Talan · RDV 2.

KATA - Reverse Polish Notation = RPN

In reverse Polish notation the operators follow their operands; for instance, to add 3 and 4, one would write "3 4 +" rather than "3 + 4". If there are multiple operations, the operator is given immediately after its second operand; so the expression written "3 – 4 + 5" in conventional notation would be written "3 4 – 5 +" in RPN: 4 is first subtracted from 3, then 5 added to it. An advantage of RPN is that it obviates the need for parentheses that are required by infix. While "3 – 4 * 5" can also be written "3 – (4 * 5)", that means something quite different from "(3 – 4) * 5". In postfix, the former could be written "3 4 5 * -", which unambiguously means "3 (4 5 *) -" which reduces to "3 20 -"; the latter could be written "3 4 – 5 *" (or 5 3 4 – *, if keeping similar formatting), which unambiguously means "(3 4 -) 5 *".

The goal of this kata is to create a program that performs calculations using reverse Polish notation.

tmp.peek() → renvoie l'objet sans le supprimer, situé au sommet de la pile
tmp.search() → -1 si objet ∅ de pile, renvoi (int) posit° objet ds pile
tmp.empty() → booleen = reste si pile vide ou non.

"34+" → 3+4
"34-5+" → 3-4+5
"345*-" → 3-(4x5)
"534-*" = "34-5*" → (3-4)x5 = 5x(3-4)

il faut un style de pile ⇐

Dès qu'il y a 2 chiffres + 1 opé on execute le calcul qu'on remplace ds le string

345*-	534-*	34-5*
320-	5(-1)*	(-1)5*
-17	-5	-5

stack => extension de vector, une pile du type dernier entré / premier sorti

Exemple :

```
void main(){
    String[] str = new String[]{"3","4","-","5","+"};
    System.out.println(calculRPN(s));
}

int calculRPN(String[] str){
    int result = 0;
    String operateurs = "+-*/";
    Stack<String> tmp = new Stack<String>();
    for(String s: str){
        if(!operateurs.contains(s)){
            tmp.push(s);
        }else{
```

si c'est un nbr ←

(≃ cast string to int) Integer.valueOf

récupère l'objet au sommet de la pile

```
            int nb1 = tmp.pop();
            int nb2 = tmp.pop();
            int index = operateurs.indexOf(s);
            switch(index){
                case 0:
                    tmp.push(String.valueOf(nb1+nb2));
                    break;
                case 1: ...
                case 2: ...
                case 3: ...
            }
        }
    }
    result = Integer.valueOf(tmp.pop());
    return result;
}
```

ajoute l'objet au sommet de la pile et le renvoie