'
                        || operatorStack.peek() == 'f' ||
operatorStack.peek() == 'v' || operatorStack.peek() == 'y' ||
(operatorStack.peek() +
"").matches("b{0,1}n{0,1}m{0,1}g{0,1}h{0,1}j{0,1}a{0,1}")))
                    Operator.processAnOperator(operandStack, operatorStack);

                operatorStack.push(tokens.get(i).charAt(0));
            }
            else if (tokens.get(i).charAt(0) == '*' ||
tokens.get(i).charAt(0) == '/' || tokens.get(i).charAt(0) == 'M' ||
tokens.get(i).charAt(0) == '%') {
                while (!operatorStack.empty() && (operatorStack.peek() ==
'*' || operatorStack.peek() == '/' || operatorStack.peek() == 'c' ||
operatorStack.peek() == 's' || operatorStack.peek() == '%'
                        || operatorStack.peek() == 't' ||
operatorStack.peek() == 'L' || operatorStack.peek() == 'l' ||
operatorStack.peek() == 'a' || operatorStack.peek() == 'M'
                        || operatorStack.peek() == 'q' ||
operatorStack.peek() == 'w' || operatorStack.peek() == 'r' ||
operatorStack.peek() == 'f' || operatorStack.peek() == 'v'
                        || operatorStack.peek() == 'y' ||
(operatorStack.peek() +
"").matches("b{0,1}n{0,1}m{0,1}g{0,1}h{0,1}j{0,1}a{0,1}")))
                    Operator.processAnOperator(operandStack, operatorStack);

                operatorStack.push(tokens.get(i).charAt(0));
            }
            else if(tokens.get(i).charAt(0) == '^') {
                while(!operatorStack.empty() && (operatorStack.peek() == '^'
|| operatorStack.peek() == 'y'))
                    Operator.processAnOperator(operandStack, operatorStack);

                operatorStack.push(tokens.get(i).charAt(0));
            }
            else if(tokens.get(i).charAt(0) == 'y') {
                while(!operatorStack.empty() && operatorStack.peek() == 'y')
                    Operator.processAnOperator(operandStack, operatorStack);

                operatorStack.push(tokens.get(i).charAt(0));
            }
```

```
            else
if(tokens.get(i).matches("t{0,1}c{0,1}s{0,1}L{0,1}l{0,1}q{0,1}w{0,1}r{0,1}f
{0,1}v{0,1}b{0,1}n{0,1}m{0,1}g{0,1}h{0,1}j{0,1}a{0,1}")) {
                operatorStack.push(tokens.get(i).charAt(0));
            }
            else if (tokens.get(i).charAt(0) == '(')
                operatorStack.push(tokens.get(i).charAt(0));
            else if (tokens.get(i).charAt(0) == ')') {
                while (operatorStack.peek() != '(')
                    Operator.processAnOperator(operandStack, operatorStack);

                operatorStack.pop();
            }
            else {
                operandStack.push(Double.valueOf(tokens.get(i)));
            }
        }

        while (!operatorStack.empty())
            Operator.processAnOperator(operandStack, operatorStack);

        return operandStack.pop();

    }


}
```

核心文件 Calculator.java 中首先用 split 函数将用户输入的字符串按操作数和操作符分割开来并存入一个 vector 中。

在 evaluateExpression 中采用双栈法存储操作数和操作符，对 split 函数分解的 vector 进行逐个扫描，并根据操作数或操作符及其优先级入栈。同时，当操作符栈中已有操作符时，在 processAnOperator 中弹栈处理操作符和操作数并将计算出的结果压回操作数栈。

在最开始实现最基本的加减乘除功能时，调用 evaluateExpression 时，逐个扫描该 vector，如遇到同级的+或-操作符，那么检查操作符栈是否为空，如果为空则直接将该操作符压栈。若不为空，再检查该操作符栈顶指针所指向的内容是否为+或-或与+-同级或比其优先级低的操作符，若有，则调用 processAnOperator 函数进行计算。如遇到*或/操作符，同理检查栈是否空，空则压栈，不空则检查栈顶指针指向内容是否为与其同级或优先级低的操作符，若有，调用 processAnOperator 计算。如遇到左括号，直接压栈。如遇右括号，当栈顶不为左括号时，调用 processAnOperator 计算，然后弹栈。如遇操作数，直接压入操作数栈。当对 vector 扫描完成后，如果操作符栈中还有未处理完的操作符，则调 processAnOperator 直到操作符栈空。

在 processAnOperator 中，先对操作符栈弹栈进行检查，若该操作符为双目运算符，则操作数栈弹两个数。若为单目运算符，则弹一个。然后对不同的操作符进行处理并将结果压回操作数栈。

- **实验分析**

  本程序按照题目完成了所有的要求，实现了 Windows 10 附件中科学型计算的功能。不足之处：本程序由于时间、水平有限，未能实现图形界面。