1. 题目
2. 题干

给定一个包含正整数、加(+)、减(-)、乘(\*)、除(/)的算数表达式(括号除外)，计算其结果。

表达式仅包含非负整数，+， - ，\*，/ 四种运算符和空格  。 整数除法仅保留整数部分。

1. 示例

示例 1:

输入: "3+2\*2"

输出: 7

示例 2:

输入: " 3/2 "

输出: 1

示例 3:

输入: " 3+5 / 2 "

输出: 5

说明：

你可以假设所给定的表达式都是有效的。

请不要使用内置的库函数 eval。

1. 题解
2. 思路

给出一串数字和运算符，计算结果，很明显这需要使用到栈。题目限制了没有括号，这样题解就更加地简单，只需要关注于数字和+，-，\*，/的入栈和出栈。能够入栈的都是最终要加起来的数，因此，规则如下：

遇到整数时，可以直接入栈；

遇到空格时，跳过；

遇到+号，则说明下一个数可能要加到栈中，那么将下一个数字num要入栈；

遇到-号，则说明下一个数字可能是要删减的，因此将下一个数字num乘以-1入栈；

遇到\*号，说明前一个数需要乘以后一个数，而前一个数在栈的顶部，那么将栈的顶部弹出，乘以后一个数并且入栈即可；

遇到/号，说明前一个数需要除以后一个数，而前一个数在栈的顶部，那么将栈的顶部弹出，除以后一个数并且入栈即可；

最后，将栈中的数相加就是结果了。

注意，题目中的每个数可能是多位的，因此要把每个数构建出来。在循环中，首先判断该位置的字符是否为空格，如果是空格，则跳到下一个循环；然后记录下当前的字符，判断该字符是否是运算符，如果是运算符，则需要将索引+1，下一个字符必为空格或者数字，如果是空格则继续增加索引；如果是数字，则将该数字通过循环构建出来，然后根据之前记录的运算符符号，来相应地进行入栈出栈操作，最后将栈中的值相加即可。

1. 代码实现

Java：

class Solution {

    public int calculate(String s) {

        Stack<Integer> out = new Stack<Integer>();

        String trim = s.trim();

        int n = trim.length();

        int i = 0, ans = 0;

        while(i < n){

            if(trim.charAt(i) == ' '){

                i++;

                continue;

            }

            char temp = trim.charAt(i);

            if(temp == '+' || temp == '-' ||temp == '\*' ||temp == '/'){

                i++;

                while(trim.charAt(i) == ' '){

                    i++;

                }

            }

            int num = 0;

            while(i < n && Character.isDigit(trim.charAt(i))){

                num = num \* 10 + trim.charAt(i) - '0';

                i++;

            }

            switch(temp){

                case '-':

                    num  = -num;

                    break;

                case '\*':

                    num = out.pop() \* num;

                    break;

                case '/':

                    num  = out.pop() / num;

                    break;

                default:

                    break;

            }

            out.push(num);

        }

        while(!out.isEmpty()){

            ans += out.pop();

        }

        return ans;

    }

}