

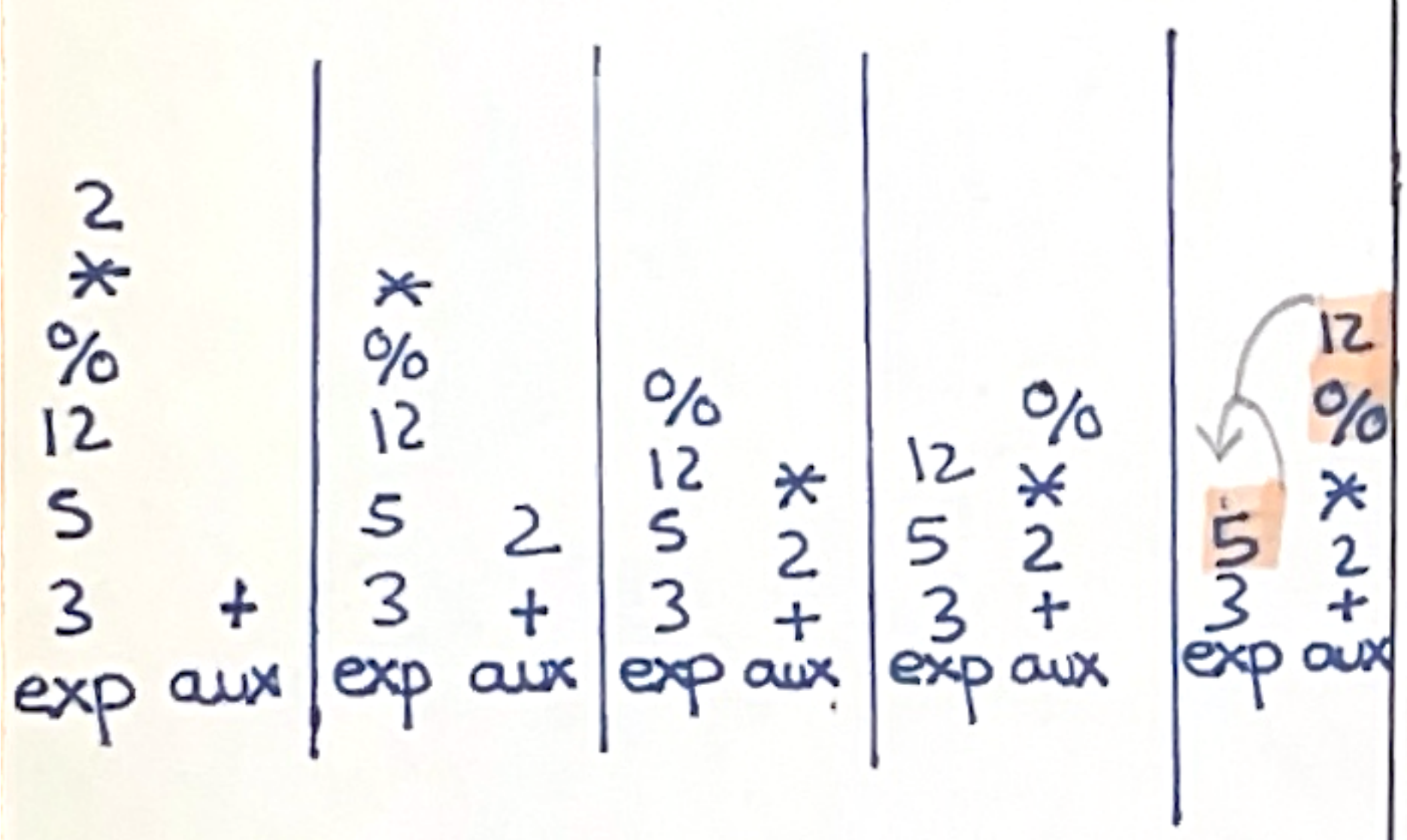
Polish Notation

$$2 + ((12 \% 5) * 3)$$

PN

$2 + ((12 \% 5) * 3)$
 $+ 2 ((12 \% 5) * 3)$
 $+ 2 (x (12 \% 5) 3)$
 $+ 2 (x (\% 12 5) 3)$
 $+ 2 x \% 12 5 3$

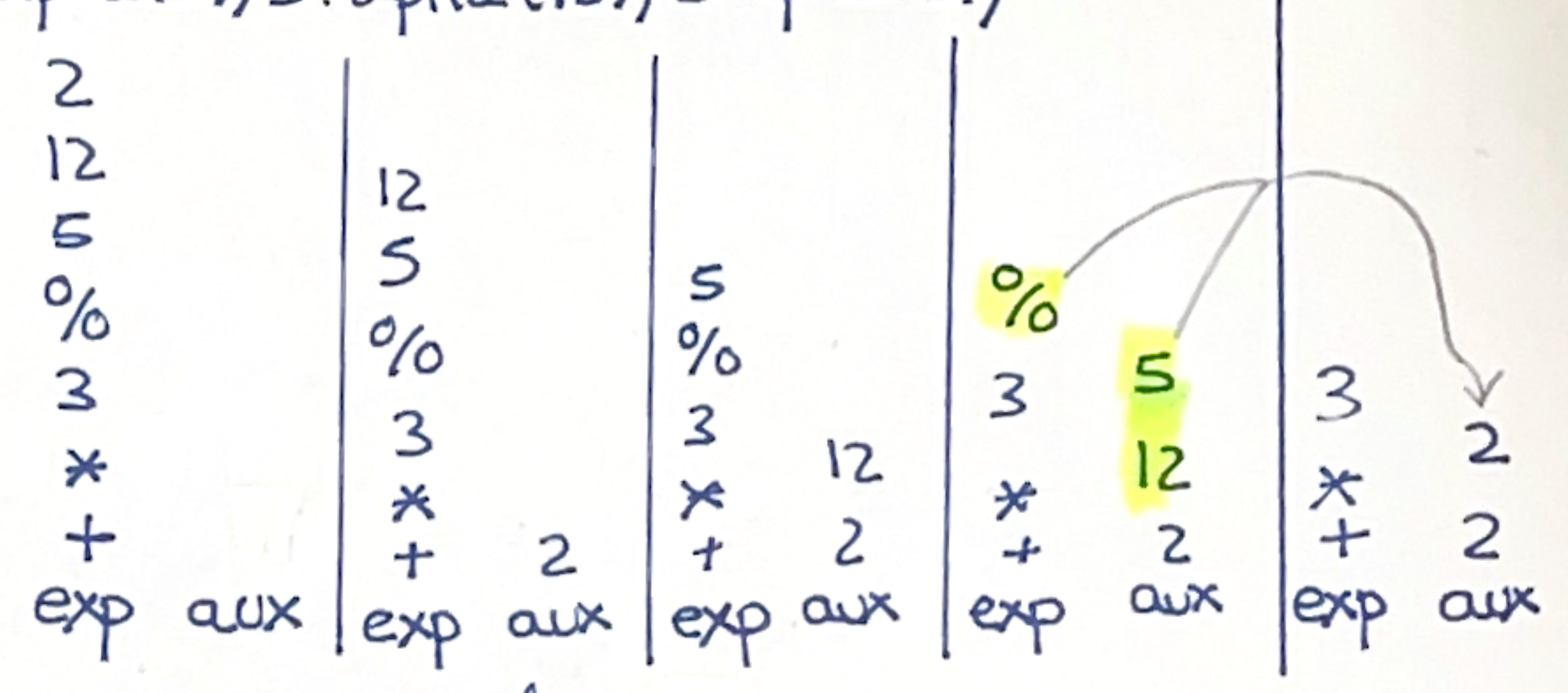
Se apila de derecha a izquierda
 $s.apila(3); s.apila(5); s.apila(12);$
 $s.apila(\%); s.apila(*); s.apila(2);$
 $s.apila(+);$



RPN (Reverse)

$2 + ((12 \% 5) * 3)$
 $2 ((12 \% 5) * 3) +$
 $2 ((12 \% 5) 3) * +$
 $2 ((12 5) \% 3) * +$
 $2 12 5 \% 3 * +$

Se apila de derecha a izquierda
 $s.apila(+); s.apila(*); s.apila(3); s.apila(\%);$
 $s.apila(5); s.apila(12); s.apila(2);$



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

Cdigo

Pila exp $\Rightarrow$ tiene la expresin en PN, ex: $+ 2 * 3 4 \Rightarrow 2 + (3 * 4)$

Double $\Rightarrow$ guarda nmeros; String $\Rightarrow$ guarda operadores

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

```

public static Double PNvalue(Pila exp){
    Pila aux = new PD();
    Double x, y;
    String op;
    try {
        while (!exp.EstVacia()) {
            if (exp.Tope() instanceof Double && (!aux.EstVacia() && aux.Tope() instanceof Double)) {
                // Qu hace?
                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" {
    
```