Evaluate Reverse Polish Notation

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

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

Some examples:

```
["2", "1", "+", "3", "*"] -> ((2 + 1) * 3) -> 9
["4", "13", "5", "/", "+"] -> (4 + (13 / 5)) -> 6
```

Solution 1

when I test ["10","6","9","3","+","-11","","/","","17","+","5","+"], in this program, the result I got is 12, I think I am right. Because when I calculate 6/(-132) = -1, not 0, so i think the result is 12 not 22.

written by yfdyyy original link here

Solution 2

Hi everyone.
The Reverse Polish Notation is a stack of operations, thus, I decided to use java.util.Stack to solve this problem. As you can see, I add every token as an integer in the stack, unless it's an operation. In that case, I pop two elements from the stack and then save the result back to it. After all operations are done through, the remaining element in the stack will be the result.
Any comments or improvements are welcome.

Cheers.

```
import java.util.Stack;

public class Solution {
    public int evalRPN(String[] tokens) {
        int a,b;
        Stack<Integer> S = new Stack<Integer>();
        for (String s : tokens) {
            if(s.equals("+")) {
                S.add(S.pop()+S.pop());
            }
            else if(s.equals("/")) {
                b = S.pop();
                a = S.pop();
                S.add(a / b);
            }
            else if(s.equals("*")) {
                S.add(S.pop() * S.pop());
            }
            else if(s.equals("-")) {
                b = S.pop();
                a = S.pop();
                S.add(a - b);
            }
            else {
                S.add(Integer.parseInt(s));
            }
        }
        return S.pop();
    }
}
```

written by pvaldes original link here

Solution 3

```
public int evalRPN(String[] a) {
  Stack<Integer> stack = new Stack<Integer>();

  for (int i = 0; i < a.length; i++) {
    switch (a[i]) {
      case "+":
        stack.push(stack.pop() + stack.pop());
        break;

      case "-":
        stack.push(-stack.pop() + stack.pop());
        break;

      case "*":
        stack.push(stack.pop() * stack.pop());
        break;

      case "/":
        int n1 = stack.pop(), n2 = stack.pop();
        stack.push(n2 / n1);
        break;

      default:
        stack.push(Integer.parseInt(a[i]));
    }
  }

  return stack.pop();
}
```

written by jeantimex original link here