System.Collections.Stack Class

[ILAsm]
```
.class public serializable beforefieldinit Stack extends System.Object
implements System.Collections.ICollection, System.Collections.IEnumerable,
System.ICloneable
```

[C#]
```
public class Stack: ICloneable, System.Collections.ICollection
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 4.0.0.0
- *Attributes:*
    - CLSCompliantAttribute(true)

Implements:

- **System.Collections.ICollection**
- **System.ICloneable**

Summary

Represents a simple last-in-first-out (LIFO) non-generic collection of objects.

Inherits From: System.Object

Library: BCL

Description

For the generic version of this collection, see ``System.Collections.Generic.Stack`1<T>``.

`System.Collections.Stack` is implemented as a circular buffer.

The capacity of a `System.Collections.Stack` is the number of elements the `System.Collections.Stack` can hold. As elements are added to a `System.Collections.Stack`, the capacity is automatically increased as required through reallocation.

If `System.Collections.Stack.Count` is less than the capacity of the stack, `System.Collections.Stack.Push` is an O(1) operation. If the capacity needs to be increased to accommodate the new element, `System.Collections.Stack.Push` becomes an O(*n*) operation, where *n* is `System.Collections.Stack.Count`. `System.Collections.Stack.Pop` is an O(1) operation.

`System.Collections.Stack` accepts `null` as a valid value and allows duplicate elements.

Stack() Constructor

```
[ILAsm]
.method public hidebysig specialname rtspecialname instance void .ctor()
cil managed

[C#]
public Stack ()
```

Summary

Initializes a new instance of the `System.Collections.Stack` class that is empty and has the default initial capacity.

Description

This constructor is an O(1) operation.

Stack(System.Collections.ICollection) Constructor

```
[ILAsm]
.method public hidebysig specialname rtspecialname instance void
.ctor(class System.Collections.ICollection col) cil managed

[C#]
public Stack (System.Collections.ICollection col)
```

Summary

Initializes a new instance of the `System.Collections.Stack` class that contains elements copied from the specified collection and has the same initial capacity as the number of elements copied.

Parameters

Parameter	Description
col	The `System.Collections.ICollection` to copy elements from.

Description

The elements are copied onto the `System.Collections.Stack` in the same order they are read by the `System.Collections.IEnumerator` of the `System.Collections.ICollection`.

This constructor is an O(*n*) operation, where *n* is the number of elements in *col*.

Exceptions

Exception	Condition
System.ArgumentNullException	*col* is `null`.

Stack(System.Int32) Constructor

[ILAsm]
```
.method public hidebysig specialname rtspecialname instance void
.ctor(int32 initialCapacity) cil managed
```

[C#]
```
public Stack (int initialCapacity)
```

Summary

Initializes a new instance of the `System.Collections.Stack` class that is empty and has the specified initial capacity or the default initial capacity, whichever is greater.

Parameters

Parameter	Description
initialCapacity	The initial number of elements that the `System.Collections.Stack` can contain.

Description

If the size of the collection can be estimated, specifying the initial capacity eliminates the need to perform a number of resizing operations while adding elements to the `System.Collections.Stack`.

This constructor is an O(*n*) operation, where *n* is *initialCapacity*.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*initialCapacity* is less than zero.

Stack.Clear() Method

[ILAsm]
```
.method public hidebysig newslot virtual instance void Clear() cil managed
```

[C#]
```
public virtual void Clear ()
```

Summary

Removes all objects from the `System.Collections.Stack`.

Description

`System.Collections.Stack.Count` is set to zero, and references to other objects from elements of the collection are also released.

This method is an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

[*Note:* This method may be an O(n) operation, where n is `System.Collections.Stack.Count`

]

Stack.Clone() Method

```
[ILAsm]
.method public hidebysig newslot virtual instance object Clone() cil
managed

[C#]
public virtual object Clone ()
```

Summary

Creates a shallow copy of the `System.Collections.Stack`.

Return Value

A shallow copy of the `System.Collections.Stack`.

Description

A shallow copy of a collection copies only the elements of the collection, whether they are reference types or value types, but it does not copy the objects that the references refer to. The references in the new collection point to the same objects that the references in the original collection point to.

In contrast, a deep copy of a collection copies the elements and everything directly or indirectly referenced by the elements.

This method is an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

Stack.Contains(System.Object) Method

[ILAsm]
```
.method public hidebysig newslot virtual instance bool Contains(object obj) cil managed
```

[C#]
```
public virtual bool Contains (object obj)
```

Summary

Determines whether an element is in the `System.Collections.Stack`.

Parameters

Parameter	Description
obj	The `System.Object` to locate in the `System.Collections.Stack`. The value can be `null`.

Return Value

`true`, if *obj* is found in the `System.Collections.Stack`; otherwise, `false`.

Description

This method determines equality by calling `System.Object.Equals`.

This method performs a linear search; therefore, this method is an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

This method uses the collection’s objects’ `System.Object.Equals` and `System.IComparable.CompareTo` methods on *obj* to determine whether *item* exists.

Stack.CopyTo(System.Array, System.Int32) Method

[ILAsm]
```
.method public hidebysig newslot virtual instance void CopyTo(class
System.Array array, int32 index) cil managed
```

[C#]
```
public virtual void CopyTo (Array array, int index)
```

Summary

Copies the `System.Collections.Stack` to an existing one-dimensional `System.Array`, starting at the specified array index.

Parameters

Parameter	Description
array	The one-dimensional `System.Array` that is the destination of the elements copied from `System.Collections.Stack`. The `System.Array` must have zero-based indexing.
index	The zero-based index in *array* at which copying begins.

Description

The elements are copied onto the array in last-in-first-out (LIFO) order, similar to the order of the elements returned by a succession of calls to `System.Collections.Stack.Pop`.

This method is an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

Exceptions

Exception	Condition
System.ArgumentNullException	*array* is `null`.
System.ArgumentOutOfRangeException	*index* is less than zero.
System.ArgumentException	*array* is multidimensional. -or-

	The number of elements in the source `System.Collections.Stack` is greater than the available space from *index* to the end of the destination *array*.
System.InvalidCastException	The type of the source `System.Collections.Stack` cannot be cast automatically to the type of the destination *array*.

Stack.GetEnumerator() Method

[ILAsm]
```
.method public hidebysig newslot virtual instance class
System.Collections.IEnumerator GetEnumerator() cil managed
```

[C#]
```
public virtual System.Collections.IEnumerator GetEnumerator ()
```

Summary

Returns an `System.Collections.IEnumerator` for the `System.Collections.Stack`.

Return Value

An `System.Collections.IEnumerator` for the `System.Collections.Stack`.

Description

Usage

For a detailed description regarding the use of an enumerator, see `System.Collections.Generic.IEnumerator<T>`.

This method is an O(1) operation.

Stack.Peek() Method

```
[ILAsm]
.method public hidebysig newslot virtual instance object Peek() cil
managed

[C#]
public virtual object Peek ()
```

Summary

Returns the object at the top of the `System.Collections.Stack` without removing it.

Return Value

The `System.Object` at the top of the `System.Collections.Stack`.

Description

This method is similar to the `System.Collections.Stack.Pop` method, but `System.Collections.Stack.Peek` does not modify the `System.Collections.Stack`.

`null` can be pushed onto the `System.Collections.Stack` as a placeholder, if needed. To distinguish between a null value and the end of the stack, check the `System.Collections.Stack.Count` property or catch the `System.InvalidOperationException`, which is thrown when the `System.Collections.Stack` is empty.

This method is an O(1) operation.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Collections.Stack` is empty.

Stack.Pop() Method

```
[ILAsm]
.method public hidebysig newslot virtual instance object Pop() cil managed

[C#]
public virtual object Pop ()
```

Summary

Removes and returns the object at the top of the `System.Collections.Stack`.

Return Value

The `System.Object` removed from the top of the `System.Collections.Stack`.

Description

This method is similar to the `System.Collections.Stack.Peek` method, but `System.Collections.Stack.Peek` does not modify the `System.Collections.Stack`.

`null` can be pushed onto the `System.Collections.Stack` as a placeholder, if needed. To distinguish between a null value and the end of the stack, check the `System.Collections.Stack.Count` property or catch the `System.InvalidOperationException`, which is thrown when the `System.Collections.Stack` is empty.

`System.Collections.Stack` is implemented as a circular buffer. This method is an O(1) operation.

Exceptions

Exception	Condition
System.InvalidOperationException	The `System.Collections.Stack` is empty.

Stack.Push(System.Object) Method

```
[ILAsm]
.method public hidebysig newslot virtual instance void Push(object obj)
cil managed

[C#]
public virtual void Push (object obj)
```

Summary

Inserts an object at the top of the `System.Collections.Stack`.

Parameters

Parameter	Description
obj	The `System.Object` to push onto the `System.Collections.Stack`. The value can be `null`.

Description

`System.Collections.Stack` is implemented as a circular buffer.

If `System.Collections.Stack.Count` already equals the capacity, the capacity of the `System.Collections.Stack` is increased by automatically reallocating the internal array, and the existing elements are copied to the new array before the new element is added.

`null` can be pushed onto the `System.Collections.Stack` as a placeholder, if needed. It occupies a slot in the stack and is treated like any object.

If `System.Collections.Stack.Count` is less than the capacity of the stack, `System.Collections.Stack.Push` is an O(1) operation. If the capacity needs to be increased to accommodate the new element, `System.Collections.Stack.Push` becomes an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

Stack.Synchronized(System.Collections.Stack) Method

```
[ILAsm]
.method public hidebysig static class System.Collections.Stack
Synchronized(class System.Collections.Stack stack) cil managed

[C#]
public static System.Collections.Stack Synchronized
(System.Collections.Stack stack)
```

Summary

Returns a synchronized (thread safe) wrapper for the `System.Collections.Stack`.

Parameters

Parameter	Description
stack	The `System.Collections.Stack` to synchronize.

Return Value

A synchronized wrapper around the `System.Collections.Stack`.

Description

To guarantee the thread safety of the `System.Collections.Stack`, all operations must be done through this wrapper.

[*Note:* The returned stack contains a reference to the original stack.

]

Enumerating through a collection is intrinsically not a thread-safe procedure. Even when a collection is synchronized, other threads can still modify the collection, which causes the enumerator to throw an exception. To guarantee thread safety during enumeration, you can either lock the collection during the entire enumeration or catch the exceptions resulting from changes made by other threads.

This method is an O(1) operation.

Exceptions

Exception	Condition

System.ArgumentNullException	*stack* is `null`.

Stack.ToArray() Method

```
[ILAsm]
.method public hidebysig newslot virtual instance object[] ToArray() cil
managed

[C#]
public virtual object[] ToArray ()
```

Summary

Copies the `System.Collections.Stack` to a new array.

Return Value

A new array containing copies of the elements of the `System.Collections.Stack`.

Description

The elements are copied onto the array in last-in-first-out (LIFO) order, similar to the order of the elements returned by a succession of calls to `System.Collections.Stack.Pop`.

This method is an O(*n*) operation, where *n* is `System.Collections.Stack.Count`.

Stack.Count Property

```
[ILAsm]
.property instance int32 Count

[C#]
public virtual int Count { get; }
```

Summary

Gets the number of elements contained in the `System.Collections.Stack`.

Property Value

The number of elements contained in the `System.Collections.Stack`.

Description

The capacity is the number of elements that the `System.Collections.Stack` can store. `System.Collections.Stack.Count` is the number of elements that are actually in the `System.Collections.Stack`.

The capacity is always greater than or equal to `System.Collections.Stack.Count`. If `System.Collections.Stack.Count` exceeds the capacity while adding elements, the capacity is automatically increased by reallocating the internal array before copying the old elements and adding the new elements.

Retrieving the value of this property is an O(1) operation.

Stack.IsSynchronized Property

[ILAsm]
```
.property instance bool IsSynchronized
```

[C#]
```
public virtual bool IsSynchronized { get; }
```

Summary

Gets a value indicating whether access to the `System.Collections.Stack` is synchronized (thread safe).

Property Value

`true`, if access to the `System.Collections.Stack` is synchronized (thread safe); otherwise, `false`. The default is `false`.

Description

To guarantee the thread safety of the `System.Collections.Stack`, all operations must be done through the wrapper returned by the `System.Collections.Stack.Synchronized` method.

Enumerating through a collection is intrinsically not a thread-safe procedure. Even when a collection is synchronized, other threads can still modify the collection, which causes the enumerator to throw an exception. To guarantee thread safety during enumeration, you can either lock the collection during the entire enumeration or catch the exceptions resulting from changes made by other threads.

Retrieving the value of this property is an O(1) operation.

Stack.SyncRoot Property

```
[ILAsm]
.property instance object SyncRoot

[C#]
public virtual object SyncRoot { get; }
```

Summary

Gets an object that can be used to synchronize access to the `System.Collections.Stack`.

Property Value

An `System.Object` that can be used to synchronize access to the `System.Collections.Stack`.

Description

To create a synchronized version of the `System.Collections.Stack`, use the `System.Collections.Stack.Synchronized` method. However, derived classes can provide their own synchronized version of the `System.Collections.Stack` using the `System.Collections.Stack.SyncRoot` property. The synchronizing code must perform operations on the `System.Collections.Stack.SyncRoot` of the `System.Collections.Stack`, not directly on the `System.Collections.Stack`. This ensures proper operation of collections that are derived from other objects. Specifically, it maintains proper synchronization with other threads that might be simultaneously modifying the `System.Collections.Stack` object.

Enumerating through a collection is intrinsically not a thread-safe procedure. Even when a collection is synchronized, other threads can still modify the collection, which causes the enumerator to throw an exception. To guarantee thread safety during enumeration, you can either lock the collection, using the `System.Collections.Stack.SyncRoot` object, during the entire enumeration or catch the exceptions resulting from changes made by other threads.

Retrieving the value of this property is an O(1) operation.