**Stack Data Structure**

A stack is a linear data structure that follows the principle of **Last In First Out (LIFO)**. This means the last element inserted inside the stack is removed first.

## LIFO Principle of Stack

In programming terms, putting an item on top of the stack is called **push** and removing an item is called **pop**.

## Basic Operations of Stack

There are some basic operations that allow us to perform different actions on a stack.

* **Push**: Add an element to the top of a stack
* **Pop**: Remove an element from the top of a stack
* **IsEmpty**: Check if the stack is empty
* **IsFull**: Check if the stack is full
* **Peek**: Get the value of the top element without removing it

## Working of Stack Data Structure

The operations work as follows:

1. A pointer called TOP is used to keep track of the top element in the stack.
2. When initializing the stack, we set its value to -1 so that we can check if the stack is empty by comparing TOP == -1.
3. On pushing an element, we increase the value of TOP and place the new element in the position pointed to by TOP.
4. On popping an element, we return the element pointed to by TOP and reduce its value.
5. Before pushing, we check if the stack is already full
6. Before popping, we check if the stack is already empty

class Stack {

private int arr[];

private int top;

private int capacity;

// Creating a stack

Stack(int size) {

arr = new int[size];

capacity = size;

top = -1;

}

// Add elements into stack

public void push(int x) {

if (isFull()) {

System.out.println("OverFlow\nProgram Terminated\n");

System.exit(1);

}

System.out.println("Inserting " + x);

arr[++top] = x;

}

// Remove element from stack

public int pop() {

if (isEmpty()) {

System.out.println("STACK EMPTY");

System.exit(1);

}

return arr[top--];

}

// Utility function to return the size of the stack

public int size() {

return top + 1;

}

// Check if the stack is empty

public Boolean isEmpty() {

return top == -1;

}

// Check if the stack is full

public Boolean isFull() {

return top == capacity - 1;

}

public void printStack() {

for (int i = 0; i <= top; i++) {

System.out.println(arr[i]);

}

}

public static void main(String[] args) {

Stack stack = new Stack(5);

stack.push(1);

stack.push(2);

stack.push(3);

stack.push(4);

stack.pop();

System.out.println("\nAfter popping out");

stack.printStack();

}

}

## Applications of Stack Data Structure

 **To reverse a word** - Put all the letters in a stack and pop them out. Because of the LIFO order of stack, you will get the letters in reverse order.

 **In compilers** - Compilers use the stack to calculate the value of expressions like 2 + 4 / 5 \* (7 - 9) by converting the expression to prefix or postfix form.