Stack is a **Last In, First Out** or **LIFO** that serves as container for elements. It provides 3 operations:

|  |  |
| --- | --- |
| **Method** | **Description** |
| push(element) | **adds** element at the end |
| pop() | **removes and returns** last inserted element |
| peek() | **returns** last inserted element |

Stack can be implemented in 2 ways

Bounded Capacity –using arrays

Unbounded Capacity – using Link List.

We will see both of the implementations in detail.

Stack can throw Exception when we try to push element in stack when stack is full.

Stack can also throw Exception when we try to pop the element in empty stack.

<https://www.cise.ufl.edu/~mssz/DatStrucAlg/DSA-Pt2.html>

Let us understand push(val) and pop() operations from above diagram.

We have 3 inputs as 1, 2, 3.

|  |  |
| --- | --- |
| Operation | Description |
| push(1) | Push 1 into the stack. 1 is top. Stack contains 1. |
| push(2) | Push 2 into the stack. 2 is top. Stack contains 1, 2. |
| push(3) | Push 3 into the stack. 3 is top. Stack contains 1, 2, 3. |
| pop() | pop() the top element. Stack contains 1, 2 |
| pop() | pop() the top element. Stack contains 1 |
| pop() | pop() the top element. Stack is empty |

So this is the bird’s eye view of what stack is and it’s working.

In next post we will implement Stack class of our own and run few test cases against it.