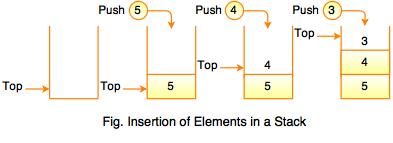
**Stack:**

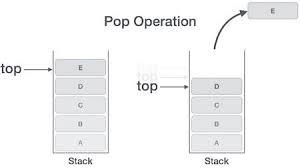
Stack is a collection of similar data items in which both insertion and deletion operations are performed based on LIFO principle”.

* Stack is an ordered list of the same type of elements.
* It is a linear list where all insertions and deletions are permitted only at one end of the list.
* Stack is a LIFO (Last In First Out) structure.
* In a stack, when an element is added, it goes to the top of the stack.

**There are two basic operations performed in a Stack:**  
  
1. Push()  
2. Pop()  
  
**1. Push()** function is used to add or insert new elements into the stack.  
  
**2. Pop()** function is used to delete or remove an element from the stack.

* When a stack is completely full, it is said to be **Overflow state** and if stack is completely empty, it is said to be **Underflow state**.
* Stack allows operations at **one end only**. Stack behaves like a real life stack, for example, in a real life, we can remove a plate or dish from the top of the stack only or while playing a deck of cards, we can place or remove a card from top of the stack only.  
  Similarly, here also, we can only access the top element of a stack.
* According to its LIFO structure, the element which is inserted last, is accessed first.



****The above diagram represents a stack insertion operation. In a stack, inserting and deleting of elements are performed at a single position which is known as, **Top**.  
  
Insertion operation can be performed using Push() function. New element is added at top of the stack and removed from top of the stack, as shown in the diagram below:

An element is removed from top of the stack. Delete operation is based on LIFO principle. This operation is performed using a Pop() function. It means that the insertion and deletion operations are performed at one end i.e at Top.

|  |  |
| --- | --- |
| **Position of Top** | **Status of Stack** |
| -1 | Stack is empty. |
| 0 | Only one element in a stack. |
| N - 1 | Stack is full. |
| N | Stack is overflow. (Overflow state) |

**Some important operation of stack:**

|  |  |
| --- | --- |
| **Operations** | **Description** |
| Peek() | The peek() function gets the top element of the stack, without deleting it. |
| isEmpty() | The isEmpty() function checks whether the stack is empty or not. |
| isFull() | The isFull() function is used to check whether the stack is full or not. |

**Time Complexities of operations on the stack:**

push(), pop(), isEmpty() and peek() all take O(1) time. We do not run any loop in any of these operations.

**Implementation:**   
There are two ways to implement a stack:

* Using array
* Using linked list

Java's library contains a [Stack](https://docs.oracle.com/javase/10/docs/api/java/util/Stack.html) class that is a specialization of [Vector](https://docs.oracle.com/javase/10/docs/api/java/util/Vector.html). Following is an example program in [Java](https://en.wikipedia.org/wiki/Java_(programming_language)) language, using that class.

**import** **java.util.Stack**;

**class** **StackDemo** {

**public** **static** void main(String[]args) {

Stack<String> stack = **new** Stack<String>();

stack.push("A"); *// Insert "A" in the stack*

stack.push("B"); *// Insert "B" in the stack*

stack.push("C"); *// Insert "C" in the stack*

stack.push("D"); *// Insert "D" in the stack*

System.out.println(stack.peek()); *// Prints the top of the stack ("D")*

stack.pop(); *// removing the top ("D")*

stack.pop(); *// removing the next top ("C")*

}

}

**Application of stack:**

**1: Expression evaluation and conversion:**

There are 3 types of expression we use in Programming, which are Infix Expression, Prefix Expression and Postfix Expression.

Infix Expression is represented as X + Y. Prefix Expression is represented as +XY and Postfix Expression is represented as XY+.

In order to evaluate these expressions in Programming, a Data Structure called Stack is used.

Similarly, Stack is also used for Converting one expression into another. For example, converting Infix to Postfix or Infix to Prefix.

**2: Backtracking:**

Backtracking is a recursive algorithm which is used for solving the optimization problem.

So, In order to find the optimized solution of a problem with Backtracking, we have to find each and every possible solution of the problem, doesn’t matter if it is correct or not.

In Backtracking, while finding the every possible solution of a problem, we store the solution of a previously calculated problem in Stack and use that solution to solve the upcoming problems.

**3. Parenthesis Checking**

In Programming, we make use of different type of parenthesis, like – (, ), {, }, which are used for opening and closing a block of code.

So, these parenthesis get stored in Stack and control the flow of our program.

**4. Function Call**

In Programming, whenever you make a call from one function to the another function. The address of the calling function gets stored in the Stack.

So, when the called function gets terminated. The program control move back to the calling function with the help of the address which was stored in the Stack.

So, Stack plays the main role when it comes to Calling a Function from other Function.

**5. String Reversal**

String Reversal is another amazing Application of Stack. Here, one by one each character of the Stack get inserted into the Stack.

So, the first character of the Stack is on the bottom of the Stack and the last character of the String is on the Top of the Stack.

After performing the pop operation in Stack, we get the String in Reverse order.

**6. Syntax Parsing**

As many of the Programming Languages are [context-free languages](https://www.geeksforgeeks.org/check-if-the-language-is-context-free-or-not/). So, Stack is also heavily used for Syntax Parsing by most of the Compilers.

**7. Memory Management**

Memory Management is the important function of the Operating System. Stack also plays the main role when it comes to Memory Management.

**8. In browsers**

The back button in a browser saves all the URLs you have visited previously in a stack. Each time you visit a new page, it is added on top of the stack. When you press the back button, the current URL is removed from the stack, and the previous URL is accessed.