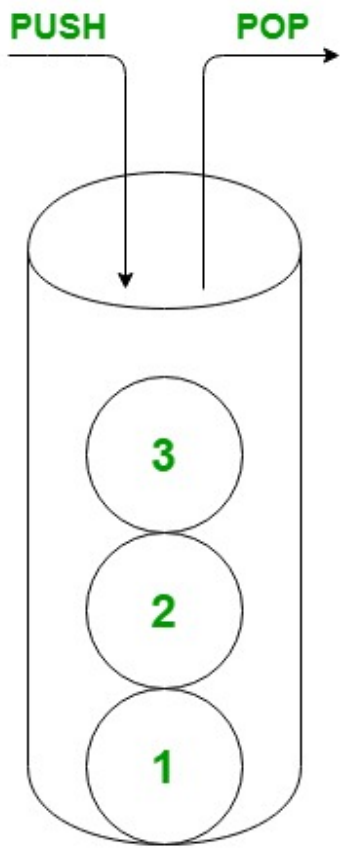
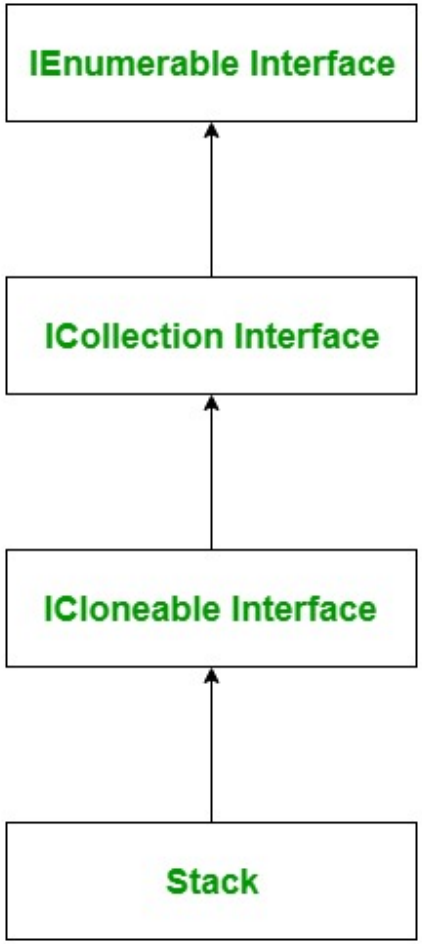
C# Stack with Examples

A Stack is used to represent a last-in, first o[illegible]ou need last-in, first-out access to items. It is of both generic and non-generic type of collecti[illegible]*System.Collections.Generic* namespace whereas non-generic stack is defined under *System.Collections* namespace, here we will discuss non-generic type stack. A stack is used to create a dynamic collection which grows, according to the need of your program. In a stack, you can store elements of the same type or different types.



The below diagram illustrates the Stack class hierarchy:

--->



--->

Important Points:

- The Stack class implements the *IEnumerable*, *ICollection*, and *ICloneable* interfaces.
- When you add an item in the list, it is called *pushing* the element.
- When you remove it, it is called *popping* the element.
- 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.
- In Stack, you are allowed to store duplicate elements.
- A Stack accepts null as a valid value for reference types.

How to create a Stack?

Stack class has *three* constructors which are used to create a stack which is as follows:

- **Stack():** This constructor is used to create an instance of the Stack class which is empty and having the default initial capacity.
- **Stack(ICollection):** This constructor is used to create an instance of the Stack class which contains elements copied from the specified collection, and has the same initial capacity as the number of elements copied.
- **Stack(Int32):** This constructor is used to create an instance of the Stack class which is empty and having specified initial capacity or the default initial capacity, whichever is greater.

Let's see how to create an stack using Stack() constructor:

Step 1: Include *System.Collections* namespace in your program with the help of using keyword.

```
using System.Collections;
```

Step 2: Create a stack using Stack class as shown below:

```
Stack stack_name = new Stack();
```

Step 3: If you want to add elements in your stack, then use *Push()* method to add elements in your stack. As shown in the below example.

Example:

```
// C# program to illustrate how to
// create a stack
using System;
using System.Collections;

class GFG {

    // Main Method
    static public void Main()
    {
```

```
        // Create a stack
        // Using Stack class
        Stack my_stack = new Stack();

        // Adding elements in the Stack
        // Using Push method
        my_stack.Push("Geeks");
        my_stack.Push("geeksforgeeks");
        my_stack.Push('G');
        my_stack.Push(null);
        my_stack.Push(1234);
        my_stack.Push(490.98);

        // Accessing the elements
        // of my_stack Stack
        // Using foreach loop
        foreach(var elem in my_stack)
        {
            Console.WriteLine(elem);
        }
    }
}
```

Output:

```
490.98
1234

G
geeksforgeeks
Geeks
```

How to remove elements from the Stack?

In Stack, you are allowed to remove elements from the stack. The Stack class provides two different methods to remove elements and the methods are:

- **Clear:** This method is used to remove all the objects from the stack.
- **Pop:** This method removes the beginning element of the stack.

Example:

```
// C# program to illustrate how to
// remove elements from the stack
using System;
using System.Collections;

class GFG {

    // Main Method
    static public void Main()
    {

        // Create a stack
        // Using Stack class
        Stack my_stack = new Stack();

        // Adding elements in the Stack
        // Using Push method
        my_stack.Push("Geeks");
```

```
            my_stack.Push("geeksforgeeks");
            my_stack.Push("geeks23");
            my_stack.Push("GeeksforGeeks");

            Console.WriteLine("Total elements present in"+
                        " my_stack: {0}", my_stack.Count);

            my_stack.Pop();

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


            // Remove all the elements
            // from the stack
            my_stack.Clear();

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

        }
    }
```

Output:

```
Total elements present in my_stack: 4
Total elements present in my_stack: 3
Total elements present in my_stack: 0
```

How to get topmost element of the Stack?

In Stack, you can easily find the topmost element of the stack by using the following methods provided by the Stack class:

- **Pop:** This method returns the object at the beginning of the stack with modification means this method remove the topmost element of the stack.
- **Peek:** This method returns the object at the beginning of the stack without removing it.

Example:

```
// C# program to illustrate how to
// get topmost elements of the stack
using System;
using System.Collections;

class GFG {

    // Main Method
    static public void Main()
    {

        // Create a stack
        // Using Stack class
        Stack my_stack = new Stack();

        // Adding elements in the Stack
        // Using Push method
        my_stack.Push("Geeks");
        my_stack.Push("geeksforgeeks");
        my_stack.Push("geeks23");
        my_stack.Push("GeeksforGeeks");

        Console.WriteLine("Total elements present in"+
                    " my_stack: {0}",my_stack.Count);

        // Obtain the topmost element
        // of my_stack Using Pop method
        Console.WriteLine("Topmost element of my_stack"
                        + " is: {0}",my_stack.Pop());

        Console.WriteLine("Total elements present in"+
                    " my_stack: {0}", my_stack.Count);

        // Obtain the topmost element
        // of my_stack Using Peek method
        Console.WriteLine("Topmost element of my_stack "+
                            "is: {0}",my_stack.Peek());
```

```
            Console.WriteLine("Total elements present "+
                    "in my_stack: {0}",my_stack.Count);

        }
    }
```

Output:

```
Total elements present in my_stack: 4
Topmost element of my_stack is: GeeksforGeeks
Total elements present in my_stack: 3
Topmost element of my_stack is: geeks23
Total elements present in my_stack: 3
```

How to check the availability of elements in the stack?

In a stack, you can check whether the given element is present or not using Contain() method. Or in other words, if you want to search an element in the given stack use Contains() method. This method returns true if the element present in the stack. Otherwise, return false.

Example:

```
// C# program to illustrate how
// to check element present in
// the stack or not
using System;
using System.Collections;

class GFG {

    // Main  Method
    static public void Main()
    {

        // Create a stack
        // Using Stack class
```

```
            Stack my_stack = new Stack();

            // Adding elements in the Stack
            // Using Push method
            my_stack.Push("Geeks");
            my_stack.Push("geeksforgeeks");
            my_stack.Push("geeks23");
            my_stack.Push("GeeksforGeeks");

            // Checking if the element is
            // present in the Stack or not
            if (my_stack.Contains("GeeksforGeeks") == true)
            {
                Console.WriteLine("Element is found...!!");
            }

            else
            {
                Console.WriteLine("Element is not found...!!");
            }
        }
    }
```

Output:

```
Element is found...!!
```

Generic Queue Vs Non-Generic Stack

GENERIC STACK	NON-GENERIC STACK
Generic stack is defined under System.Collections.Generic namespace.	Non-Generic stack is defined under System.Collections namespace.
Generic stack can only store same type of elements.	Non-Generic stack can store same type or different types of elements.

There is a need to define the type of the elements in the stack.	There is no need to define the type of the elements in the stack.
It is type- safe.	It is not type-safe.

Recommended Posts:

Stack.ToString() Method in C# with examples

C# | Stack<T>.TrimExcess Method with Examples

C# | How to create a Stack

Implementing Stack in C#

C# | Stack Class

Stack.Pop() Method in C#

Stack.Contains() Method in C#

C# | Create a Stack from a collection

Stack.Peek Method in C#

C# | Convert Stack to array

Stack.Clone() Method in C#

C# | Check if a Stack contains an element

C# | Remove all objects from the Stack

Stack.Synchronized() Method in C#

C# | Copy the Stack to an Array

ankita_saini
Check out this Author's contributed articles.

If you like GeeksforGeeks and would like to contribute, you can also write an article using contribute.geeksforgeeks.org or mail your article to contribute@geeksforgeeks.org. See your article appearing on the GeeksforGeeks main page and help other Geeks.

Please Improve this article if you find anything incorrect by clicking on the "Improve Article" button below.

Article Tags : C# CSharp-Collections-Namespace CSharp-Collections-Stack CSharp-Stack-Class

2

0

To-do Done

No votes yet.

Feedback/ Suggest Improvement Add Notes Improve Article

Please write to us at contribute@geeksforgeeks.org to report any issue with the above content.

ng code in comment? Please use ide.geeksforgeeks.org, generate link and share the link here.

Load Comments

A computer science portal for geeks

5th Floor, A-118,
Sector-136, Noida, Uttar Pradesh - 201305
feedback@geeksforgeeks.org

COMPANY

About Us
Careers
Privacy Policy
Contact Us

LEARN

Algorithms
Data Structures
Languages
CS Subjects
Video Tutorials

PRACTICE

Courses
Company-wise
Topic-wise
How to begin?

CONTRIBUTE

Write an Article
Write Interview Experience
Internships
Videos