Implement a stack using ArrayList

arraylist

0	1	2	3	4	
5	10	15	18	7	

st.push(5)

st.push(10)

" (15)

" (18)

" (7)

globally declared

```java
public class Solution {

    static ArrayList<Integer> st;

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

        st = new ArrayList<Integer>();

        int t = scn.nextInt();
        for (int i = 0; i < t; i++) {
            String str = scn.next();
            if (str.equals("push")) {
                int x = scn.nextInt();
                pushElement(x);
            } else if ( str.equals("display") ) {
                displayStack();
            } else if ( str.equals("size") ) {
                sizeOfStack();
            } else if ( str.equals("pop") ) {
                removeElement();
            } else {
                System.out.println("Invalid Input");
            }
        }
    }
```

initialised inside

```java
    public static void pushElement(int x) {
        st.add(x);
    }

    public static void removeElement() {
        if (st.size() == 0) {
            // System.out.println("Underflow");
            return;
        }
        st.remove( st.size() - 1 );
    }

    public static void sizeOfStack() {
        System.out.println( st.size() );
    }

    public static void displayStack() {
        for (int i = 0; i < st.size(); i++) {
            System.out.print( st.get(i) + " " );
        }
        System.out.println();
    }
}
```

T.C of all functions are O(1)

valid parentheses 10 ☆☆☆ M.M. Imp

str = "((()()))()" valid

str = ")()()))" invalid

str = ")))(((" invalid

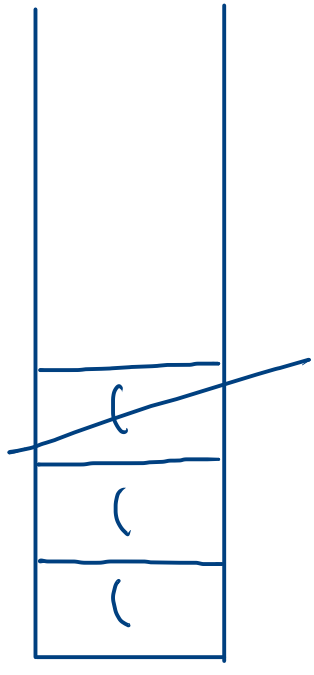
str = ((( )) ( ))

<u>approch</u> :- we will try to delete all valid pairs from stack

then

if str is valid, stack will be empty in last, else some values will be there

str = (((_)) ( ))



if ch == '('

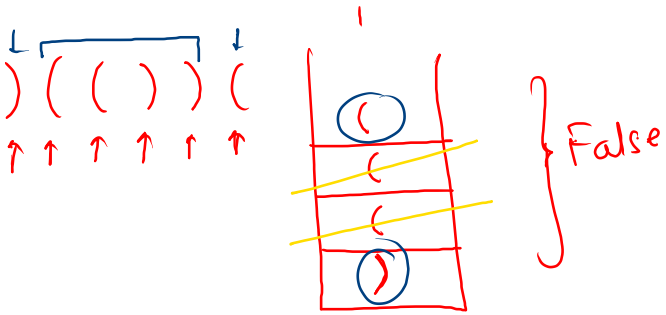
push in stack

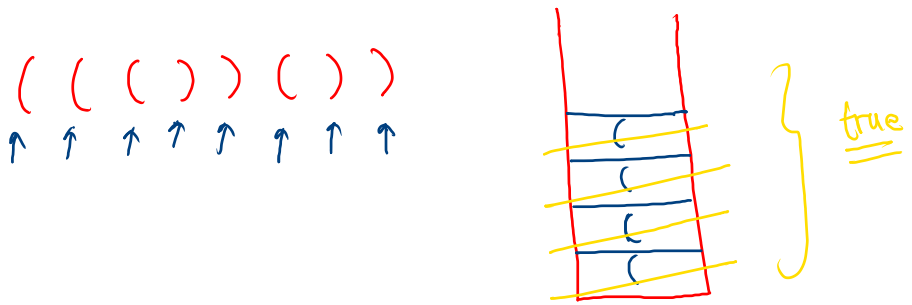
else if ch == ')'

then element at top of stack must be '('

code

```java
public static boolean validParanthese(String str) {
    Stack<Character> st = new Stack<Character>();
    for (int i = 0; i < str.length(); i++) {
        char ch = str.charAt(i);
        if ( st.size() > 0 && ch == ')' && st.peek() == '(' ) {
            st.pop();
        } else {
            st.push(ch);
        }
    }
    return st.isEmpty();
}
```





Postfix expression calculation

infix expression $\Rightarrow ((4+5)*(7-2)) \Rightarrow (9*5) \Rightarrow 45$

ans

postfix expression $\Rightarrow ((\underline{4\ 5+})*(\underline{7\ 2-}))$

$\Rightarrow 45+72-*$

prefix expression $\Rightarrow ((\underline{+45})*(\underline{-72}))$

$\Rightarrow *+45-72$

infix $\Rightarrow (((1*2)/(3+7))-(7\%2))$

postfix $\Rightarrow (((\underline{12*})/(\underline{37+}))-(\underline{72\%}))$

$\Rightarrow (12*37+/)-(72\%)$

$\Rightarrow 12*37+/72\%-$

$$\Rightarrow ((\underbrace{1 * 2}) * \underbrace{(3 + 7)} / \underbrace{(5 + 7)})$$

$$\Rightarrow ((12*) * \underbrace{(37+)} / \underbrace{(57+)})$$

$$\Rightarrow (12*) * (37+57+/)$$

$$\Rightarrow \underline{\underline{12*37+57+/*}}$$

priority order

%

/ *

+ −