Stack and Queue

Imp:

- → Infix Evaluation
- → Infix → Postfix & Prefix Conversion
- → Prefix / Postfix - Infix

Expression ⇒ a + b

Infix Expression : a + b → operator, operands. ⇒ 2 * 3 + 6 + 7

Prefix Expression : + ab ⇒ * + 2 3 6

Postfix Expression : ab + ⇒ 2 3 6 * +

Infix Evaluation

Easy

1. You are given an infix expression.
2. You are required to evaluate and print it's value.

Constraints

1. Expression is balanced
2. The only operators used are +, -, *, /
3. Opening and closing brackets - () - are used to impact precedence of operations
4. + and - have equal precedence which is less than * and /. * and / also have equal precedence.
5. In two operators of equal precedence give preference to the one on left.
6. All operands are single digit numbers.

→ 2+6*4/8-3

Str → " 2 + 6 * 4 | 8 - 3 "

⇒ 2

1) balanced
2) operators → +, -, *, /
3) ( )
4) +, - < *, /
5) + , -
6) single digits.

String str = " 2 + 6 * 4 / 8 - 3 " ⇒ str = (2+3)*(5-6)+(7*8)

6 * 4 / 8 → 2

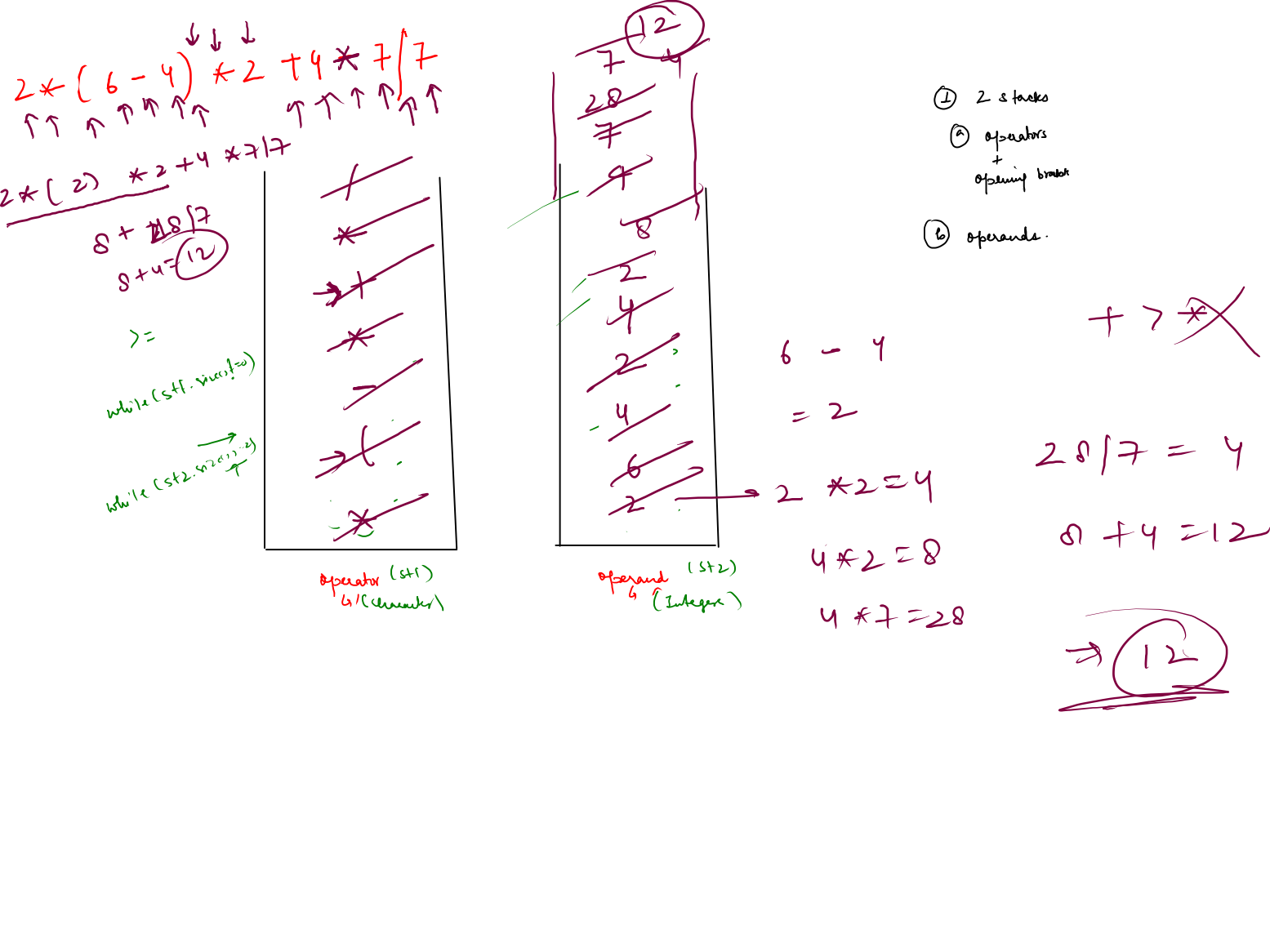
12 + 6 + 234 * 2 - 4 ⇒ +, -, *, /

⇒

(2+6) * 4 / 8 - 3

a) +, - ⇒ same precedence
b) *, / ⇒ same precedence.

*, / > + -


2 * (6 - 4) * 2 + 4 * 7 / 7
2 * (2) * 2 + 4 * 7 / 7
8 + 28 / 7
8 + 4 = 12
>=
while (st1.size() != 0)
while (st2.size() != 1)
operator (st1)
(Character)
operand (st2)
(Integer)
12
7
28
7
4
8
2
4
2
4
6
2
1) 2 stacks
2) operators + opening bracket
b) operands.
6 - 4
= 2
2 * 2 = 4
4 * 2 = 8
4 * 7 = 28
+ > *
28 / 7 = 4
8 + 4 = 12
12

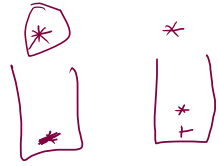
→ Steps to evaluate infix expression :-

(1) Make 2 stacks (1) for operands (Integer) (st1) (2) for operators | opening bracket (character) (st2)

precedence(ch) >= precedence(a,b)

(2) (a) ch = '(' → st2

(b) ch = +, -, *, / ⇒ i) while (precedence(st2.peek()) >= p(ch))
↳ take 1 operator
↳ take 2 operands → st1.push(op^n)
ii) st1.push(a);



(c) ch == ')' perform 2a)i) (stop when st2.peek() == '(')

(3) When done with expression perform 2a)i) until either (st1.size() == 1) or (st2.size() != 0)

+, - = 1

*, / = 2

(=> 0

your(*) = 2 > 1 (+) = 1

(2 * 4 - 5) * (7 / 7 - 6 + 5 - 9) * 10

Ans = -90

$-9 * 10 = -90$

$3 * -90 = -270$

$= -270$

2-3 dry run
+
7-8 coding

10 mins

12 mins

$2 * 4 = 8$

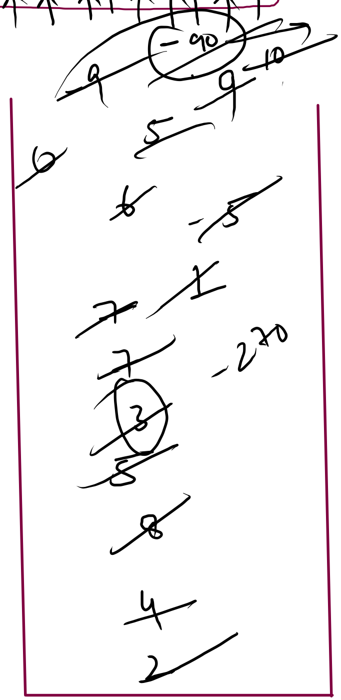
$8 - 5 = 3$

$7 / 7 = 1$

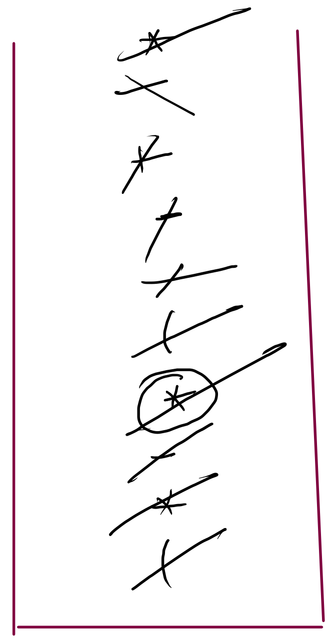
$1 - 6 = -5$

$5 - 5 = 0$

$0 - 9 = -9$



st1 (operands)



st2 (operators)

```java
public static int precedence(char ch){
    if(ch == '+' || ch == '-') return 1;
    else if (ch == '*' || ch == '/') return 2;
    else return 0;
}


public static int operation(int a, int b, char ch){
    if (ch == '+') return a+b;
    else if (ch == '-') return a-b;
    else if (ch == '*') return a*b;
    else return a/b;
}
```

```java
public static void main(String[] args) {
    /* Enter your code here. Read input from STDIN. Print output to STDOUT. Your class sh
    Scanner scn = new Scanner(System.in);
    String str = scn.nextLine();

    Stack<Integer> st1 = new Stack<>();
    Stack<Character> st2 = new Stack<>();

    for(int i=0;i<str.length();i++){
        char ch = str.charAt(i);

        if (ch == '('){
            st2.push(ch);
        } else if (ch == ')'){
            while(st2.peek() != '('){
                char op = st2.pop();
                int b = st1.pop();
                int a = st1.pop();
                int ans = operation(a,b,op);
                st1.push(ans);
            }
            st2.pop();
        } else if (ch == '+' || ch == '-' || ch == '*' || ch == '/'){
            while(st2.size() >0 && precedence(ch) <= precedence(st2.peek())){
                char op = st2.pop();
                int b = st1.pop();
                int a = st1.pop();
                int ans = operation(a,b,op);
                st1.push(ans);
            }
            st2.push(ch);
        } else if (ch >='0' && ch <='9'){
            st1.push(ch-'0');
        }
    }

    while(st2.size() > 0){
        char op = st2.pop();
        int b = st1.pop();
        int a = st1.pop();
        int ans = operation(a,b,op);
        st1.push(ans);
    }


    System.out.println(st1.peek());

}
```

1) Knight Tour → Sunday.

2) Time complexity of recursion

2nd Sept. → Saturday