牛客网-华为机试练习题 50

题目描述

请实现如下接口

/* 功能：四则运算

* 输入：strExpression：字符串格式的算术表达式，如: "3+2{1+2[-4/(8-6)+7]}"

* 返回：算术表达式的计算结果

*/

public static int calculate(String strExpression)

{

/* 请实现*/

return 0;

}

约束：

1. pucExpression字符串中的有效字符包括[‘0’-‘9’],‘+’,‘-’, ‘*’,‘/’ ,‘(’， ‘)’,‘[’, ‘]’,‘{’ ,‘}’。
2. pucExpression算术表达式的有效性由调用者保证;

输入描述:

输入一个算术表达式

输出描述:

得到计算结果

示例1

输入

```
3+2*{1+2*[-4/(8-6)+7]}
```

输出

```
25
```

解决代码：

```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.List;
import java.util.Stack;
```

```
  //四则运算带括号
public class Main {
    public static void main(String[] args) throws IOException {
        BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
        String line = "";
        while ((line = br.readLine()) != null) {
            Stack<Character> stack = new Stack<Character>();
            List<Object> list = new ArrayList<Object>();

            for (int i = 0; i < line.length(); i++) {
                String T = "";
                boolean isN = false; // 负号
                if (i == 0 && line.charAt(i) == '-') {
                    isN = true;
                    ++i;
                } else if (line.charAt(i) == '-' && (line.charAt(i - 1) == '-' ||
line.charAt(i - 1) == '+'
                        || line.charAt(i - 1) == '*' || line.charAt(i - 1) == '/' ||
line.charAt(i - 1) == '('
                        || line.charAt(i - 1) == '[' || line.charAt(i - 1) == '{')) {
                    isN = true;
                    ++i;
                }
                while (i < line.length() && line.charAt(i) >= '0' && line.charAt(i) <= '9') {
                    T = T + line.charAt(i++);
                }
                if (!T.equals("")) {
                    --i;
                    if (isN) {
                        list.add(0 - (new Integer(T)));
                    } else {
                        list.add(new Integer(T));
                    }
                } else {
                    char op = line.charAt(i);
                    if (op == '+' || op == '-' || op == '*' || op == '/') {
                        if (stack.isEmpty()) {
                            stack.push(op);
                        } else if (isUpperPro(op, (char) stack.peek())) {
                            stack.push(op);
                        } else {
                            while (!stack.isEmpty()
                                    && (stack.peek() != '(' || stack.peek() != '[' ||
stack.peek() != '{')
                                    && !isUpperPro(op, (char) stack.peek())) {
                                list.add(stack.pop());
                            }
                            stack.push(line.charAt(i));
                        }
                    } else if (op == '(' || op == '[' || op == '{') {
                        stack.push(op);
                    } else if (line.charAt(i) == ')') {
                        while (stack.peek() != '(') {
                            list.add(stack.pop());
                        }
                        stack.pop();
                    } else if (line.charAt(i) == ']') {
                        while (stack.peek() != '[') {
                            list.add(stack.pop());
                        }
```

```
                    stack.pop();
                } else if (line.charAt(i) == '}') {
                    while (stack.peek() != '{') {
                        list.add(stack.pop());
                    }
                    stack.pop();
                }
            }
        }
        while (!stack.isEmpty()) {
            list.add(stack.pop());
        }
        Stack<Integer> Pstack = new Stack<Integer>();
        Iterator<Object> it = list.iterator();
        while (it.hasNext()) {
            Object temp = it.next();
            if (temp instanceof Integer) {
                Pstack.push((Integer) temp);
            } else if (temp instanceof Character) {
                int N2 = Pstack.pop();
                int N1 = Pstack.pop();
                int res = getRes(N1, N2, (char) temp);
                Pstack.push(res);
            }
        }
        System.out.println(Pstack.pop());
    }
}

public static int getRes(int n1, int n2, char temp) {
    if (temp == '-') {
        return n1 - n2;
    }
    if (temp == '+') {
        return n1 + n2;
    }
    if (temp == '*') {
        return n1 * n2;
    }
    if (temp == '/') {
        return n1 / n2;
    }
    return 0;
}

public static boolean isUpperPro(char op, char peek) {
    if (peek == '(' || peek == '[' || peek == '{') {
        return true;
    }
    if ((op == '+' || op == '-') && (peek == '*' || peek == '/')) {
        return false;
    }
    if ((op == '*' || op == '/') && (peek == '+' || peek == '-')) {
        return true;
    }
    if ((op == '+' || op == '-') && (peek == '+' || peek == '-')) {
        return false;
    }
    if ((op == '*' || op == '/') && (peek == '*' || peek == '/')) {
        return false;
```

```
        }
        return false;
    }
}
```