Polish Notation

2+((12%5)*3)

PN

2 + ((12 %o 5)x3)

+2((12 %o 5)x3)

+2(x(12%o5)3)

+2(x(%o 12 5)3)

+2 x %o 12 5 3

Se apila de derecha a izquierda

S.apila(3); S.apila(5); S.apila(12);
S.apila(%o); S.apila(*); S.apila(2);
S.apila(+);

exp	aux
2	
*	
%	
12	
5	
3	+

exp	aux
*	
%	
12	
5	2
3	+

exp	aux
%	
12	*
5	2
3	+

exp	aux
12	*
5	2
3	+

exp	aux
12	%
5	*
3	+

exp	aux
5	2
3	+

RPN (Reverse)

2+((12 %o 5)x3)

2((12%o5)*3)+

2((12%o5)3)*+

2((12 5)%3)*+

2 12 5 %o 3*+

Se apila de derecha a izquierda

S. apila(+); S. apila(*); S. apila(3); S. apila(%o);
S. apila(5); S. apila(12); S. apila(2);

exp	aux
2	
12	
5	
%	
3	
*	
+	

exp	aux
12	
5	
%	
3	
*	
+	2

exp	aux
5	
%	
3	12
*	
+	2

exp	aux
%	
3	5
*	12
+	2

exp	aux
3	2
*	
+	2

Se va desapilando en aux, cuando en aux hay dos núm seguidas y un operando en exp, se opera y se guarda en aux

Código

Pila exp ⇒ tiene la expresión en PN, ex: +2*34 ⇒ 2+(3x4)

Double ⇒ guarda números; String ⇒ guarda operandos

Proceso: Se recorre exp de tope a abajo, excluyendo posibilidades. Aux: pila que ayuda

```
public static Double PNvalue(Pila exp){
  Pila aux = new PD();  Double x, y; String op;
  try{while(!exp.EsVacia()){
      if(exp.Tope() instanceOf Double && (!aux.EsVacia() && aux.Tope() instanceOf Double)){
        x = (Double) aux.Tope(); 1
        aux = aux.Desapila(); 2
        y = (Double) exp.Tope(); 3
        exp = exp.Desapila 4
        op = (String) aux.Tope(); 5
        aux = aux.Desapila(); 6
      exp.Apila( eval(x, op, y)); 7
      {else} aux.Apila(exp.Tope()); 8
        exp = exp.Desapila(); 9
      }
  {return (Double) aux.Tope()
}catch(TADVacioException){} sout "ex" {{
```

¿Qué hace?

1		2		3		4		5	
	x				y				op
	1								
3	-	3	-	3	-		-		-
exp	aux	exp	aux	exp	aux	exp	aux	exp	aux

6		7		8		9	
		eval(1,-,3) = -2					
				-2			-2
exp	aux	exp	aux	exp	aux	exp	aux