Implementation of Stack in C#

Stack represents a last-in, first out collection of objects. It is used when you need a last-in, first-out access to items. When you add an item in the list, it is called **pushing** the item and when you remove it, it is called **popping** the item. This class comes under *System.Collections* namespace

Characteristics of Stack Class:

- ✓ The capacity of a Stack is the number of elements the Stack can hold. As elements are added to a Stack, the capacity is automatically increased as required through reallocation.
- ✓ If Count is less than the capacity of the stack, Push is an *O(1)* operation. If the capacity needs to be increased to accommodate the new element, Push becomes an *O(n)* operation, where *n* is Count. Pop is an *O(1)* operation.
- ✓ Stack accepts null as a valid value and allows duplicate elements.

Stack Constructors

Constructor	Description
Stack()	Initializes a new instance of the Stack class that is empty and has the default initial capacity**.**
Stack(ICollection)	Initializes a new instance of the Stack class that contains elements copied from the specified collection and has the same initial capacity as the number of elements copied.
Stack(Int32)	Initializes a new instance of the Stack class that is empty and has the specified initial capacity or the default initial capacity, whichever is greater.

Stack Properties

Property	Description
Count	Gets the number of elements contained in the Stack.
IsSynchronized	Gets a value indicating whether access to the Stack is synchronized (thread safe).
SyncRoot	Gets an object that can be used to synchronize access to the Stack.

Stack Methods

Methods	Usage
Clear()	Removes all objects from the Stack.
Clone()	Creates a shallow copy of the Stack.
Contains(Object)	Determines whether an element is in the Stack.
CopyTo(Array, Int32)	Copies the Stack to an existing one-dimensional Array, starting at the specified array index.
Equals(Object)	Determines whether the specified object is equal to the current object.
GetEnumerator()	Returns an IEnumerator for the Stack.
GetHashCode()	Serves as the default hash function
GetType()	Gets the Type of the current instance.
MemberwiseClone()	Creates a shallow copy of the current Object.

Peek()	Returns the object at the top of the Stack without removing it.
Pop()	Removes and returns the object at the top of the Stack.
Push(Object)	Inserts an object at the top of the Stack.
Synchronized(Stack)	Returns a synchronized (thread safe) wrapper for the Stack.
ToArray()	Copies the Stack to a new array.
ToString()	Returns a string that represents the current object.

Example Program

```
using System;
// C# code to create a Stack
using System.Collections;
class StackPush
{
      public static void Main()
    {
        // Creating a Stack
        Stack myStack = new Stack();

        // Inserting the elements into the Stack
        myStack.Push("CSHARP");
        myStack.Push("C++");
        myStack.Push("JAVA");
        myStack.Push("RUBY");
        myStack.Push("PHYTON");
        myStack.Push(null);
        myStack.Push(1234);
        myStack.Push(412.30);
        // Accessing the elements of myStack Using foreach loop
        foreach(var elem in myStack)
        {
            Console.WriteLine(elem);
        }

         //Accessing elements at the top
        Console.WriteLine("Element at the top is : " + myStack.Peek());

         // Checking if elements contains
        Console.WriteLine("Checking if PHYTON is available : " + myStack.Contains("PHYTON"));


        // Displaying the count of elements contained in the Stack
        Console.Write("Total number of elements in the Stack are : ");
        Console.WriteLine(myStack.Count);

         myStack.Pop();
         myStack.Pop();
        // After Pop method
        Console.WriteLine("Total elements present in "+
                     "myStack after Popping: {0}", myStack.Count);

         // Removing all elements from Stack
        myStack.Clear();
         // Total number of elemeents after removing all
        Console.Write("Total number of elements in the Stack after removing all are : ");
        Console.WriteLine(myStack.Count);
        Console.Read();
    }
}
```

Output

```
412.3
1234

PHYTON
RUBY
JAVA
C++
CSHARP
Element at the top is : 412.3
Checking if PHYTON is available : True
Total number of elements in the Stack are : 8
Total elements present in myStack after Popping: 6
Total number of elements in the Stack after removing all are : 0
```