

System.Collections.Generic.Stack<T> Class

```
[ILAsm]
.class public serializable beforefieldinit Stack`1<T> extends
System.Object implements System.Collections.Generic.IEnumerable`1<!0>,
System.Collections.ICollection, System.Collections.IEnumerable

[C#]
public class Stack<T>: System.Collections.Generic.IEnumerable<T>,
System.Collections.ICollection
```

Assembly Info:

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

Implements:

- **System.Collections.Generic.IEnumerable<T>**
- **System.Collections.ICollection**

Summary

Represents a variable size last-in-first-out (LIFO) collection of instances of the same arbitrary type.

Inherits From: System.Object

Library: BCL

Description

System.Collections.Generic.Stack<T> is implemented as an array.

The capacity of a System.Collections.Generic.Stack<T> is the number of elements the System.Collections.Generic.Stack<T> can hold. As elements are added to a System.Collections.Generic.Stack<T>, the capacity is automatically increased as required by reallocating the internal array. The capacity can be decreased by calling System.Collections.Generic.Stack<T>.TrimExcess.

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

1 `System.Collections.Generic.Stack<T>` accepts null as a valid value for reference
2 types and allows duplicate elements.

3

Stack<T>() Constructor

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

[C#]
public Stack ()
```

Summary

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

Description

The capacity of a `System.Collections.Generic.Stack<T>` is the number of elements that the `System.Collections.Generic.Stack<T>` can hold. As elements are added to a `System.Collections.Generic.Stack<T>`, the capacity is automatically increased as required by reallocating the internal array.

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.Generic.Stack<T>`.

The capacity can be decreased by calling `System.Collections.Generic.Stack<T>.TrimExcess`.

This constructor is an $O(1)$ operation.

Stack<T> (System.Collections.Generic.IEnumerable<T>) Constructor

```
[ILAsm]
.method public hidebysig specialname rtspecialname instance void
.ctor(class System.Collections.Generic.IEnumerable`1<T> collection) cil
managed

[C#]
public Stack (System.Collections.Generic.IEnumerable<T> collection)
```

Summary

Initializes a new instance of the `System.Collections.Generic.Stack<T>` class that contains elements copied from the specified collection and has sufficient capacity to accommodate the number of elements copied.

Parameters

Parameter	Description
<i>collection</i>	The collection to copy elements from.

Description

The capacity of a `System.Collections.Generic.Stack<T>` is the number of elements that the `System.Collections.Generic.Stack<T>` can hold. As elements are added to a `System.Collections.Generic.Stack<T>`, the capacity is automatically increased as required by reallocating the internal array.

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.Generic.Stack<T>`.

The capacity can be decreased by calling `System.Collections.Generic.Stack<T>.TrimExcess`.

The elements are copied onto the `System.Collections.Generic.Stack<T>` in the same order they are read by the `System.Collections.Generic.IEnumerator<T>` of the collection.

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

Exceptions

Exception	Condition
-----------	-----------

System.ArgumentNullException

collection is null.

1

2

Stack<T> (System.Int32) Constructor

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

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

Summary

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

Parameters

Parameter	Description
<i>capacity</i>	The initial number of elements that the <code>System.Collections.Generic.Stack<T></code> can contain.

Description

The capacity of a `System.Collections.Generic.Stack<T>` is the number of elements that the `System.Collections.Generic.Stack<T>` can hold. As elements are added to a `System.Collections.Generic.Stack<T>`, the capacity is automatically increased as required by reallocating the internal array.

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.Generic.Stack<T>`.

The capacity can be decreased by calling `System.Collections.Generic.Stack<T>.TrimExcess`.

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

Exceptions

Exception	Condition
<code>System.ArgumentOutOfRangeException</code>	<i>capacity</i> is less than zero.

Stack<T>.Clear() Method

```
[ILAsm]  
.method public hidebysig instance void Clear() cil managed  
  
[C#]  
public void Clear ()
```

Summary

Removes all objects from the `System.Collections.Generic.Stack<T>`.

Description

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

The capacity remains unchanged. To reset the capacity of the `System.Collections.Generic.Stack<T>`, call `System.Collections.Generic.Stack<T>.TrimExcess`. Trimming an empty `System.Collections.Generic.Stack<T>` sets the capacity of the `System.Collections.Generic.Stack<T>` to the default capacity.

This method is an $O(n)$ operation, where n is `System.Collections.Generic.Stack<T>.Count`.

Stack<T>.Contains(T) Method

```
[ILAsm]
.method public hidebysig instance bool Contains(!0 item) cil managed

[C#]
public bool Contains (T item)
```

Summary

Determines whether an element is in the `System.Collections.Generic.Stack<T>`.

Parameters

Parameter	Description
<i>item</i>	The object to locate in the <code>System.Collections.Generic.Stack<T></code> . The value can be null for reference types.

Return Value

true if *item* is found in the `System.Collections.Generic.Stack<T>`; otherwise, false.

Description

This method determines equality using the default equality comparer `System.Collections.Generic.EqualityComparer<T>.Default` for *T*, the type of values in the list.

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

Stack<T>.CopyTo(T[], System.Int32) Method

```
[ILAsm]
.method public hidebysig instance void CopyTo(!0[] array, int32
arrayIndex) cil managed

[C#]
public void CopyTo (T[] array, int arrayIndex)
```

Summary

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

Parameters

Parameter	Description
<i>array</i>	The one-dimensional <code>System.Array</code> that is the destination of the elements copied from <code>System.Collections.Generic.Stack<T></code> . The <code>System.Array</code> must have zero-based indexing.
<i>arrayIndex</i>	The zero-based index in <i>array</i> 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.Generic.Stack<T>.Pop`.

This method is an $O(n)$ operation, where n is `System.Collections.Generic.Stack<T>.Count`.

Exceptions

Exception	Condition
System.ArgumentNullException	<i>array</i> is null.
System.ArgumentOutOfRangeException	<i>arrayIndex</i> is less than zero.
System.ArgumentException	The number of elements in the source <code>System.Collections.Generic.Stack<T></code> is greater than the available space from <i>arrayIndex</i> to the end of the destination <i>array</i> .

1

2

Stack<T>.GetEnumerator() Method

```
[ILAsm]
.method public hidebysig instance valuetype
System.Collections.Generic.Stack`1/Enumerator<!0> GetEnumerator() cil
managed

[C#]
public System.Collections.Generic.Stack<T>.Enumerator GetEnumerator ()
```

Summary

Returns an enumerator for the `System.Collections.Generic.Stack`1<T>`.

Return Value

An `System.Collections.Generic.Stack`1<T>.Enumerator` for the `System.Collections.Generic.Stack`1<T>`.

Description

Usage

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

Default implementations of collections in `System.Collections.Generic` are not synchronized.

This method is an $O(1)$ operation.

Stack<T>.Peek() Method

```
[ILAsm]
.method public hidebysig instance !0 Peek() cil managed

[C#]
public T Peek ()
```

Summary

Returns the object at the top of the `System.Collections.Generic.Stack<T>` without removing it.

Return Value

The object at the top of the `System.Collections.Generic.Stack<T>`.

Description

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

If type *T* is a reference type, null can be pushed onto the `System.Collections.Generic.Stack<T>` as a placeholder, if needed.

This method is an O(1) operation.

Exceptions

Exception	Condition
<code>System.InvalidOperationException</code>	The <code>System.Collections.Generic.Stack<T></code> is empty.

Stack<T>.Pop() Method

```
[ILAsm]
.method public hidebysig instance !0 Pop() cil managed

[C#]
public T Pop ()
```

Summary

Removes and returns the object at the top of the `System.Collections.Generic.Stack<T>`.

Return Value

The object removed from the top of the `System.Collections.Generic.Stack<T>`.

Description

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

If type *T* is a reference type, null can be pushed onto the `System.Collections.Generic.Stack<T>` as a placeholder, if needed.

`System.Collections.Generic.Stack<T>` is implemented as an array. This method is an O(1) operation.

Exceptions

Exception	Condition
System.InvalidOperationException	The <code>System.Collections.Generic.Stack<T></code> is empty.

Stack<T>.Push(T) Method

```
[ILAsm]
.method public hidebysig instance void Push(!0 item) cil managed

[C#]
public void Push (T item)
```

Summary

Inserts an object at the top of the `System.Collections.Generic.Stack<T>`.

Parameters

Parameter	Description
<i>item</i>	The object to push onto the <code>System.Collections.Generic.Stack<T></code> . The value can be null for reference types.

Description

`System.Collections.Generic.Stack<T>` is implemented as an array.

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

If type *T* is a reference type, null can be pushed onto the `System.Collections.Generic.Stack<T>` as a placeholder, if needed. It occupies a slot in the stack and is treated like any object.

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

Stack<T>.System.Collections.Generic.IEnumerable<T>.GetEnumerator() Method

```
[ILAsm]
.method private hidebysig newslot virtual final instance class
System.Collections.Generic.IEnumerator`1<T>
System.Collections.Generic.IEnumerable<T>.GetEnumerator() cil managed

[C#]
System.Collections.Generic.IEnumerator<T> IEnumerable<T>.GetEnumerator ()
```

Summary

Returns an enumerator that iterates through the collection.

Return Value

An `System.Collections.Generic.IEnumerator`1<T>` that can be used to iterate through the collection.

Description

Usage

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

Default implementations of collections in `System.Collections.Generic` are not synchronized.

This method is an O(1) operation.

Stack<T>.System.Collections.ICollection.CopyTo(System.Array, System.Int32) Method

```
[ILAsm]  
.method private hidebysig newslot virtual final instance void  
System.Collections.ICollection.CopyTo(class System.Array array, int32  
arrayIndex) cil managed
```

```
[C#]  
void ICollection.CopyTo (Array array, int arrayIndex)
```

Summary

Copies the elements of the `System.Collections.ICollection` to an `System.Array`, starting at a particular `System.Array` index.

Parameters

Parameter	Description
<i>array</i>	The one-dimensional <code>System.Array</code> that is the destination of the elements copied from <code>System.Collections.ICollection</code> . The <code>System.Array</code> must have zero-based indexing.
<i>arrayIndex</i>	The zero-based index in <i>array</i> at which copying begins.

Description

[*Note:* If the type of the source `System.Collections.ICollection` cannot be cast automatically to the type of the destination *array*, the non-generic implementations of `System.Collections.ICollection.CopyTo` throw `System.InvalidCastException`, whereas the generic implementations throw `System.ArgumentException`.

]

This method is an $O(n)$ operation, where n is `System.Collections.Generic.Stack<T>.Count`.

Exceptions

Exception	Condition
<code>System.ArgumentNullException</code>	<i>array</i> is null.

System.ArgumentOutOfRangeException	<i>arrayIndex</i> is less than zero.
System.ArgumentException	<p><i>array</i> is multidimensional.</p> <p>-or-</p> <p><i>array</i> does not have zero-based indexing.</p> <p>-or-</p> <p>The number of elements in the source <code>System.Collections.ICollection</code> is greater than the available space from <i>arrayIndex</i> to the end of the destination <i>array</i>.</p> <p>-or-</p> <p>The type of the source <code>System.Collections.ICollection</code> cannot be cast automatically to the type of the destination <i>array</i>.</p>

1

2

Stack<T>.System.Collections.IEnumerable.GetEnumerator() Method

```
[ILAsm]
.method private hidebysig newslot virtual final instance class
System.Collections.Generic.IEnumerator`1<T>
System.Collections.Generic.IEnumerable<T>.GetEnumerator() cil managed

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

Summary

Returns an enumerator that iterates through a collection.

Return Value

An `System.Collections.IEnumerator` that can be used to iterate through the collection.

Description

Usage

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

Default implementations of collections in `System.Collections.Generic` are not synchronized.

This method is an O(1) operation.

Stack<T>.ToArray() Method

```
[ILAsm]
.method public hidebysig instance !0[] ToArray() cil managed

[C#]
public T[] ToArray ()
```

Summary

Copies the `System.Collections.Generic.Stack<T>` to a new array.

Return Value

A new array containing copies of the elements of the `System.Collections.Generic.Stack<T>`.

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.Generic.Stack<T>.Pop`.

This method is an $O(n)$ operation, where n is `System.Collections.Generic.Stack<T>.Count`.

Stack<T>.TrimExcess() Method

```
[ILAsm]  
.method public hidebysig instance void TrimExcess() cil managed  
  
[C#]  
public void TrimExcess ()
```

Summary

Sets the capacity to the actual number of elements in the `System.Collections.Generic.Stack<T>`, if that number is less than 90 percent of current capacity.

Description

This method can be used to minimize a collection's memory overhead if no new elements will be added to the collection. The cost of reallocating and copying a large `System.Collections.Generic.Stack<T>` can be considerable, however, so the `System.Collections.Generic.Stack<T>.TrimExcess` method does nothing if the list is at more than 90 percent of capacity. This avoids incurring a large reallocation cost for a relatively small gain.

This method is an $O(n)$ operation, where n is `System.Collections.Generic.Stack<T>.Count`.

To reset a `System.Collections.Generic.Stack<T>` to its initial state, call the `System.Collections.Generic.Stack<T>.Clear` method before calling `System.Collections.Generic.Stack<T>.TrimExcess` method. Trimming an empty `System.Collections.Generic.Stack<T>` sets the capacity of the `System.Collections.Generic.Stack<T>` to the default capacity.

Stack<T>.Count Property

```
[ILAsm]  
.property instance int32 Count  
  
[C#]  
public int Count { get; }
```

Summary

Gets the number of elements contained in the `System.Collections.Generic.Stack<T>`.

Property Value

The number of elements contained in the `System.Collections.Generic.Stack<T>`.

Description

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

The capacity is always greater than or equal to `System.Collections.Generic.Stack<T>.Count`. If `System.Collections.Generic.Stack<T>.Count` exceeds the capacity while adding elements, the capacity is increased by automatically 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<T>.System.Collections.ICollection.IsSynchronized Property

```
[ILAsm]  
.property instance bool System.Collections.ICollection.IsSynchronized  
  
[C#]  
bool System.Collections.ICollection.IsSynchronized { get; }
```

Summary

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

Property Value

true if access to the System.Collections.ICollection is synchronized (thread safe); otherwise, false. In the default implementation of System.Collections.Generic.Stack<T>, this property always returns false.

Description

Default implementations of collections in System.Collections.Generic are not synchronized.

Enumerating through a collection is intrinsically not a thread-safe procedure. In the rare case where enumerations contend with write accesses, you must lock the collection during the entire enumeration. To allow the collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization.

System.Collections.ICollection.SyncRoot returns an object that can be used to synchronize access to the System.Collections.ICollection. Synchronization is effective only if all threads lock this object before accessing the collection.

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

Stack<T>.System.Collections.ICollection.SyncRoot Property

```
[ILAsm]  
.property instance object System.Collections.ICollection.SyncRoot  
  
[C#]  
object System.Collections.ICollection.SyncRoot { get; }
```

Summary

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

Property Value

An object that can be used to synchronize access to the System.Collections.ICollection. In the default implementation of System.Collections.Generic.Stack<T>, this property always returns the current instance.

Description

Default implementations of collections in System.Collections.Generic are not synchronized.

Enumerating through a collection is intrinsically not a thread-safe procedure. To guarantee thread safety during enumeration, you can lock the collection during the entire enumeration. To allow the collection to be accessed by multiple threads for reading and writing, you must implement your own synchronization.

System.Collections.ICollection.SyncRoot returns an object that can be used to synchronize access to the System.Collections.ICollection. Synchronization is effective only if all threads lock this object before accessing the collection. The following code shows the use of the System.Collections.ICollection.SyncRoot property for C#.

```
ICollection ic =...;  
lock (ic.SyncRoot) {  
    // Access the collection.  
}
```

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