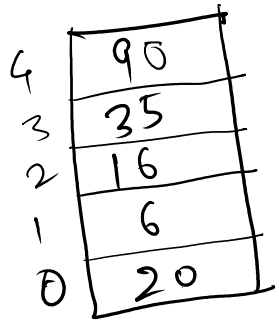
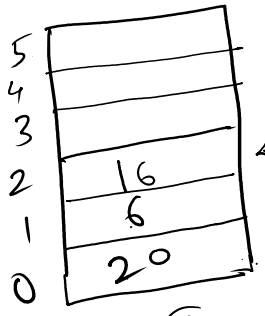
Stack

↳ It is a data structure, where the elements are inserted in LIFO or FILO order.



Last In First Out

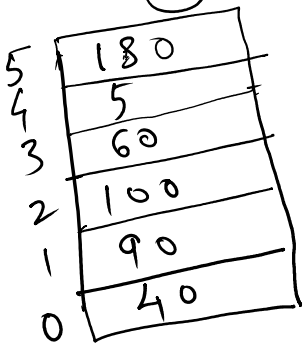
First In Last Out



top = -1.

arr[top] : arr[2] = 16.

(6) ← top.



top = -1.

Stack internally uses a variable which is top, which initially -1 and when values are added then top increments.

1. push(Type element) → we add element into stack using this method.

1. push() ⇒ It is used to insert the value in stack using this method.

2. pop() ⇒ It is used to remove the value at the top which means last inserted value.

In Java, Stack is a class which has inbuilt methods.

Syntax

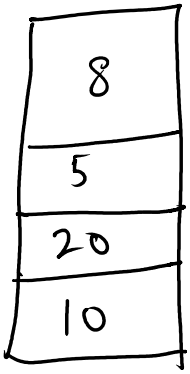
```
Stack<Type> stack_name = new Stack<>();
```

3. peek() :- It is used to display the value present on top of stack.

Example.

[10, 20, 5, 8]

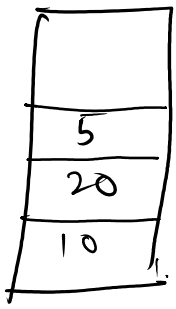
```
Stack<Integer> st = new Stack<>();
st.push(10);
st.push(20);
st.push(5);
st.push(8);
int x = st.pop();
```



```
int x = st.pop();
S.o.p(x); // 8

int x = st.peek();
S.o.p(x) // 5
```

no change



4. size() → It returns the length of stack

Stack Syntax Learning

```
    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print output to
        Stack<Integer> st = new Stack<>();
        Scanner sc = new Scanner(System.in);
        int t = sc.nextInt();
        for(int i=0;i<t;i++){
            int n = sc.nextInt();
            int x =0;
            if(n==3){
                x = sc.nextInt();
            }
            switch(n){
                case 1 : System.out.println(st.size());
                    break;
                case 2 : if(st.isEmpty()){
                    System.out.println(-1);
                    }else{
                    st.pop();
                    }
                    break;
                case 3 : st.push(x);
                    break;
                case 4 : if(st.isEmpty()){
                    System.out.println(-1);
                    }else{
                    System.out.println(st.peek());
                    }
                    break;

            }
        }
    }
}
```

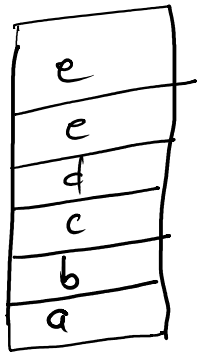
Reverse String

str = "abcdee"

Output

"eedcba"

After push →



after pop ↙

e, e, d, c, b, a

Code

```
Stack<Character> st = new Stack<>();
for(int i=0; i<str.length(); i++){
    st.push(str.charAt(i));
}
String revstr = "";
While(!st.isEmpty()){
    revstr += st.pop(); //eedcba
```

```
}
S.O.pln(revstr);
```

Delete Consecutive

n = 4

aa ab ab ac

Output

2

Input 2 :

n = 4

aa ab ab aa

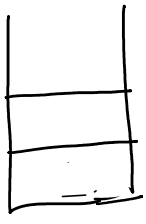
~~aa aa~~

Output

0

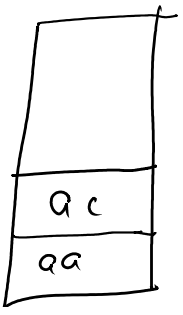
Solution using stack

[0:aa , 1:ab , 2:ab , 3:aa]

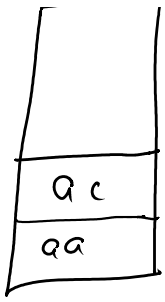


0 1 2 3

aa ab ab ac



→ st.size()



→ st.size()

Code:-

```
Scanner sc = new Scanner(System.in);
int n = sc.nextInt();
String str[] = new String [n];
for(int i=0; i<n; i++) {
  str[i] = sc.next();
}
Stack<String> st = new Stack<>();
for(int i=0; i<str.length; i++) {
 if (st.isEmpty() || !str[i].equals(st.peek())) {
    st.push(str[i]);
 }else if (str[i].equals(st.peek()) {
     st.pop();
 }
}
S.O.Pln(st.size());
```

Reverse Words in a given String

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

public class Solution {

    public static void main(String[] args) {
        /* Enter your code here. Read input from STDIN. Print
        Scanner sc = new Scanner(System.in);
        String input = sc.nextLine();
        String str[] = input.trim().split("\\s+");
        Stack<String> st = new Stack<>();
        for(int i=0;i<str.length;i++){
            st.push(str[i]);
        }
        String output = "";
        while(!st.isEmpty()){
            output+= st.pop()+" ";
        }
        System.out.println(output);

    }
}
```