STACK DATA STRUCTURE (LECTURE 1)

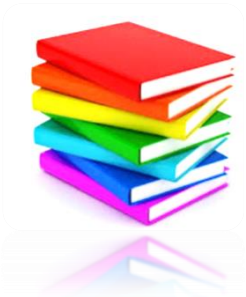
- **Data structure** is a systematic arrangement of data in a computer's memory or sometimes on a disk in order to use data efficiently.

Stack Data Structure

- Stack is a **linear data structure** which follows **LIFO (Last in First out)** principle in inserting and removing elements. Stack is an ordered list of similar data type.
- In a stack all insertions and deletions are made at the top end. Insertions and deletions are restricted from the middle and at the end of a stack.
- Ex: a stack of books

Stack Operations/ Functions

- **push()** - insert a new element into the stack
- **pop()** - remove and return top element from the stack
- **peek()** – return the top element without removing
- **isFull()** - check if stack is full
- **isEmpty()** - check if stack is empty



- Stack is said to be in **Overflow state** when it is completely full and is said to be in **Underflow state** if it is completely empty.

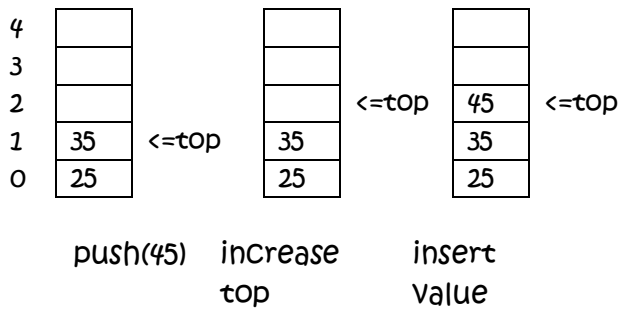
Stack Usage in a Computer System

- **OS:** micro process operations
- **Memory:** storing parameters and return values of functions for any programming language

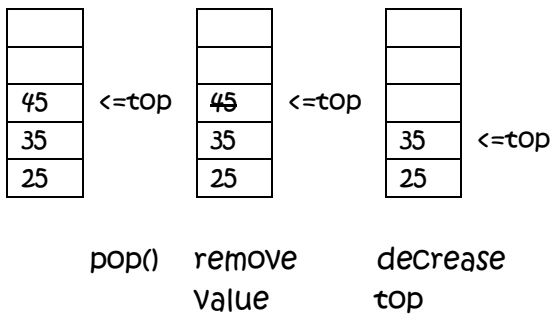
Stack applications

- Recursion handling
- String reversal
- Syntax parsing
- Function call
- Expression evaluation and conversion (prefix, postfix and infix expressions)
- Parenthesis checking
- Backtracking (Conversion of decimal to other number system, Maze tracer, Undo operations)

Stack - push



Stack - pop



Stack implementation

```
class StackX {
	private int maxSize;			//max number of locations
	private double[] stackArray;		//array definition
	private int top;			//top index definition

	public StackX(int s) {			//constructor
		maxSize = s;
		stackArray = new double[maxSize];	//array implementation
		top = -1;
	}

	public void push(double j) {		//push method
		if(top == maxSize-1) {
			System.out.println("Stack overflow");
		}
		else
			stackArray[++top] = j;
	}

	public double pop() {			//pop method
		if(top == -1) {
			System.out.println("Stack underflow");
			return -99;
		}
		else
			return stackArray[top--];
	}

	public double peek() {			//peek method
		if(top == -1) {
			System.out.println("Stack underflow");
			return -99;
		}
		else
			return stackArray[top];
	}

	public boolean isEmpty() {
		return (top == -1);
	}

	public boolean isFull() {
		return (top == maxSize - 1);
	}
}//end of class

class Main{		//main class
	public static void main(String[] args) {
		StackX s = new StackX(5);	//instantiation and constructor calling
		s.push(15);		//push method calling
		s.push(25);
		s.push(35);
		s.push(45);
		System.out.println(s.pop());		//pop method calling
		System.out.println(s.peek());		//peek method calling
	}
}
```