Stack

- push() → It pushes the element in the top.
- pop() → It deletes elements from the top
- isEmpty() → It gives T/F value if element is present or not
- size() → It gives no of elements in the Stack
- peek() → It gives the value of top element in stack.

```
Stack<Type> st = new Stack();
```

Java ⇐

- Stack → class name
- Type
- st → variable name
- new → used to allocate memory
- Stack() → constructor()

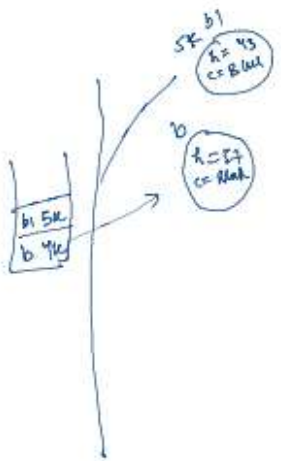
```
class Solution {
    Solution(){
        SOP("I am constructor");
    }
}
```

① no return statement (Type)
② Same as class name

```
Solution s = new Solution();
```

```
public class Bird {
    int h;
    String c;
    Bird(h, c);
        this.h = h
        this.c = c
}

Bird b = new Bird(34, Black);
Bird b1 = new Bird(43, Blue);
b.height =
b.colour =
```



1. Java will provide you by default constructor if no constructor is created by user.
2. Constructor is executed every time when object is created.
3. Constructor can have any # of parameters.
4. Constructor is used to initialize the properties of object.


```
public class Solution {
    ArrayList<Integer> stack;
    public Solution(){
        stack = new ArrayList<>();
    }
    public void push(int x){
        stack.add(x);
    }
    public void pop(){
        if(stack.size()>0){
            // System.out.println("Hi");
            stack.remove(stack.size()-1);
            // System.out.println(stack.size());
        }
        //
    }
    public void size(){
        System.out.println(stack.size());
    }
    public void display(){
        for(int i=0;i<stack.size();i++){
            System.out.print(stack.get(i)+" ");
        }
        System.out.println();
    }
```

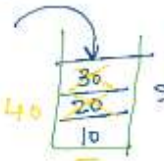
constructor

Solution stack = new Solution()

stack.push()
stack.pop()
stack.peek()
stack.size()
stack.display()

= 2
(10 40]

using ArrayList



40

st Java stack

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);

    int n = scn.nextInt();
    Solution stack = new Solution();
    int i =1;
    while(i<=n){
        String str = scn.next();
        //String[] command= str.split(" ");
        //String s = command[0];
        if(str.equals("push")){
            int x = Integer.parseInt(scn.next());
            stack.push(x);
        }else if(str.equals("pop")){
            stack.pop();
        }else if(str.equals("display")){
            stack.display();
        }else{
            stack.size();
        }

        i++;
    }
    /* Enter your code here. Read input from STDIN. Prin
}
```

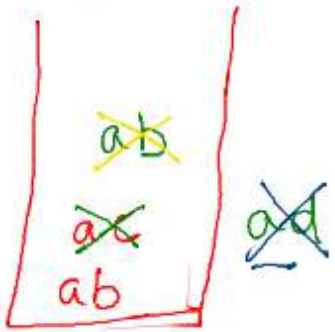
push 10
push 20
push 30

pop()
pop()
size() = 2

push 40
display

Question Delete consecutive

String = [ab ac ac ad ab ab ad]

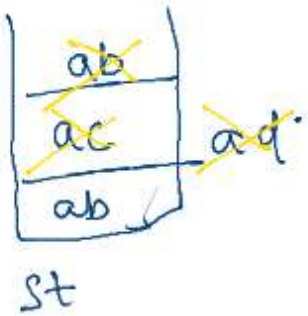


```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    int n = scn.nextInt();
    String[] arr = new String[n];
    for(int i=0;i<n;i++){
        arr[i]= scn.next();
    }

    Stack<String> st = new Stack<>();

    for(int i=0;i<arr.length;i++){
        String s = arr[i];

        if(st.size()>0 && st.peek().equals(s)){
            st.pop();
        }else{
            st.push(s);
        }
    }
    System.out.println(st.size());
    /* Enter your code here. Read input from STDIN. Print outpu
}
```



Question → Valid parenthesis | Balanced Parenthesis

((()((()))) ())

()()()

(()()) ≠ false

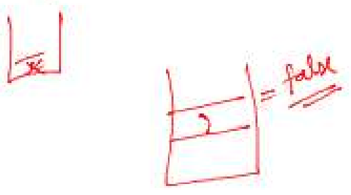
()()) ≠ false

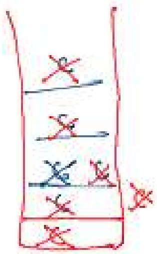
when opening '(' is completely paired by ')' we can say the string is balanced

0 1 2 3 4 5 6 7 8 9 10 11 12

((()((()))) ())

"()))"





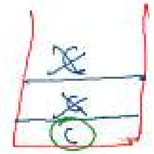
if (str.charAt(i)== '(')
    st.push('(');

if (st.size()>0 && st.peek()== '(') {
    st.pop();
}

true

0 1 2 3 4

(()()



st

false

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str= scn.nextLine();
    Stack<Character> st = new Stack<>();

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

        if(ch =='('){
            st.push(ch);
        }else{
            if(st.size()>0 && st.peek()=='('){
                st.pop();
            }else{
                System.out.println("false");
                return;
            }
        }
        i++;
    }

    if(st.size()>0){
        System.out.println("false");
    }else{
        System.out.println("true");
    }
    /* Enter your code here. Read input from STDIN. Print output to
}
```

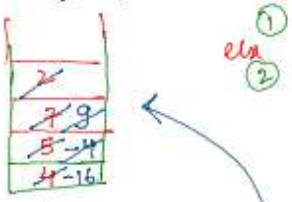
Qus Expression.

1. infix = a+b (operator in b/w) — 10+20=30
2. prefix = +ab (operator before operands) — +1*20=30
3. postfix = ab+ (operands before operator) — 10 20 + =30

ab+ ⇒ evaluate and print output

4572+-*

i i i i



if( ) ↓ a
1. operands – push it
else
2. operator.
pop two element
a
b
and do operation
4
push again

a=-4
b=4
a*b
= -16

a=2
b=7
+ = 9

a=9
b=5
- = -4
b-a

-18

```
public static void main(String[] args) {
    Scanner scn = new Scanner(System.in);
    String str= scn.nextLine();

    Stack<Integer> st = new Stack<>();

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

        if(ch >='0' && ch<='9'){
            st.push(ch-'0');
        }else{
            int a= st.pop();
            int b = st.pop();
            int val=0;
            if(ch=='+'){
                val= a+b;

            }else if(ch=='-'){
                val =b-a;
            }else if(ch=='*'){
                val = a*b;
            }else{
                val = b/a;
            }
            st.push(val);
        }
    }

    System.out.println(st.peek());
```