150. Evaluate Reverse Polish Notation

Description | Hints | Submissions | Discuss | Solution

Pick One

Evaluate the value of an arithmetic expression in Reverse Polish Notation.

Valid operators are `+`, `-`, `*`, `/`. Each operand may be an integer or another expression.

Note:

- Division between two integers should truncate toward zero.
- The given RPN expression is always valid. That means the expression would always evaluate to a result and there won't be any divide by zero operation.

Example 1:

```
Input: ["2", "1", "+", "3", "*"]
Output: 9
Explanation: ((2 + 1) * 3) = 9
```

Example 2:

```
Input: ["4", "13", "5", "/", "+"]
Output: 6
Explanation: (4 + (13 / 5)) = 6
```

Example 3:

```
Input: ["10", "6", "9", "3", "+", "-11", "*", "/", "*", "17", "+", "5", "+"]
Output: 22
Explanation:
  ((10 * (6 / ((9 + 3) * -11))) + 17) + 5
= ((10 * (6 / (12 * -11))) + 17) + 5
= ((10 * (6 / -132)) + 17) + 5
= ((10 * 0) + 17) + 5
= (0 + 17) + 5
= 17 + 5
= 22
```

```java
 * 这道题目考察的是逆波兰表达式的生成和运算方法。
 */
public class L150 {
      public int evalRPN(String[] tokens) {
         Stack<Integer> stack = new Stack<>(); //栈，用于遍历初始字符串数组
         int a, b;    //临时存放栈中弹出两个元素
         /*
          * 遍历初始字符串数组，若当前字符为运算符，则从栈中弹出两个元素，并用该运算符对它们
          * 进行运算，然后再将运算结果压入栈，若读到的是数字，则直接将其压入栈，不作其他操作
          */
         for(int i = 0; i < tokens.length; i++) {
             String temp = tokens[i];
             switch (temp) {
            case "+":
                a = stack.pop();
                b = stack.pop();
                stack.push(b + a);
                break;
            case "-":
                a = stack.pop();
                b = stack.pop();
                stack.push(b - a);
                break;
            case "*":
                a = stack.pop();
                b = stack.pop();
                stack.push(b * a);
                break;
            case "/":
                a = stack.pop();
                b = stack.pop();
                if(a == 0)
                    return -1;
                stack.push(b / a);
                break;
            default:
                stack.push(Integer.parseInt(temp));
            }
         }
         return stack.peek();
      }
}
```