STACK

Stack is a fundamental data structure that adheres to the **Last In First Out (LIFO)** principle. Just like stacking plates, where the last plate you place is the first one you'll pick up, a stack organizes and manages data with an emphasis on the order of access. Stacks are a foundational concept in computer science, enabling efficient and organized handling of data and operations. Incorporating the LIFO principle, stacks prove their utility in diverse applications across programming and algorithmic problem-solving.

Key Methods of Stack

1. **push:** This method adds an element to the top of the stack.
2. **pop:** This method removes and returns the top element of the stack.
3. **peek:** This method returns the value of top element without removing it.
4. **count:** This method returns the number of elements in the stack.
5. **isFull**: This method checks if the stack has reached its maximum capacity (if applicable).
6. **isEmpty:** This method checks if the stack is empty.

Benefits of LIFO

The LIFO property ensures that the most recent item added to the stack is the first one to be removed. This property simplifies certain programming tasks and provides an intuitive way to manage data flow.

Pros

+ **LIFO Behavior**: The Last In First Out (LIFO) property of a stack matches well with certain real-world scenarios and programming tasks, making it intuitive to manage data in a chronological order.

+ **Simplicity**: Stacks are relatively simple to understand and implement. They involve only a few basic operations (push, pop, peek), which makes them accessible for beginners.

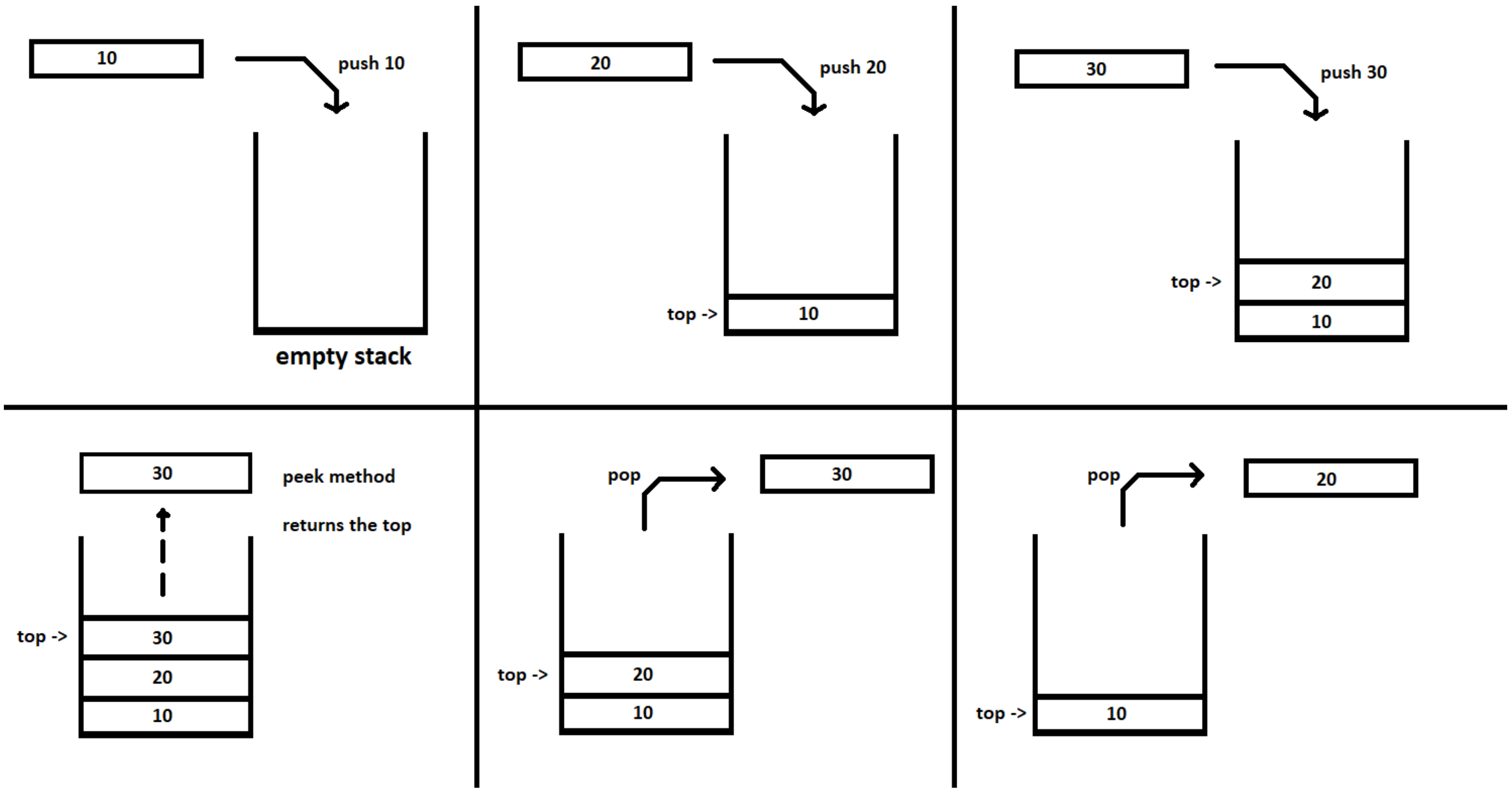
+ **Efficient Insertions and Removals**: Pushing and popping elements onto/from a stack are both constant-time operations, making them efficient for managing data that follows a specific order.

Cons

- Limited Access: Stacks are restrictive in terms of access. You can only interact with the top element of the stack, which may limit their usability for certain scenarios.

- Not Suitable for All Use Cases: While the LIFO behavior is beneficial for certain tasks, it's not suitable for all scenarios. For instance, when you need to access elements in a different order or when you require more complex data organization.

The visualization of the usage of key methods is provided below:


10
push 10
empty stack
20
push 20
top ->
10
30
push 30
top ->
20
10
30
peek method
returns the top
top ->
30
20
10
pop
30
top ->
20
10
pop
20
top ->
10