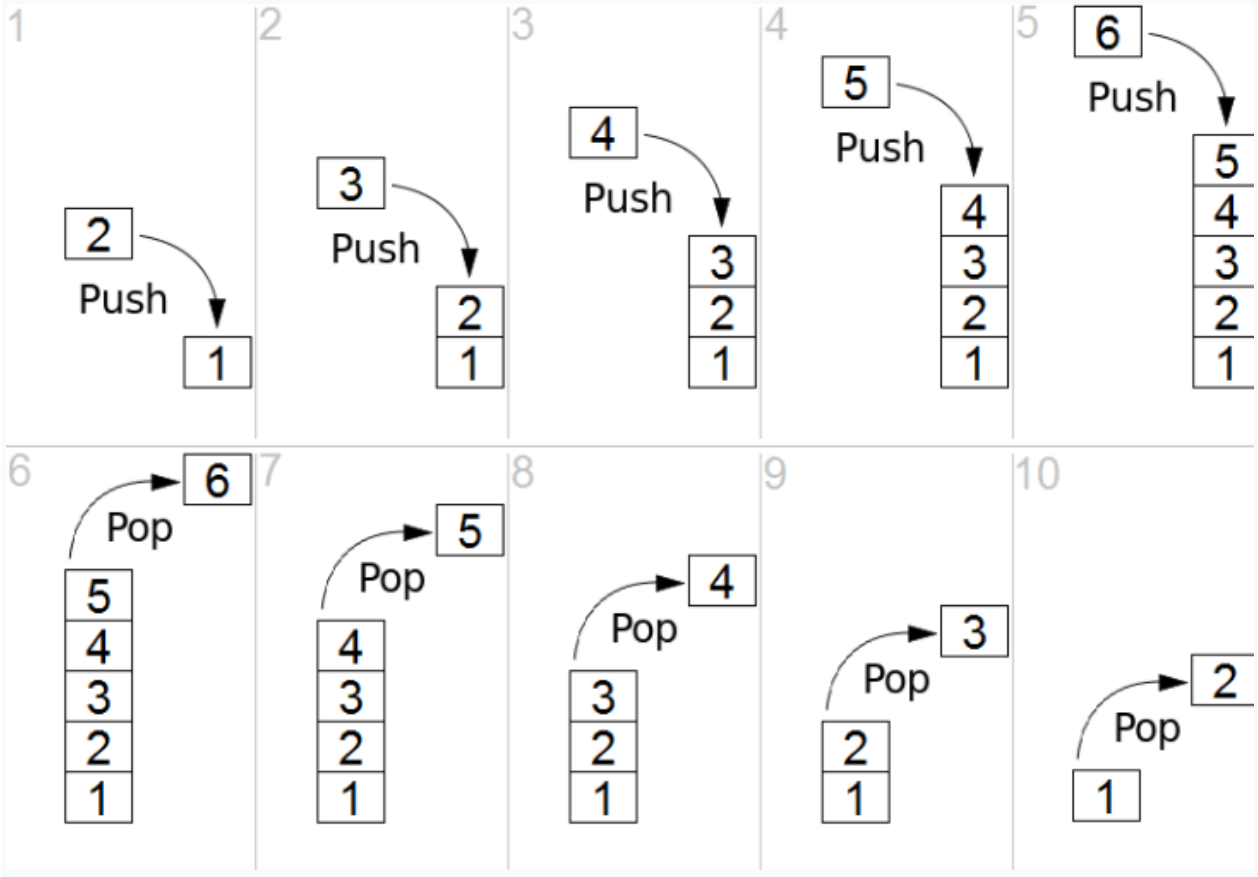
What Is a Stack ?

A Stack Is an Abstraction Data Structures Based On LIFO Principle (Last in First Out).
You can think of the stack data structure as Stack of plates.
you can only take a plate from the top of the stack, and you can only add a plate to the top of the stack.



LIFO Principle :

Stack allows all operations to occur at Only One Side.

LIFO Mean :

Last Element Inserted Must Be The First Element To Remove.

Stack Operations:

1. Push : Insert Element At The Top Of Stack.
2. Pop : Delete Element At The Top Of Stack.
3. Peek : Return Element At The Top Of Stack Without Deleting.

What Is Meant By Abstraction Data Structure ?

In General Abstraction Term Mean Define What ,, But Not Defined How .
Abstraction Data Structure is The Data Structure Which Described By Its Behavior Or Operations And Can Be Implemented By Different Data Structures.
Example : Stack Can Be Implemented By Array Or LinkedList.
But in Both Cases We Must Apply The LIFO Principle.
So What ?
→ Build Stack Based On LIFO Principle.
How ?
→ Stack Can Implemented By Array Or Linked List.

Stack Using Array

1.Stack Class Structure And Code :

```
4 references
public class Stack<T>
{
    T[] Array;
    int Top = -1;

    3 references
    public Stack(int arraySize = 10)
    {
        Array = new T[arraySize];
    }

    7 references
    public void Push(T newElement)...
    4 references
    public T Pop()...
    1 reference
    public T Peek()...
    1 reference
    private bool IsStackFull()...
    2 references
    private bool IsStackEmpty()...
}
```

2.Push Operation ：

2.1 Code :

```
7 references
public void Push(T newElement)
{
    if (IsStackFull())
    {
        throw new StackOverFlowException("Stack Is Full Exception");
    }

    Top++;
    Array[Top] = newElement;

}
```

```
1 reference
private bool IsStackFull()
{
    return Top >= Array.Length - 1;
}
```

2.2 Time Complexity = 1

3.Pop Operation :

3.1 Code :

```
4 references
public T Pop()
{
    if (IsStackEmpty())
    {
        throw new StackUnderFlowException("Stack Is Empty Exception");
    }

    T lastElement = Array[Top];
    Top--;
    return lastElement;
}

2 references
private bool IsStackEmpty()
{
    return Top == -1;
}
```

3.2 Time Complexity = 1

4.Peek Operation :

4.1 Code :

```
1 reference
public T Peek()
{
    if (IsStackEmpty())
    {
        throw new StackUnderFlowException("Stack Is Empty Exception");
    }

    return Array[Top];
}
```

4.2 Time Complexity = 1

5.Example :

```
static void Main(string[] args)
{
    try
    {
        Stack<int> Stack = new Stack<int>(3);
        //Stack.Pop();
        //Stack.Peek();
        Stack.Push(1);
        Stack.Push(2);
        Stack.Push(3);
        //Stack.Push(4);
        Console.WriteLine(Stack.Peek());
        Console.WriteLine(Stack.Pop());
        Console.WriteLine(Stack.Pop());
        Console.WriteLine(Stack.Pop());
    }
    catch(StackUnderFlowException ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (StackOverFlowException ex)
    {
        Console.WriteLine(ex.Message);
    }
```

Stack Using LinkedList

1.Stack Class Structure And Code :

```
1 reference
public class Stack<T>
{
    LinkedList<T> LinkedList = new LinkedList<T>();

    3 references
    public void Push(T newElement)...

    3 references
    public T Pop()...

    1 reference
    public T Peek()...

    2 references
    private bool IsStackEmpty()...
}
```

2.Push Operation ：

2.1 Code :

```
3 references
public void Push(T newElement)
{
    LinkedListNode<T> newNode = new LinkedListNode<T>(newElement);
    LinkedList.AddFirst(newNode);
}
```

2.2 Time Complexity = 1

3.Pop Operation :

3.1 Code :

```
3 references
public T Pop()
{
    if (IsStackEmpty())
    {
        throw new StackUnderFlowException("Stack Is Empty Exception");
    }

    T elementValue = LinkedList.First.Value;
    LinkedList.RemoveFirst();
    return elementValue;
}
```

```
2 references
private bool IsStackEmpty()
{
    return LinkedList.Count == 0;
}
```

3.2 Time Complexity = 1

4.Peek Operation ：

4.1 Code :

```
1 reference
public T Peek()
{
    if (IsStackEmpty())
    {
        throw new StackUnderFlowException("Stack Is Empty Exception");
    }
    return LinkedList.First.Value;
}
```

4.2 Time Complexity = 1

5.Example :

```
0 references
static void Main(string[] args)
{
    try
    {
        Stack<int> Stack = new Stack<int>();
        //Stack.Pop();
        //Stack.Peek();
        Stack.Push(1);
        Stack.Push(2);
        Stack.Push(3);
        //Stack.Push(4);
        Console.WriteLine(Stack.Peek());
        Console.WriteLine(Stack.Pop());
        Console.WriteLine(Stack.Pop());
        Console.WriteLine(Stack.Pop());
    }
    catch (StackUnderFlowException ex)
    {
        Console.WriteLine(ex.Message);
    }
    catch (StackOverFlowException ex)
    {
        Console.WriteLine(ex.Message);
    }
```