The concept of (Stack).

اسم الطالب : احمد محمد عبد المنعم السبكي

الرقم القومي : 30003251702792

المستوي : اقل من 36

الشعبه : الرياضه البحته وعلوم الحاسب

اسم المقرر: (Data Structures)

البريدالاكاديمي: ahmed.mohamed9000@science.menofia.edu.eg

اشرف: ا.دْ/ عمرو مسعد

INTRODUCTION:

A stack is a method of creating a list so that the ultimate object introduced to the listing is likewise the primary one that would be removed. It can be illustrated as placing a few cups in a box which could receive best one on pinnacle of another (no adjoining cup). When it comes time to get one of these cups, you need to first access the top one; that is, the closing one that was added. In other words, at one time, the objects in the list appear within the reverse order they were brought.

A stack is an ordered collection of objects where the addition of latest items and the removal of existing items continually takes vicinity at the same quit. This end is commonly called the “top”, and the alternative end is known as the “base”.

Items which might be toward the bottom have been in the stack the longest. The most recently delivered item is usually on the pinnacle of the stack and thus will be eliminated first. The stack affords an ordering based on duration of time in the collection; the “age” of any given object will increase as you pass from top to base.

There are many examples of stacks in everyday situations. Consider a stack of plates on a table, in which it’s best viable to add or eliminate plates to or from the pinnacle. Or imagine a stack of books on a desk. The best e book whose cowl is visible is the only on top. To get admission to the others, we should first cast off those sitting on top of them.

One of the most useful features of stacks comes from the observation that the insertion order is the opposite of the removal order.

RESERCH PROJECT:

Stack Class:

Stack represents a last-in, first out collection of object.

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 limit of a Stack is the quantity of components the Stack can hold. As components are added to a Stack, the limit is consequently expanded as required through reallocation.

•If Count is not exactly the limit of the stack, Push is an O(1) activity. On the off chance that the limit should be expanded to oblige the new component, Push turns into an O(n) activity, where n is Count. Pop is an O(1) activity.

•Stack acknowledges invalid as a substantial worth and permits copy components

Constructors

1. Stack: Initializes a new instance of the Stack class that is empty and has the default initial capacity
2. 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.
3. 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.

EXAMPLE(1)

```
using System;

using System.Collections;

class GFG {

        public static void Main()

        {

                Stack Stack1 = new Stack();

                Stack1.Push("1st Element");

                Stack1.Push("2nd Element");

                Stack1.Push("3rd Element");

                Stack1.Push("4th Element");

                Stack1.Push("5th Element");

                Stack1.Push("6th Element");


                Console.Write("Total number of elements are ");

                Console.WriteLine(Stack1.Count);

                Console.WriteLine("Element at the top is " + Stack1.Peek());

                Console.WriteLine("Element at the top is " + Stack1.Peek());

                Console.Write("Total number of elements are ");


                Console.WriteLine(Stack1.Count);

        }

}
```

Output:

```
Total number of elements are 6

Element at the top is 6th Element

Element at the top is 6th Element

Total number of elements  are  6
```

Properties

1. Count**:** Gets the number of elements contained in the Stack.
2. IsSynchronized: Gets a value indicating whether access to the Stack is synchronized.
3. SyncRoot:Gets an object that can be used to synchronize access to the Stack.

EXAMPLE(2)

```
using System;
using System.Collections;

class GFG {
    public static void Main()
    {
        Stack Stack1 = new Stack();
        Stack1.Push("Ahmed");
        Stack1.Push("Mohamed");
        Stack1.Push("Khaled");
       Stack1.Push("Hasan");
        Stack1.Push("Omar");
        Stack1.Push("Aly");
        Console.Write("Total number of elements are ");

        Console.WriteLine(Stack1.Count);
    }
}
```

Output:

```
Total number of elements in are 6
```

Methods

1. Clear():Removes all objects from the Stack.
2. Clone(): Creates a shallow copy of the Stack
3. Contains(object): Determines whether an element is in the Stack
4. Copy To(Arry,Int32): Copies the Stack to an existing one-dimensional Array, starting at the specified array index
5. Equals(object): Determines whether the specified object is equal to the current object
6. GetEnumerator():Returns an IEnumerator for the Stack
7. GetHashCode():Serves as the default hash function

8. GetType():Gets the Type of the current instance
9. MemberwiseClone():Creates a shallow copy of the current Object
10. Peek():Returns the object at the top of the Stack without removing it
11. Pop():Removes and returns the object at the top of the Stack
12. push(object): Inserts an object at the top of the Stack
13. Synchronized(stack): Returns a synchronized (thread safe) wrapper for the Stack
14. ToArray():Copies the Stack to a new array
15. ToString():Returns a string that represents the current object

Notes:

1. Stack stores the values in LIFO (Last in First out) style. The element which is added last will be the element to come out first.
2. Use the Push() method to add elements into Stack.
3. The Pop() method returns and removes elements from the top of the Stack. Calling the Pop() method on the empty Stack will throw an exception.
4. The Peek() method always returns top most element in the Stack.

EXAMPLE(3)

```
using System;
using System.Collections ;
class STK {
    public static void Main()
    {
        Stack Stack1 = new Stack();

        Stack1.Push(9);
        Stack1.Push(8);
        Stack1.Push(7);
        Stack1.Push(6);
        Stack1.Push(5);

        Object[] arr1 = Stack1.ToArray();



        foreach(Object i in arr1)
        {
            Console.WriteLine(i);
        }
    }
}
```

Output:

```
5
6
7
8
9
```

EXAMPLE(4)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DemoApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   Stack st = new Stack();
   st.Push(1);
   st.Push(2);
   st.Push(3);
   st.Pop();
   foreach (Object obj in st)
   {
    Console.WriteLine(obj);
   }
    Console.ReadKey();
  }
 }
}
```

Output:

```
2

1
```

EXAMPLE(5)

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace DemoApplication
{
 class Program
 {
  static void Main(string[] args)
  {
   Stack st = new Stack();
   st.Push(1); st.Push(2); st.Push(3);

   foreach (Object obj in st)
   {
    Console.WriteLine(obj);
   }
    Console.WriteLine(); Console.WriteLine();
    Console.WriteLine("The number of elements in the stack= " +st.Count);
    Console.WriteLine("Does the stack contain the elements 3? "+st.Contains(3));
    Console.ReadKey();
  }
 }
}
```

Output:

```
3

2

1

The number of elements in the stack=3

Does the stack contain the elements 3?true
```

REFERENCES:

1. Data Structure, Dr/Amr Mausad
2. https://www.tutorialsteacher.com/csharp/csharp-stack
3. https://developerslogblog.wordpress.com/2019/07/23/how-to-implement-a-stack-in-c/
4. https://youtu.be/Sn6Qa26EeQc ,2015/9/5
5. https://www.tutorialspoint.com/csharp/pdf/csharp_stack.pdf
6. http://www.uomisan.edu.iq/library/admin/book/19226579694.pdf

   DATA STRUCTURES AND ALGORITHMS USING C#, (Chapter5)

   Dr/ MICHAEL MCMILLAN