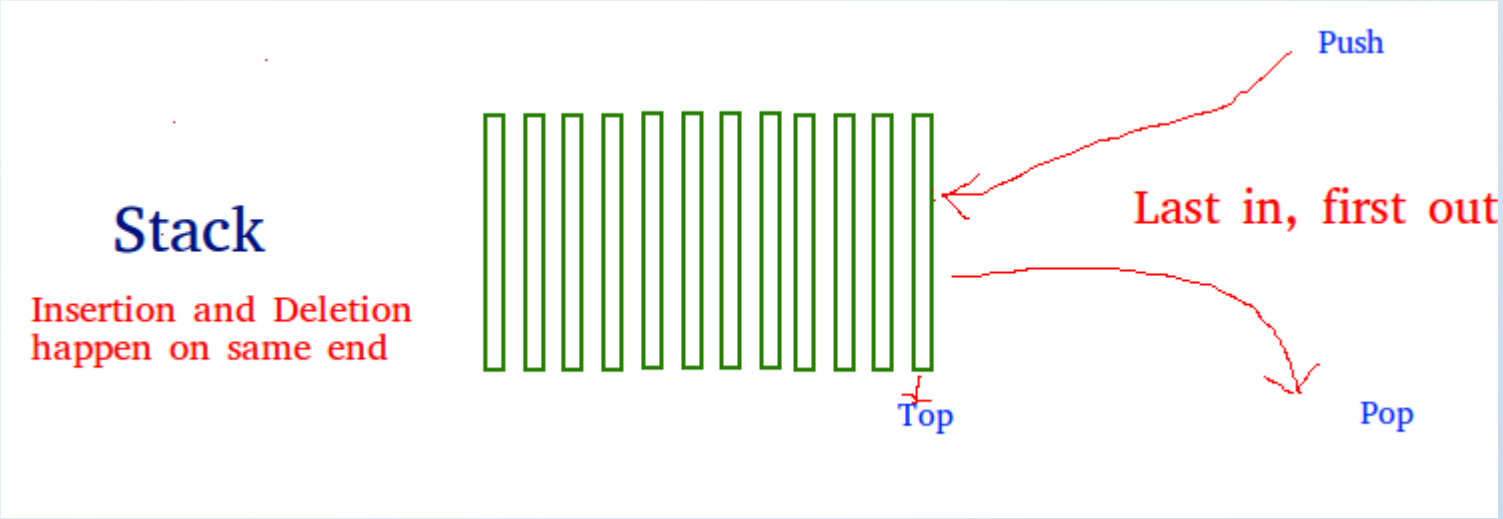
Stack Data Structure
(Introduction and Program)

What is stack?

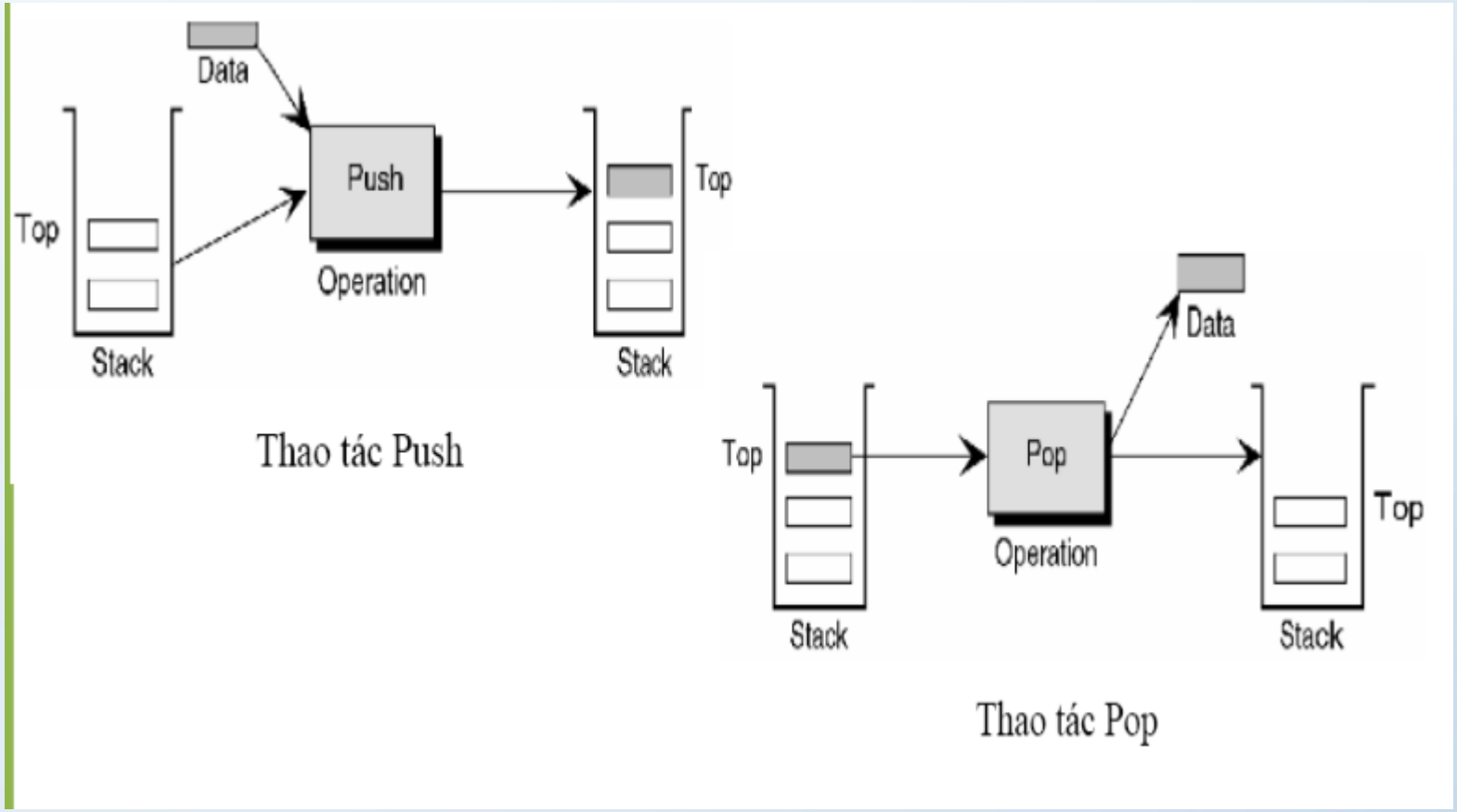
- Stack is a linear data structure which follows a particular order in which the operations are performed.
- The order may be LIFO(Last In First Out) or FILO(First In Last Out).



3 basic operations

- **Push:** Adds an item in the stack. If the stack is full, then it is said to be an Overflow condition.
- **Pop:** Removes an item from the stack. The items are popped in the reversed order in which they are pushed. If the stack is empty, then it is said to be an Underflow condition.
- **Peek or Top:** Returns top element of stack.
- Other: **isEmpty:** Returns true if stack is empty, else false.

Operators


Data
Top
Push
Operation
Top
Stack
Stack
Thao tác Push
Data
Top
Pop
Operation
Top
Stack
Stack
Thao tác Pop

How to understand a stack practically?

- There are many real-life examples of a stack.
- Consider the simple example of plates stacked over one another in a canteen.
- The plate which is at the top is the first one to be removed, i.e. the plate which has been placed at the bottommost position remains in the stack for the longest period of time.
- So, it can be simply seen to follow LIFO/FILO order.

Time Complexities of operations on stack:

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

Push()

```
boolean push(int x)
{
    if (top >= (MAX - 1)) {
        System.out.println("Stack Overflow");
        return false;
    }
    else {
        a[++top] = x;
        System.out.println(x + " pushed into stack");
        return true;
    }
}
```

Pop()

```
int pop()
{
    if (top < 0) {
        System.out.println("Stack Underflow");
        return 0;
    }
    else {
        int x = a[top--];
        return x;
    }
}
```

Peek()

```
int peek()
{
    if (top < 0) {
        System.out.println("Stack Underflow");
        return 0;
    }
    else {
        int x = a[top];
        return x;
    }
}
```

isEmpty()

```
boolean isEmpty()
{
    return (top < 0);
}
```

Practices

- Evaluation of an Infix Expression that is Fully Parenthesized
- Input: (((2 * 5) - (1 * 2)) / (9 - 7))
- Output: 4
- Analysis: Five types of input characters
  - * Opening bracket
  - * Numbers
  - * Operators
  - * Closing bracket
  - * New line character

The end!