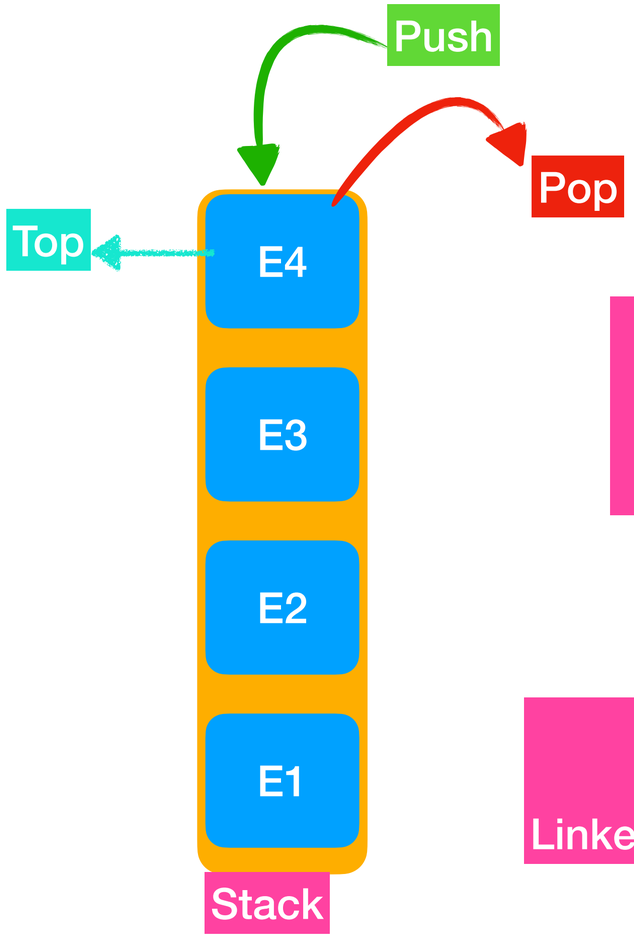


Design Stack

Last In First Out

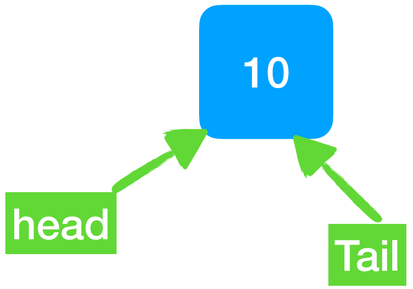
public void push(E) —> Add the Element -> Time Complexity : O(1)
public E pop( ) -> Removes the Last element -> Time Complexity : O(1)
public E top() -> returns last Element -> Time Complexity : O(1)
public int size() -> returns the size -> Time Complexity : O(1)

We can go with either ArrayList
or
LinkedList to design Stack. [LinkedList saves the memory]

Stack → Implementing Stack with LinkedList

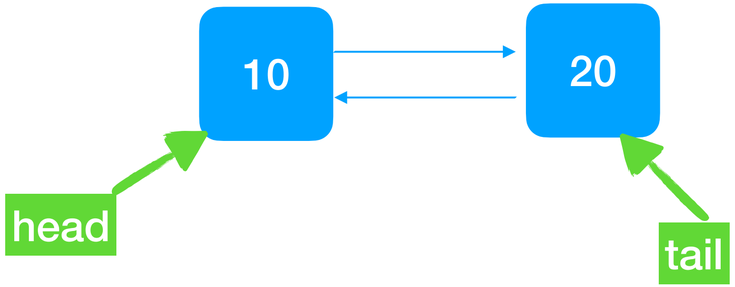
In a LinkedList tail would act as a top.

push(10)




```
push( E)
{
Node node = new Node(E);
if(head == null)
{
head = tail = node;
return;
}
tail.next = node;
node.prev = tail;
tail = node;
}
```

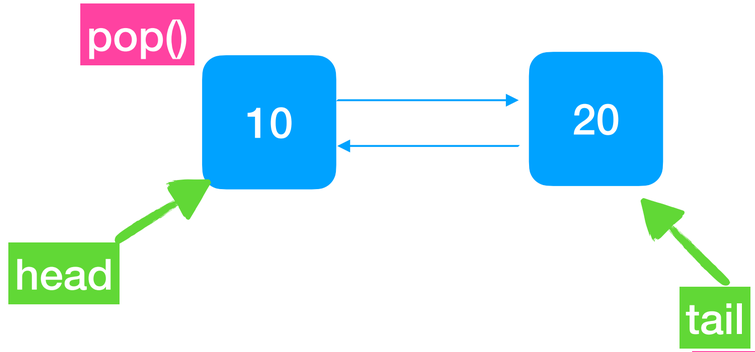
push(20)



top() → tail.val = 20

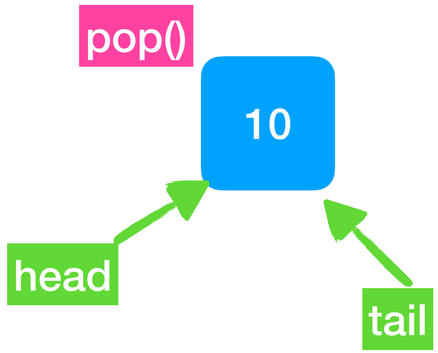
```
top()
{
if(tail == null)
{
Stack is Empty;
}
return tail.val;
}
```

pop()



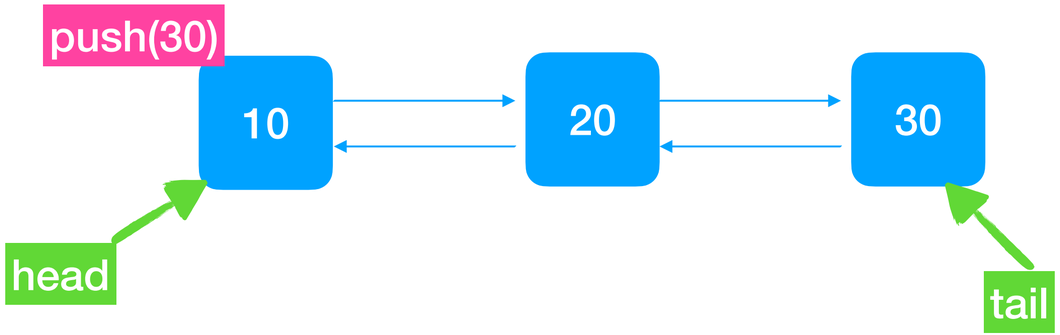

```
pop( )
{
if(tail.prev == head)
{
tail = head = null;
return;
}

Node prev = tai.prev;
prev.next = null;
tail.prev = null;
tail = prev;
}
```

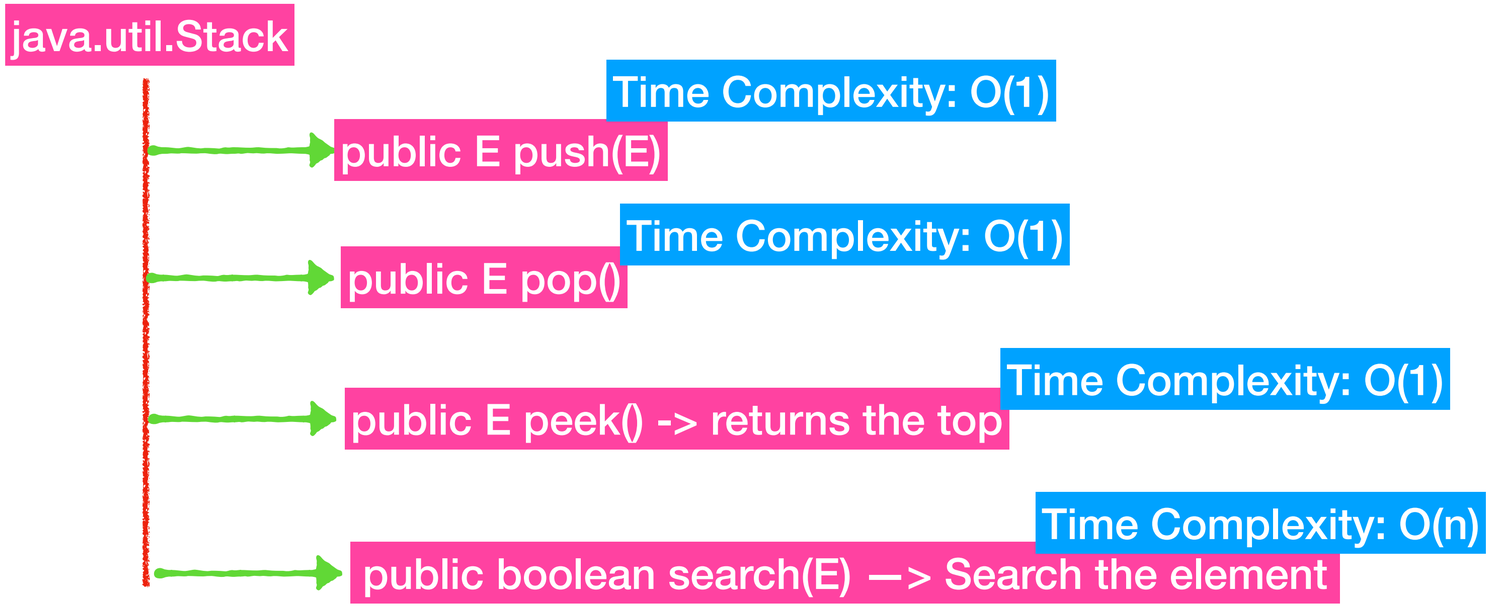
pop()



top() → tail.val = 10

push(30)




java.util.Stack
Time Complexity: O(1)
public E push(E)
Time Complexity: O(1)
public E pop()
Time Complexity: O(1)
public E peek() -> returns the top
Time Complexity: O(n)
public boolean search(E) —> Search the element