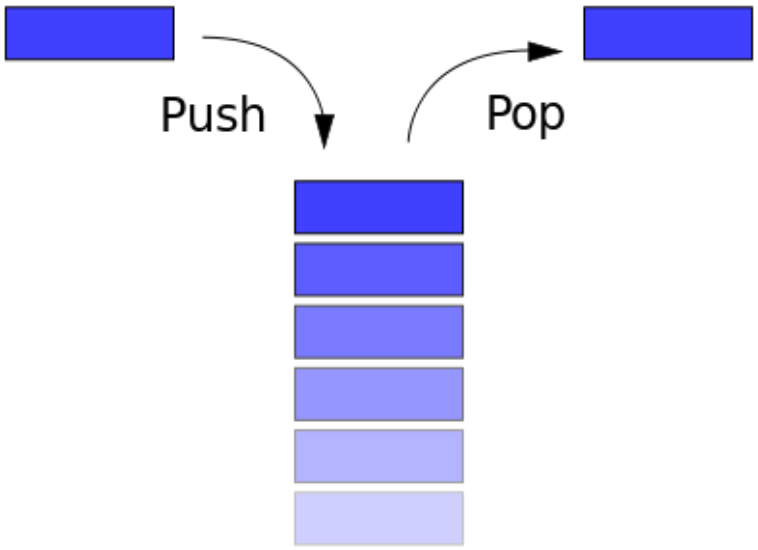
[0942] Stacks

Stacks provide storage for data items that need to be stored and retrieved in a "Last In, First Out" (LIFO) manner. Basically, if you need to store several items and you have process the newest items before you can process the older items, then you can use a stack. Stacks are ONLY useful for those kind of "LIFO" situations.

Stacks have two main methods - Push() and Pop(). Push() adds a data item to the stack. Pop() removes a data item from the "top" of the stack. (Some stacks also have "Peek()" which shows you the item on the top of the stack without removing it.)



In C#, the generic Stack collection class has several useful methods:

stack.Peek()
stack.Pop()
stack.Push()
stack.ToArray() - converts a stack into an array

In addition, Stack has a constructor that can take any other Collection as a parameter allowing you to easily convert a List or array into a Stack.

Here's an example of the Stack class in use:

```
using System;
using System.Collections.Generic;

namespace LWTech.Testing
{
   public class StackDemo
   {
      public static void Main()
      {
         var myStrings = new Stack<string>();
```

```
        // Push 10 strings on to the stack
        for (int i = 0; i < 10; i++)
            myStrings.Push("String #" + i);

        Console.WriteLine("There are " + myStrings.Count + " strings on the stack.");

        // Pop all the strings off of the stack
        while (myStrings.Count > 0)
            Console.WriteLine(myStrings.Pop());

        Console.WriteLine("There are now " + myStrings.Count + " strings on the stack.");
    }
  }
}
```

Output:

```
There are 10 strings on the stack.
String #9
String #8
String #7
String #6
String #5
String #4
String #3
String #2
String #1
String #0
There are now 0 strings on the stack.
```