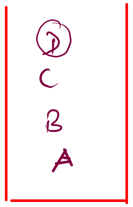
Revision

Stack → DS
↳ LIFO



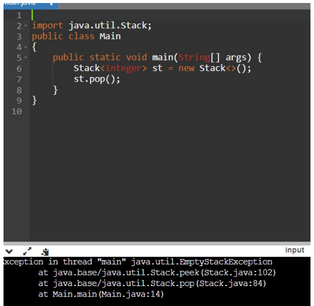
remove / get.
↳ top ele
"D"

D.S.

| | |
|---|---|
| Initialize | Stack < Integer > st = new Stack<>(); |
| add | • push ( ) |
| get | • peek ( ) |
| remove | • pop( ) |
| size | • size ( ) |

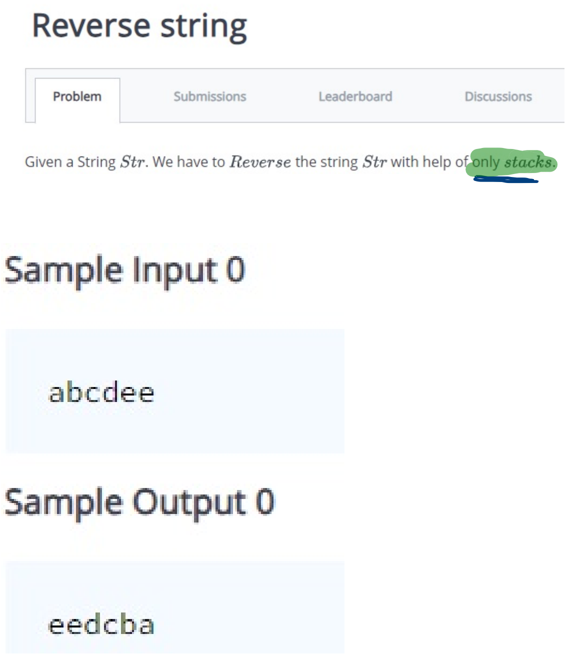
size == 0

Can I remove / get from stack?



```
import java.util.Stack;
public class Main
{
    public static void main(String[] args) {
        Stack<Integer> st = new Stack<>();
        st.pop();
    }
}
```

```
xception in thread "main" java.util.EmptyStackException
        at java.base/java.util.Stack.peek(Stack.java:102)
        at java.base/java.util.Stack.pop(Stack.java:84)
        at Main.main(Main.java:14)
```



## Reverse string

Problem | Submissions | Leaderboard | Discussions

Given a String *Str*. We have to *Reverse* the string *Str* with help of only *stacks*.

### Sample Input 0

```
abcdee
```

### Sample Output 0

```
eedcba
```

$s \rightarrow \boxed{a}\boxed{b}\boxed{c}\boxed{d}\boxed{e}\boxed{e}$

$ans \rightarrow$ eedcba.

?

"e"e"d"cba"

ans = ""

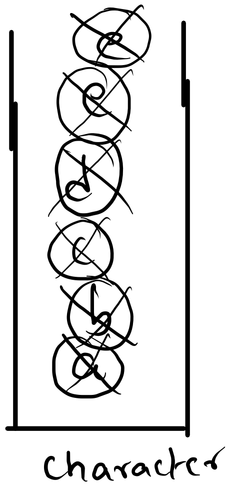
ch = last.

ans += ch.

eedcba.

$s \rightarrow$ abcdee

ans += ch.

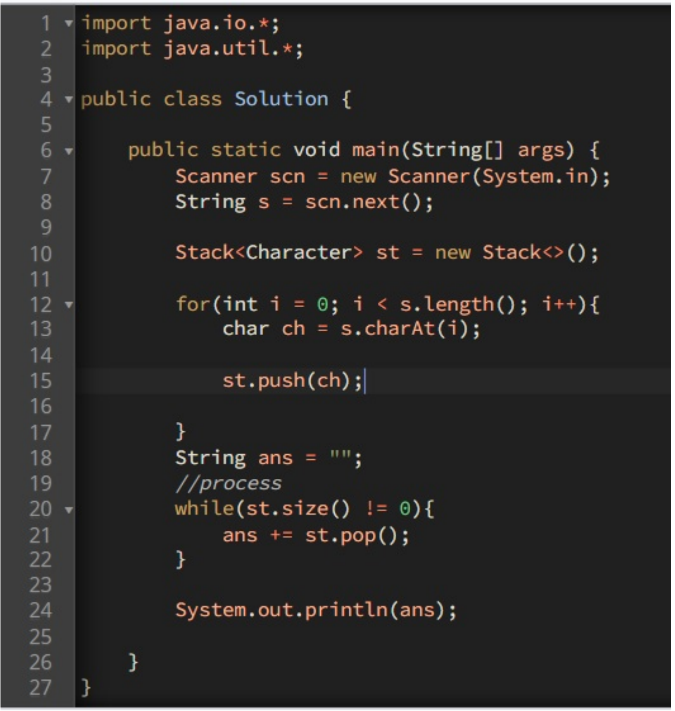


123
↓
string.

st.size() != 0

ans → e e d c b a

✓ ans.charAt(3) → b



```java
import java.io.*;
import java.util.*;

public class Solution {

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

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

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

            st.push(ch);

        }
        String ans = "";
        //process
        while(st.size() != 0){
            ans += st.pop();
        }

        System.out.println(ans);

    }
}
```

ans = ""

ans = $\boxed{\text{"a"}}$

1. = "ab"

2. = "abc"

3 = "abcd"

⋮

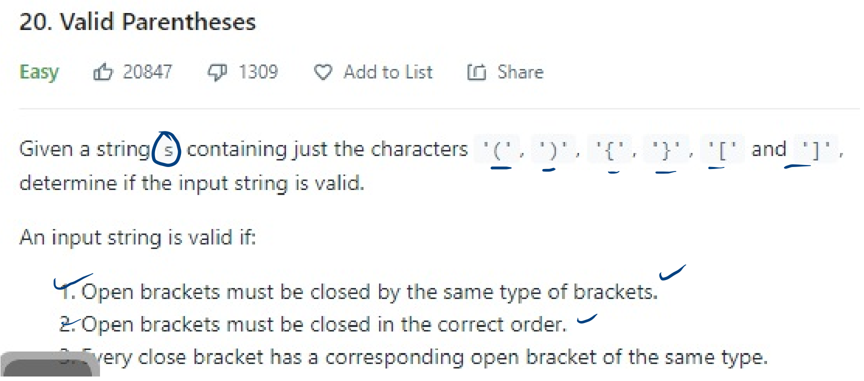
+ $\boxed{n-1}$ $n$th

$1 + 2 + 3 + \dots + n-1$

$\sum_{1}^{n} n = \frac{n(n+1)}{2}$

$= \frac{(n-1)(n)}{2}$

$= O(n^2)$



20. Valid Parentheses

Easy 20847 1309 Add to List Share

Given a string s containing just the characters '(', ')', '{', '}', '[' and ']', determine if the input string is valid.

An input string is valid if:

1. Open brackets must be closed by the same type of brackets. ✓
2. Open brackets must be closed in the correct order. ✓
3. Every close bracket has a corresponding open bracket of the same type.

s → $\overset{\checkmark}{(}\ \overset{\checkmark}{)}$ → valid.

s → "{ }" → valid.

s → $\overset{\checkmark}{\{}\ \overset{\checkmark}{(}\ \overset{\checkmark}{)}\ \overset{\times}{]}$ → not valid.

s → "[ ( ] )" ↳ false/n.v.

s → " ) ( " ↳ not valid.

s → " ( ( ) ) " ↳ true.

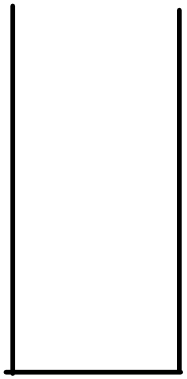
s → " ( " ⟶ not valid.

s → " $\overset{\checkmark}{(}\ \overset{\checkmark}{)}\ \overset{\times}{]}$ " → false.

Valid.

s → " { ( [ $\overset{i}{(}$ ) ] ) } "

i



→ Empty

↳ true

or false

Open → push

close.

↳ ch(i) → )

peek → (

ch(i) → }

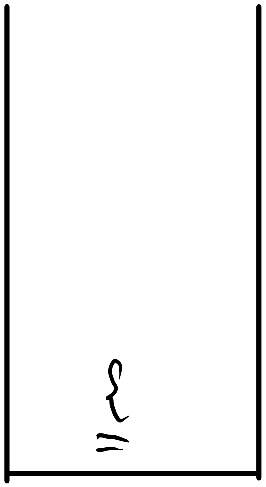
peek → {

ch(i) ⇒ ]

peek → [

Invalid.

$$\{\ (\ )\ [\ ]\ \underset{\underset{i}{=}}{]}\ \}$$



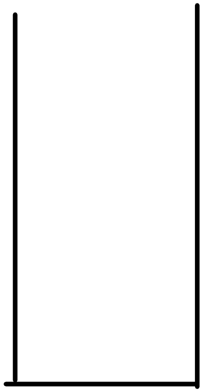
$\rightsquigarrow$ ans $\rightarrow$ false $\leftarrow$.

Invalid.

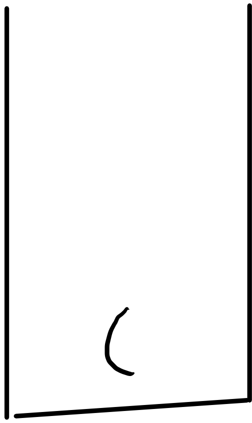
s → " ) "
i

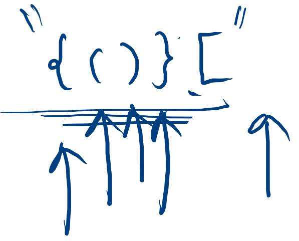
or s → " ( "
i

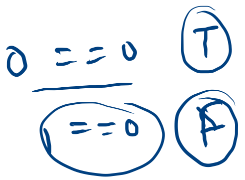
stack empty but string remaining ⇒ F

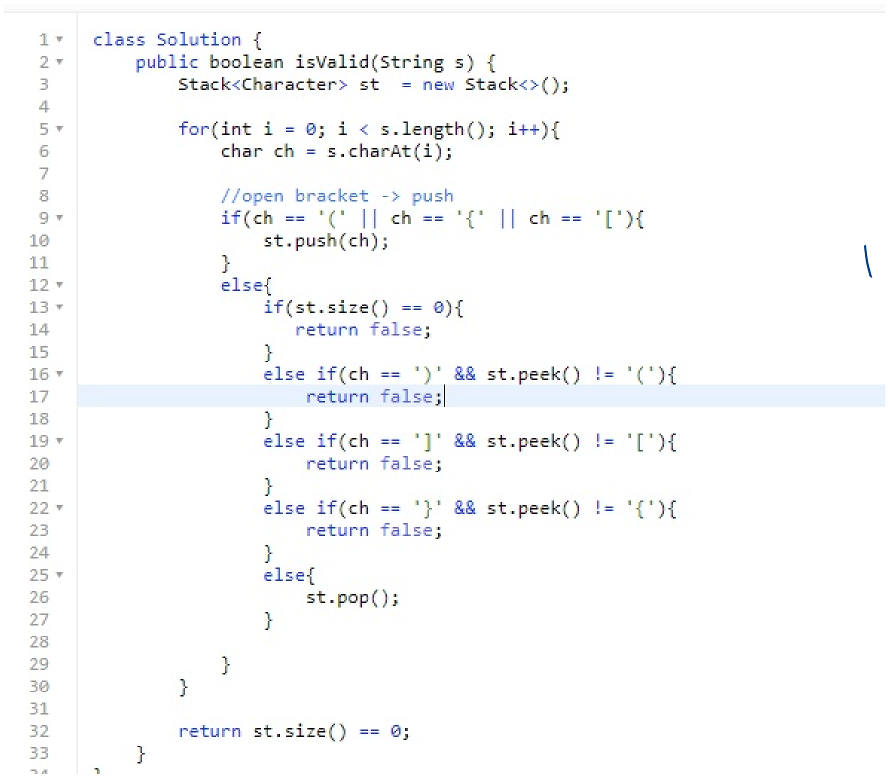


F ⇐

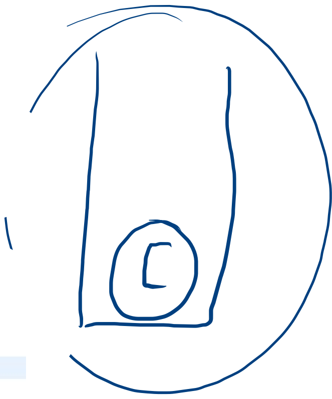


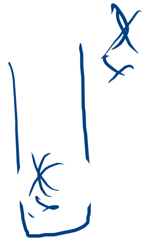






```java
class Solution {
    public boolean isValid(String s) {
        Stack<Character> st  = new Stack<>();

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

            //open bracket -> push
            if(ch == '(' || ch == '{' || ch == '['){
                st.push(ch);
            }
            else{
                if(st.size() == 0){
                   return false;
                }
                else if(ch == ')' && st.peek() != '('){
                    return false;
                }
                else if(ch == ']' && st.peek() != '['){
                    return false;
                }
                else if(ch == '}' && st.peek() != '{'){
                    return false;
                }
                else{
                    st.pop();
                }

            }
        }

        return st.size() == 0;
    }
}
```





# Postfix expression calculation

$a+b \rightarrow$ infix

$+ab \rightarrow$ prefix

$ab+ \rightarrow$ postfix.

## Sample Input 0

```
4572+-*
```

4 5 7 2 + - *

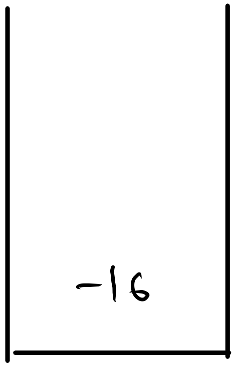
=

## Sample Output 0

```
-16
```

$\underline{4} * \underline{-4} =$

if we don't have operator we will not perform operation



$\underline{+\ 2\ 3.}\ \rightsquigarrow$ result $\rightarrow$ invalid.

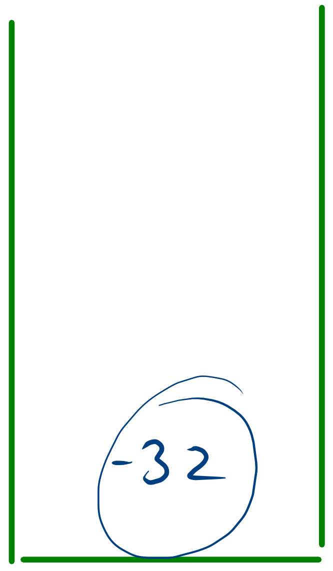
$\underline{\underline{2\ 3\ 4}}\ \rightsquigarrow$ invalid.

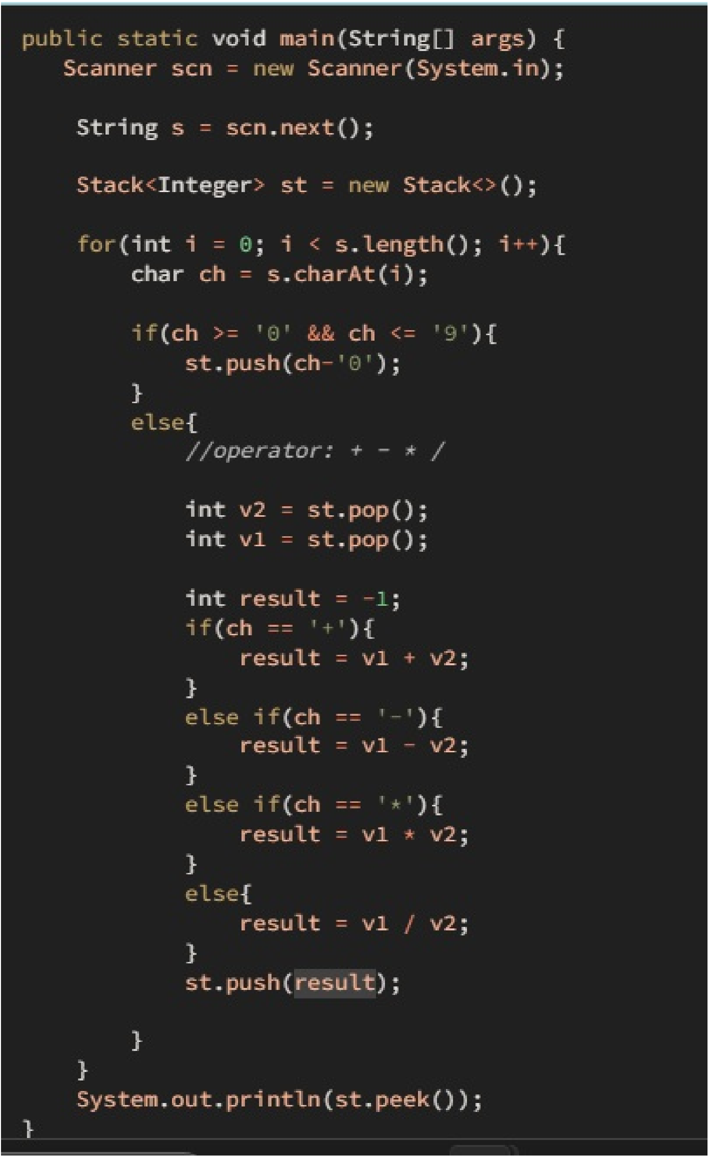
$+\ -\ *\ \rightsquigarrow$ invali$\underline{d}$.

$2\ 3\ 4\ 5\ \underline{+}\ \rightsquigarrow$ invalid.

$\left.\right\}$ not as i/p.

2 3 7 + − 4 *





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

    String s = scn.next();

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

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

        if(ch >= '0' && ch <= '9'){
            st.push(ch-'0');
        }
        else{
            //operator: + - * /

            int v2 = st.pop();
            int v1 = st.pop();

            int result = -1;
            if(ch == '+'){
                result = v1 + v2;
            }
            else if(ch == '-'){
                result = v1 - v2;
            }
            else if(ch == '*'){
                result = v1 * v2;
            }
            else{
                result = v1 / v2;
            }
            st.push(result);

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

$$\begin{tikzcd}
\text{ArrayList} \arrow[r] \arrow[d] & \text{add} \\
 & \text{get} \\
\text{class.} \arrow[ur, dash] \arrow[uur, dash] \arrow[r, dash] \arrow[dr, dash] & \text{set} \\
 & \text{remove}
\end{tikzcd}$$

```
~~~~~~   int age;
______   String name;
```
$\Big]\rightarrow$ data members

```
public void printname()
{
          → body – ✓
}
```
$\Big]\rightarrow$ member function.

List.

↳ Interface.

```
List {      public void  add( int x);



}
```

List

Interface

don't have body of method.

add

get

set

remove

size

→ Class.

(implementation

AL

Linked

Vector

Stack.