

# System.Collections.ArrayList Class

```
[ILASM]
.class public serializable ArrayList extends System.Object
implements System.ICloneable,
System.Collections.ICollection,
System.Collections.IEnumerable, System.Collections.IList

[C#]
public class ArrayList: ICloneable, ICollection,
IEnumerable, IList
```

## Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

## Type Attributes:

- DefaultMemberAttribute("Item") [*Note:* This attribute requires the RuntimeInfrastructure library.]

## Implements:

- **System.Collections.IList**
- **System.Collections.ICollection**
- **System.Collections.IEnumerable**
- **System.ICloneable**

## Summary

Implements a variable-size **System.Collections.IList** that uses an array of objects to store its elements.

## Inherits From: System.Object

**Library:** BCL

**Thread Safety:** This class is safe for multiple readers and no concurrent writers.

## Description

**System.Collections.ArrayList** implements a variable-size **System.Collections.IList** that uses an array of objects to store the elements. A **System.Collections.ArrayList** has a

1       **System.Collections.ArrayList.Capacity**, which is the allocated  
2       length of the internal array. The total number of elements contained  
3       by a list is its **System.Collections.ArrayList.Count**. As elements are  
4       added to a list, its capacity is automatically increased as required by  
5       reallocating the internal array.

## 6       **Example**

7

8       The following example shows how to create, initialize, and display the  
9       values of a **System.Collections.ArrayList**.

```
10       [C#]
11
12       using System;
13       using System.Collections;
14
15       public class SamplesArrayList {
16
17           public static void Main() {
18
19               // Create and initialize a new ArrayList.
20               ArrayList myAL = new ArrayList();
21               myAL.Add("Hello");
22               myAL.Add("World");
23               myAL.Add("!");
24
25               // Display the properties and values of the ArrayList.
26               Console.WriteLine("myAL");
27               Console.WriteLine("Count: {0}", myAL.Count);
28               Console.WriteLine("Capacity: {0}", myAL.Capacity);
29               Console.Write("Values:");
30               PrintValues(myAL);
31           }
32
33       public static void PrintValues(IEnumerable myList) {
34
35           IEnumerator myEnumerator = myList.GetEnumerator();
36           while (myEnumerator.MoveNext())
37               Console.Write(" {0}", myEnumerator.Current);
38           Console.WriteLine();
39       }
40   }
```

41       The output is

42       myAL

43       Count: 3

```
1  
2     Capacity: 16  
3  
4     Values: Hello World !  
5
```

# 1 ArrayList() Constructor

```
2 [ILASM]  
3 public rtspecialname specialname instance void .ctor()  
4 [C#]  
5 public ArrayList()
```

## 6 Summary

7 Constructs and initializes a new instance of the  
8 **System.Collections.ArrayList** class that is empty and has the  
9 default initial **System.Collections.ArrayList.Capacity**.

## 10 Description

11 The default initial **System.Collections.ArrayList.Capacity** of a  
12 **System.Collections.ArrayList** is 16.

13

# 1 ArrayList(System.Int32) Constructor

```
2 [ILASM]  
3 public rtspecialname specialname instance void .ctor(int32  
4 capacity)  
5  
6 [C#]  
7 public ArrayList(int capacity)
```

## 7 Summary

8 Constructs and initializes a new instance of the  
9 **System.Collections.ArrayList** class that is empty and has the  
10 specified initial **System.Collections.ArrayList.Capacity**.

## 11 Parameters

12  
13

Parameter	Description
<i>capacity</i>	A <b>System.Int32</b> that specifies the number of elements that the new instance is initially capable of storing.

14  
15

## 15 Description

16 If *capacity* is zero, the **System.Collections.ArrayList.Capacity** of  
17 the current instance is set to 16 when the first element is added.

## 18 Exceptions

19  
20

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>capacity</i> < 0.

21  
22  
23

# 1 ArrayList(System.Collections.ICollection) 2 Constructor

```
3 [ILASM]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.Collections.ICollection c)  
  
6 [C#]  
7 public ArrayList(ICollection c)
```

## 8 Summary

9 Constructs and initializes a new instance of the  
10 **System.Collections.ArrayList** class with the elements from the  
11 specified **System.Collections.ICollection**. The initial  
12 **System.Collections.ArrayList.Count** and  
13 **System.Collections.ArrayList.Capacity** of the new list are both  
14 equal to the number of elements in the specified collection.

## 15 Parameters

16  
17

Parameter	Description
<i>c</i>	The <b>System.Collections.ICollection</b> whose elements are copied to the new list.

18

## 19 Description

20 The elements in the new **System.Collections.ArrayList** instance are  
21 in the same order as contained in *c*.

## 22 Exceptions

23  
24

Exception	Condition
<b>System.ArgumentNullException</b>	<i>c</i> is <b>null</b> .

25

26

27

# 1 ArrayList.Adapter(System.Collections.IList) 2 t) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.ArrayList Adapter(class  
6 System.Collections.IList list)  
  
7 [C#]  
8 public static ArrayList Adapter(IList list)
```

## 9 Summary

10 Creates a **System.Collections.ArrayList** that is a wrapper for the  
11 specified **System.Collections.IList**.

## 12 Parameters

13  
14

Parameter	Description
<i>list</i>	The <b>System.Collections.IList</b> to wrap.

15  
16  
17

## 16 Return Value

18 The **System.Collections.ArrayList** wrapper for *list*.

## 19 Description

20 This method returns a **System.Collections.ArrayList** that contains a  
21 reference to the **System.Collections.IList** *list*. Any modifications to  
22 the elements in either the returned list or *list* are reflected in the  
23 other.

24

25 [Note: The **System.Collections.ArrayList** class provides generic  
26 **System.Collections.ArrayList.Reverse**,  
27 **System.Collections.ArrayList.BinarySearch** and  
28 **System.Collections.ArrayList.Sort** methods. This wrapper provides  
29 a means to use those methods on **System.Collections.IList** *list*  
30 without implementing the methods for the list. Performing these  
31 operations through the wrapper may be less efficient than operations  
32 applied directly to the list.]

## 33 Exceptions

34  
35

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

1  
2  
3

<b>System.ArgumentNullException</b>	<i>list</i> is <b>null</b> .
-------------------------------------	------------------------------

# 1 ArrayList.Add(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual int32 Add(object value)  
4 [C#]  
5 public virtual int Add(object value)
```

## 6 Summary

7 Adds the specified **System.Object** to the end of the current instance.

## 8 Parameters

9  
10

Parameter	Description
<i>value</i>	The <b>System.Object</b> to be added to the end of the current instance.

11  
12  
13

## Return Value

14 A **System.Int32** that specifies the index of the current instance at  
15 which *value* has been added.

## 16 Behaviors

17 As described above.

## 18 Exceptions

19  
20

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

21  
22  
23

# 1 ArrayList.AddRange(System.Collections.ICollection) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void AddRange(class  
5 System.Collections.ICollection c)  
  
6 [C#]  
7 public virtual void AddRange(ICollection c)
```

## 8 Summary

9 Adds the elements of the specified **System.Collections.ICollection**  
10 to the end of the current instance.

## 11 Parameters

12  
13

Parameter	Description
c	The <b>System.Collections.ICollection</b> whose elements are added to the end of the current instance.

14  
15

## Description

## 16 Behaviors

17 As described above.

## 18 Default

19 If the **System.Collections.ArrayList.Count** of the current instance  
20 plus the size of the collection being added is greater than the  
21 **System.Collections.ArrayList.Capacity** of the current instance, the  
22 capacity of the current instance is either doubled or increased to the  
23 new **System.Collections.ArrayList.Count**, whichever is greater. The  
24 internal array is reallocated to accommodate the new elements and  
25 the existing elements are copied to the new array before the new  
26 elements are added.

27  
28  
29  
30  
31  
32  
33  
34  
35

[Note: For the default implementation, if the current instance can accommodate the new elements without increasing the **System.Collections.ArrayList.Capacity**, this method is an  $O(n)$  operation, where  $n$  is the number of elements to be added. If the capacity needs to be increased to accommodate the new elements, this method becomes an  $O(n+m)$  operation, where  $n$  is the number of elements to be added and  $m$  is **System.Collections.ArrayList.Count**.]

1 **Exceptions**  
2  
3

<b>Exception</b>	<b>Condition</b>
<b>System.ArgumentNullException</b>	<i>c</i> is <b>null</b> .
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

4  
5  
6

# 1 ArrayList.BinarySearch(System.Object, 2 System.Collections.IComparer) Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 BinarySearch(object  
5 value, class System.Collections.IComparer comparer)  
  
6 [C#]  
7 public virtual int BinarySearch(object value, IComparer  
8 comparer)
```

## 9 Summary

10 Searches the current instance for the specified **System.Object** using  
11 the specified **System.Collections.IComparer** implementation.

## 12 Parameters

13  
14

Parameter	Description
<i>value</i>	The <b>System.Object</b> for which to search.
<i>comparer</i>	The <b>System.Collections.IComparer</b> implementation to use when comparing elements. Specify <b>null</b> to use the <b>System.IComparable</b> implementation of each element.

15  
16  
17

## Return Value

18 A **System.Int32** that specifies the results of the search as follows:

Return Value	Description
The index of <i>value</i> in the current instance.	<i>value</i> was found.
The bitwise complement of the index of the first element that is greater than <i>value</i> .	<i>value</i> was not found, and at least one element in the current instance is greater than <i>value</i> .
The bitwise complement of the <b>System.Collections.ArrayList.Count</b> of the current instance.	<i>value</i> was not found, and <i>value</i> is greater than all elements in the current instance.

19  
20  
21  
22

[Note: If *value* is not found and the current instance is already sorted, the bitwise complement of the return value indicates the index in the current instance where *value* would be found.]

1 **Description**

2 [Note: A null reference can be compared with any type; therefore,  
3 comparisons with a null reference do not generate exceptions. A null  
4 reference evaluates to less than any non-null object, or equal to  
5 another null reference, when the two are compared.]

6 **Behaviors**

7 As described above.

8 **Default**

9 This method uses **System.Array.BinarySearch** to search for *value*.

10  
11 *value* is compared to elements in a binary search of the current  
12 instance until an element with a value greater than or equal to *value* is  
13 found. If *comparer* is **null**, the **System.IComparable** implementation  
14 of the element being compared -- or of *value* if the element being  
15 compared does not implement the interface -- is used to make the  
16 comparison. If *value* does not implement the **System.IComparable**  
17 interface and is compared to an element that does not implement the  
18 interface either, **System.ArgumentException** is thrown. If the  
19 current instance is not already sorted, correct results are not  
20 guaranteed.

21  
22 [Note: For the default implementation, this method is an O(log*n*)  
23 operation where *n* is equal to the  
24 **System.Collections.ArrayList.Count** of the current instance.]

25 **Exceptions**

26  
27

Exception	Condition
<b>System.ArgumentException</b>	<p><i>comparer</i> is <b>null</b>, and both <i>value</i> and at least one element in the current instance do not implement the <b>System.IComparable</b> interface.</p> <p>-or-</p> <p><i>comparer</i> is <b>null</b>, and <i>value</i> is not assignment-compatible with at least one element in the current instance.</p>

28  
29  
30

# 1 ArrayList.BinarySearch(System.Object)

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 BinarySearch(object  
5 value)  
6 [C#]  
7 public virtual int BinarySearch(object value)
```

### 8 Summary

9 Searches the current instance for the specified **System.Object**.

### 10 Parameters

11  
12

Parameter	Description
<i>value</i>	The <b>System.Object</b> for which to search.

13

### 14 Return Value

15

16 A **System.Int32** that specifies the results of the search as follows:

Return Value	Description
The index of <i>value</i> in the current instance.	<i>value</i> was found.
The bitwise complement of the index of the first element that is greater than <i>value</i> .	<i>value</i> was not found, and at least one element in the current instance is greater than <i>value</i> .
The bitwise complement of the <b>System.Collections.ArrayList.Count</b> of the current instance.	<i>value</i> was not found, and <i>value</i> is greater than all elements in the current instance.

17

18 [Note: If *value* is not found and the current instance is already sorted,  
19 the bitwise complement of the return value indicates the index in the  
20 current instance where *value* would be found.]

### 21 Description

22 [Note: A null reference can be compared with any type; therefore,  
23 comparisons with a null reference do not generate exceptions. A null  
24 reference evaluates to less than any non-null object, or equal to  
25 another null reference, when the two are compared.]

1 **Behaviors**

2 As described above.

3 **Default**

4 This method uses **System.Array.BinarySearch** to search for *value*.

5

6 *value* is compared to elements in a binary search of the current  
7 instance until an element with a value greater than or equal to *value* is  
8 found. The **System.IComparable** implementation of the element  
9 being compared -- or of *value* if the element being compared does not  
10 implement the interface -- is used to make the comparison. If *value*  
11 does not implement the **System.IComparable** interface and is  
12 compared to an element that does not implement the interface either,  
13 **System.ArgumentException** is thrown. If the current instance is not  
14 already sorted, correct results are not guaranteed.

15

16 [Note: For the default implementation, this method is an  $O(\log n)$   
17 operation where  $n$  is equal to the

18

**System.Collections.ArrayList.Count** of the current instance.]

19 **Exceptions**

20

21

Exception	Condition
<b>System.ArgumentException</b>	Both <i>value</i> and at least one element of the current instance do not implement the <b>System.IComparable</b> interface.  -or-  <i>value</i> is not assignment-compatible with at least one element in the current instance.

22

23

24

# 1 ArrayList.BinarySearch(System.Int32, 2 System.Int32, System.Object, 3 System.Collections.IComparer) Method

```
4 [ILASM]  
5 .method public hidebysig virtual int32 BinarySearch(int32  
6 index, int32 count, object value, class  
7 System.Collections.IComparer comparer)
```

```
8 [C#]  
9 public virtual int BinarySearch(int index, int count,  
10 object value, IComparer comparer)
```

## 11 Summary

12 Searches the specified range in the current instance for the specified  
13 **System.Object** using the specified **System.Collections.IComparer**  
14 implementation.

## 15 Parameters

16  
17

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the index at which searching starts. This value is greater than or equal to zero, and less than the <b>System.Collections.ArrayList.Count</b> of the current instance.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to search, beginning with <i>index</i> . This value is greater than or equal to zero, and less than or equal to the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> .
<i>value</i>	The <b>System.Object</b> for which to search.
<i>comparer</i>	The <b>System.Collections.IComparer</b> implementation to use when comparing elements. Specify <b>null</b> to use the <b>System.IComparable</b> implementation of each element.

18  
19  
20

## Return Value

21 A **System.Int32** that specifies the results of the search as follows:

Return Value	Description
The index of <i>value</i> in the current instance.	<i>value</i> was found.
The bitwise complement of the	<i>value</i> was not found. and at least one element in

index of the first element that is greater than <i>value</i> .	the range of <i>index</i> to <i>index</i> + <i>count</i> in the current instance is greater than <i>value</i> .
The bitwise complement of ( <i>index</i> + <i>count</i> )	<i>value</i> was not found, and <i>value</i> is greater than all elements in the range of <i>index</i> to <i>index</i> + <i>count</i> in the current instance.

1  
2  
3  
4  
5

[*Note*: If *value* is not found and the current instance is already sorted, the bitwise complement of the return value indicates the index in the current instance where *value* would be found in the range of *index* to *index* + *count*.]

6 **Description**

7  
8  
9  
10

[*Note*: A null reference can be compared with any type; therefore, comparisons with a null reference do not generate exceptions. A null reference evaluates to less than any non-null object, or equal to another null reference, when the two are compared.]

11 **Behaviors**

12

As described above.

13 **Default**

14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29

This method uses **System.Array.BinarySearch** to search for *value*.

*value* is compared to elements in a binary search of the range of *index* to *index* + *count* in the current instance until an element with a value greater than or equal to *value* is found or the end of the range is reached. If *comparer* is **null**, the **System.IComparable** implementation of the element being compared -- or of *value* if the element being compared does not implement the interface -- is used to make the comparison. If *value* does not implement the **System.IComparable** interface and is compared to an element that does not implement the interface either, **System.ArgumentException** is thrown. If the current instance is not already sorted, correct results are not guaranteed.

[*Note*: For the default implementation, this method is an O(log*count*) operation.]

30 **Exceptions**

31  
32

Exception	Condition
<b>System.ArgumentException</b>	<b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> < <i>count</i> .  -or-

	<p><i>comparer</i> is <b>null</b>, and both <i>value</i> and at least one element of the current instance do not implement the <b>System.IComparable</b> interface.</p> <p>-or-</p> <p><i>comparer</i> is <b>null</b>, and <i>value</i> is not assignment-compatible with at least one element in the current instance.</p>
<p><b>System.ArgumentOutOfRangeException</b></p>	<p><i>index</i> &lt; 0.</p> <p>-or-</p> <p><i>count</i> &lt; 0.</p>

- 1
- 2
- 3

# 1 ArrayList.Clear() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void Clear()  
4 [C#]  
5 public virtual void Clear()
```

## 6 Summary

7 Sets the elements in the current instance to zero, **false**, or **null**,  
8 depending upon the element type.

## 9 Description

10 [Note: This method is implemented to support the  
11 **System.Collections.IList** interface.]

## 12 Behaviors

13 Reference-type elements are set to **null**. Value-type elements are set  
14 to zero, except for **System.Boolean** elements, which are set to **false**.

## 15 Default

16 This method uses **System.Array.Clear** to reset the values of the  
17 current instance. **System.Collections.ArrayList.Count** is set to zero.  
18 **System.Collections.ArrayList.Capacity** is not changed.

## 19 Usage

20 To reset the **System.Collections.ArrayList.Capacity** of the current  
21 instance, call **System.Collections.ArrayList.TrimToSize** or set the  
22 **System.Collections.ArrayList.Capacity** property directly.

## 23 Exceptions

24  
25

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

26  
27  
28

# 1 ArrayList.Clone() Method

```
2 [ILASM]  
3 .method public hidebysig virtual object Clone()  
4 [C#]  
5 public virtual object Clone()
```

## 6 Summary

7 Returns a **System.Object** that is a copy of the current instance.

## 8 Return Value

9

10 A **System.Object** that is a copy of the current instance.

## 11 Description

12 [*Note:* This method is implemented to support the  
13 **System.ICloneable** interface.]

## 14 Behaviors

15 As described above.

## 16 Default

17 This method uses **System.Array.Copy** to clone the current instance.

18

# 1 ArrayList.Contains(System.Object)

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual bool Contains(object item)  
5  
6 [C#]  
7 public virtual bool Contains(object item)
```

### 7 Summary

8 Determines whether the specified **System.Object** is contained in the  
9 current instance.

### 10 Parameters

Parameter	Description
<i>item</i>	The <b>System.Object</b> to locate in the current instance.

### 13 Return Value

14 **true** if *item* is contained in the current instance; otherwise, **false**.

### 17 Description

18 [Note: This method is implemented to support the  
19 **System.Collections.IList** interface.]

### 20 Behaviors

21 As described above.

### 22 Default

23 This method determines equality by calling the  
24 **System.Object.Equals** implementation of the type of *item*.

25  
26 [Note: For the default implementation, this method is an  $O(n)$   
27 operation where  $n$  is equal to the  
28 **System.Collections.ArrayList.Count** of the current instance. If the  
29 current instance is sorted, it is more efficient to call  
30 **System.Collections.ArrayList.BinarySearch** method.]

31

# 1 ArrayList.CopyTo(System.Array, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void CopyTo(class  
5 System.Array array, int32 arrayIndex)  
  
6 [C#]  
7 public virtual void CopyTo(Array array, int arrayIndex)
```

## 8 Summary

9 Copies the elements from the current instance to the specified  
10 **System.Array**, starting at the specified index of the array.

## 11 Parameters

12  
13

Parameter	Description
<i>array</i>	The one-dimensional <b>System.Array</b> that is the destination of the elements copied from the current instance. The <b>System.Array.Length</b> of this array is greater than or equal to the sum of <i>arrayIndex</i> and the <b>System.Collections.ArrayList.Count</b> of the current instance.
<i>arrayIndex</i>	A <b>System.Int32</b> that specifies the first index of <i>array</i> to which the elements of the current instance are copied. This value is greater than or equal to zero, and less than <i>array.Length</i> .

14  
15

## 15 Description

16 [Note: This method is implemented to support the  
17 **System.Collections.IList** interface.]

## 18 Behaviors

19 As described above.

## 20 Default

21 This method uses **System.Array.Copy** to copy the current instance to  
22 *array*.

## 23 Exceptions

24  
25

Exception	Condition
<b>System.ArgumentNullException</b>	<i>array</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>arrayIndex</i> < 0.

<p><b>System.ArgumentException</b></p>	<p><i>array</i> has more than one dimension.</p> <p>-or-</p> <p><i>arrayIndex</i> &gt;= <i>array.Length</i>.</p> <p>-or-</p> <p><i>arrayIndex</i> +  <b>System.Collections.ArrayList.Count</b>  of the current instance &gt; <i>array.Length</i>.</p>
<p><b>System.InvalidCastException</b></p>	<p>At least one element in the current instance is not assignment-compatible with the type of <i>array</i>.</p>

1  
2  
3

# 1 ArrayList.CopyTo(System.Int32, 2 System.Array, System.Int32, 3 System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig virtual void CopyTo(int32 index,  
6 class System.Array array, int32 arrayIndex, int32 count)  
  
7 [C#]  
8 public virtual void CopyTo(int index, Array array, int  
9 arrayIndex, int count)
```

## 10 Summary

11 Copies the specified range of elements from the current instance to  
12 the specified **System.Array**, starting at the specified index of the  
13 array.

## 14 Parameters

15  
16

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the index in the current instance at which copying begins. This value is greater than or equal to 0, and less than the <b>System.Collections.ArrayList.Count</b> of the current instance.
<i>array</i>	The one-dimensional <b>System.Array</b> that is the destination of the elements copied from the current instance.
<i>arrayIndex</i>	A <b>System.Int32</b> that specifies the first index of <i>array</i> to which the elements of the current instance are copied. This value is greater than or equal to zero, and less than <i>array.Length</i> minus <i>count</i> .
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to copy. This value is greater than or equal to 0, and less than both the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> and <i>array.Length</i> minus <i>arrayIndex</i> .

17  
18

## 19 Behaviors

20 As described above.

## 21 Default

22 This method uses **System.Array.Copy** to copy the current instance to  
23 *array*.

1 Exceptions  
2  
3

Exception	Condition
<b>System.ArgumentNullException</b>	<i>array</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or- <i>arrayIndex</i> < 0. -or- <i>count</i> < 0.
<b>System.ArgumentException</b>	<i>array</i> has more than one dimension. -or- <i>index</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance. -or- <i>count</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> . -or- <i>count</i> >= <i>array.Length</i> - <i>arrayIndex</i> .
<b>System.InvalidCastException</b>	At least one element of the current instance is not assignment-compatible with the type of <i>array</i> .

4  
5  
6

# 1 ArrayList.CopyTo(System.Array) Method

```
2 [ILASM]
3 .method public hidebysig virtual void CopyTo(class
4 System.Array array)
5
6 [C#]
7 public virtual void CopyTo(Array array)
```

## 7 Summary

8 Copies the elements from the current instance to the specified  
9 **System.Array**.

## 10 Parameters

Parameter	Description
<i>array</i>	The one-dimensional <b>System.Array</b> that is the destination of the elements copied from the current instance. The <b>System.Array.Length</b> of this array is greater than or equal to the <b>System.Collections.ArrayList.Count</b> of the current instance.

13  
14

## 15 Behaviors

16 As described above.

## 17 Default

18 This method uses **System.Array.Copy** to copy the current instance to  
19 *array*.

## 20 Exceptions

21  
22

Exception	Condition
<b>System.ArgumentNullException</b>	<i>array</i> is <b>null</b> .
<b>System.ArgumentException</b>	<i>array</i> has more than one dimension.
	-or- <b>System.Collections.ArrayList.Count</b> of the current instance > <i>array</i> .Length.
<b>System.InvalidCastException</b>	At least one element in the current instance is

1  
2  
3

	not assignment-compatible with the type of <i>array</i> .
--	---

# 1 ArrayList.FixedSize(System.Collections.Ar 2 rayList) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.ArrayList FixedSize(class  
6 System.Collections.ArrayList list)  
  
7 [C#]  
8 public static ArrayList FixedSize(ArrayList list)
```

## 9 Summary

10 Returns a **System.Collections.ArrayList** wrapper with a fixed size.

## 11 Parameters

12  
13

Parameter	Description
<i>list</i>	The <b>System.Collections.ArrayList</b> to wrap.

14  
15  
16

## Return Value

17 A **System.Collections.ArrayList** wrapper with a fixed size.

## 18 Description

19 This method returns a fixed-size **System.Collections.ArrayList** that  
20 contains a reference to *list*. Operations that attempt add to or delete  
21 from this new list will throw **System.NotSupportedException**. Any  
22 modifications of the elements in either the returned list or *list* will be  
23 reflected in the other.

24  
25  
26  
27  
28

[Note: The **System.Collections.ArrayList.IsFixedSize** property of  
the new list is **true**. Every other property value of the new list  
references the same property value of *list*.

29 By performing operations on the new list, this wrapper can be used to  
30 prevent additions to and deletions from the  
31 **System.Collections.ArrayList** *list*. The elements of the list can still  
32 be modified by operations on the returned list.]

## 33 Exceptions

34  
35

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

1  
2  
3

<b>System.ArgumentNullException</b>	<i>list</i> is <b>null</b> .
-------------------------------------	------------------------------

# 1 ArrayList.GetEnumerator() Method

```
2 [ILASM]
3 .method public hidebysig virtual class
4 System.Collections.IEnumerator GetEnumerator()
5
6 [C#]
7 public virtual IEnumerator GetEnumerator()
```

## 7 Summary

8 Returns a **System.Collections.IEnumerator** for the current instance.

## 9 Return Value

10

11 A **System.Collections.IEnumerator** for the current instance.

## 12 Description

13 If the elements of the current instance are modified while an  
14 enumeration is in progress, a call to  
15 **System.Collections.IEnumerator.MoveNext** or  
16 **System.Collections.IEnumerator.Current** throws  
17 **System.InvalidOperationException**.

18

19 [*Note:* For detailed information regarding the use of an enumerator,  
20 see **System.Collections.IEnumerator**.

21

22 This property is implemented to support the  
23 **System.Collections.IList** interface.]

## 24 Behaviors

25 As described above.

26

# 1 ArrayList.GetEnumerator(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual class  
5 System.Collections.IEnumerator GetEnumerator(int32 index,  
6 int32 count)  
  
7 [C#]  
8 public virtual IEnumerator GetEnumerator(int index, int  
9 count)
```

## 10 Summary

11 Returns a **System.Collections.IEnumerator** for the specified section  
12 of the current instance.

## 13 Parameters

14  
15

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the index of the current instance before which the enumerator is initially placed. This value is greater than or equal to 0, and less than the <b>System.Collections.ArrayList.Count</b> of the current instance.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements, beginning with <i>index</i> , in the current instance over which the enumerator can iterate. This value is greater than or equal to 0, and less than or equal to the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> .

16  
17  
18

## Return Value

19 A **System.Collections.IEnumerator** that can iterate over the range  
20 of *index* to *index* + *count* in the current instance.

## 21 Description

22 The enumerator only enumerates over the range of the current  
23 instance from *index* to *index* + *count*. If the elements of the current  
24 instance are modified while an enumeration is in progress, a call to  
25 **System.Collections.IEnumerator.MoveNext** or  
26 **System.Collections.IEnumerator.Current** will throw  
27 **System.InvalidOperationException**.

28  
29  
30

[Note: For detailed information regarding the use of an enumerator,  
see **System.Collections.IEnumerator**.]

1 **Behaviors**

2 As described above.

3 **Default**

4 The enumerator is initially placed just before the element at position  
5 *index* in the current instance. A call to  
6 **System.Collections.IEnumerator.Reset** returns the enumerator to  
7 this position.

8  
9 If the enumerator is positioned over an element in the range of *index* -  
10 1 to *index* + *count*, and the elements of the current instance have not  
11 been modified while the enumeration was in progress, a call to  
12 **System.Collections.IEnumerator.MoveNext** either returns **true**  
13 and advances the enumerator one element in the current instance, or  
14 returns **false** and leaves the enumerator in its current position.

15 **Exceptions**

16  
17

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or- <i>count</i> < 0.
<b>System.ArgumentException</b>	<i>index</i> + <i>count</i> > <b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> .

18  
19  
20

# 1 ArrayList.GetRange(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual class  
5 System.Collections.ArrayList GetRange(int32 index, int32  
6 count)  
  
7 [C#]  
8 public virtual ArrayList GetRange(int index, int count)
```

## 9 Summary

10 Returns a **System.Collections.ArrayList** that represents the  
11 specified range of the current instance.

## 12 Parameters

13  
14

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index in the current instance at which the range starts. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>count</i> , inclusive.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements in the range. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> , inclusive.

15

## 16 Return Value

17

18 A **System.Collections.ArrayList** that represents the range in the  
19 current instance from *index* to *index + count*.

## 20 Behaviors

21 As described above.

## 22 Default

23 This method does not create copies of the elements: the new  
24 **System.Collections.ArrayList** instance is a view window into the  
25 source list. Therefore, all subsequent changes to the source list must  
26 be done through this view window **System.Collections.ArrayList**. If  
27 changes are made directly to the source list, the view window list is  
28 invalidated and any operations on it throw  
29 **System.InvalidOperationException**.

1 **Exceptions**  
2  
3

<b>Exception</b>	<b>Condition</b>
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or- <i>count</i> < 0.
<b>System.ArgumentException</b>	<b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> < <i>count</i> .

4  
5  
6

# 1 ArrayList.IndexOf(System.Object) Method

```
2 [ILASM]
3 .method public hidebysig virtual int32 IndexOf(object
4 value)
5 [C#]
6 public virtual int IndexOf(object value)
```

## 7 Summary

8 Searches the current instance, returning the index of the first  
9 occurrence of the specified **System.Object**.

## 10 Parameters

11  
12

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in current instance.

13  
14  
15

## 14 Return Value

16 A **System.Int32** that specifies the index of the first occurrence of  
17 *value* in the current instance, if found; otherwise, -1.  
18  
19 [Note: This provides the caller with a standard code for a failed  
20 search.]

## 21 Description

22 [Note: This method is implemented to support the  
23 **System.Collections.IList** interface.]

## 24 Behaviors

25 As described above.

## 26 Default

27 This method uses **System.Array.IndexOf** to search the current  
28 instance for *value*.

29  
30 [Note: For the default implementation, this method performs a linear  
31 search. On average, this is an  $O(n/2)$  operation, where *n* is *count*. The  
32 longest search is an  $O(n)$  operation.]

33

# 1 ArrayList.IndexOf(System.Object, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 IndexOf(object  
5 value, int32 startIndex, int32 count)  
  
6 [C#]  
7 public virtual int IndexOf(object value, int startIndex,  
8 int count)
```

## 9 Summary

10 Searches the current instance, returning the index of the first  
11 occurrence of the specified **System.Object** in the specified range.

## 12 Parameters

13  
14

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in current instance.
<i>startIndex</i>	A <b>System.Int32</b> that specifies the index at which to begin searching. This value is less than or equal to zero, and greater than the <b>System.Collections.ArrayList.Count</b> of the current instance.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to search. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>startIndex</i> , inclusive.

15  
16  
17

## Return Value

18 A **System.Int32** that specifies the index of the first occurrence of  
19 *value* in the current instance, within the range *startIndex* to *startIndex*  
20 + *count*, if found; otherwise, -1.

21  
22  
23

[Note: This provides the caller with a standard code for a failed search.]

## 24 Description

## 25 Behaviors

26 As described above.

## 27 Default

28 This method uses **System.Array.IndexOf** to search the current  
29 instance for *value*.

1  
2  
3  
4  
5  
6  
7

[*Note:* For the default implementation, this method performs a linear search. On average, this is an  $O(n/2)$  operation, where  $n$  is *count*. The longest search is an  $O(n)$  operation.]

### Exceptions

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>startIndex</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.  -or- <i>count</i> < 0.  -or- <i>count</i> > <b>System.Collections.ArrayList.Count</b> of the current instance - <i>startIndex</i> .

8  
9  
10

# 1 ArrayList.IndexOf(System.Object, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 IndexOf(object  
5 value, int32 startIndex)  
  
6 [C#]  
7 public virtual int IndexOf(object value, int startIndex)
```

## 8 Summary

9 Searches the current instance, returning the index of the first  
10 occurrence of the specified **System.Object** in the specified range.

## 11 Parameters

12  
13

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in current instance.
<i>startIndex</i>	A <b>System.Int32</b> that specifies the index at which searching begins. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance, inclusive.

14  
15  
16

## 15 Return Value

17 A **System.Int32** that specifies the index of the first occurrence of  
18 *value* in the current instance, if found within the range *startIndex* to  
19 the end of the current instance; otherwise, -1.

20  
21 [Note: This provides the caller with a standard code for a failed  
22 search.]

## 23 Description

## 24 Behaviors

25 As described above.

## 26 Default

27 This method uses **System.Array.IndexOf** to search the current  
28 instance for *value*.

29  
30 [Note: For the default implementation, this method performs a linear  
31 search. On average, this is an  $O(n/2)$  operation, where *n* is *count*. The  
32 longest search is an  $O(n)$  operation.]

1 **Exceptions**  
2  
3

<b>Exception</b>	<b>Condition</b>
<b>System.ArgumentOutOfRangeException</b>	<i>startIndex</i> < 0. -or- <i>startIndex</i> > <b>System.Collections.ArrayList.Count</b> of the current instance.

4  
5  
6

# 1 ArrayList.Insert(System.Int32, 2 System.Object) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void Insert(int32 index,  
5 object value)  
  
6 [C#]  
7 public virtual void Insert(int index, object value)
```

## 8 Summary

9 Inserts the specified **System.Object** into the current instance at the  
10 specified index.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the index in the current instance at which <i>value</i> is inserted. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance, inclusive.
<i>value</i>	The <b>System.Object</b> to insert.

14  
15

## Description

16 [Note: This method is implemented to support the  
17 **System.Collections.IList** interface.]

## 18 Behaviors

19 As described above.

## 20 Default

21 If the **System.Collections.ArrayList.Count** of the current instance is  
22 equal to the **System.Collections.ArrayList.Capacity** of the current  
23 instance, the capacity of the list is doubled by automatically  
24 reallocating the internal array before the new element is inserted. If  
25 *index* is equal to the **System.Collections.ArrayList.Count** of the  
26 current instance, *value* is added to the end of the current instance.

## 27 Exceptions

28  
29

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or- <i>index</i> > <b>System.Collections.ArrayList.Count</b> of the current instance.
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

1  
2  
3

# 1 ArrayList.InsertRange(System.Int32, 2 System.Collections.ICollection) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void InsertRange(int32  
5 index, class System.Collections.ICollection c)  
  
6 [C#]  
7 public virtual void InsertRange(int index, ICollection c)
```

## 8 Summary

9 Inserts the elements of the specified **System.Collections.ICollection**  
10 at the specified index of the current instance.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the index in the current instance at which the new elements are inserted. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance, inclusive.
<i>c</i>	The <b>System.Collections.ICollection</b> whose elements are inserted into the current instance.

14  
15

## 16 Behaviors

17 As described above.

## 18 Default

19 If the **System.Collections.ArrayList.Count** of the current instance  
20 plus the size of **System.Collections.ICollection** *c* is greater than the  
21 **System.Collections.ArrayList.Capacity** of the current instance, the  
22 capacity of the current instance is either doubled or increased to the  
23 new count, whichever yields a greater capacity. The internal array is  
24 reallocated to accommodate the new elements. If *index* is equal to the  
25 **System.Collections.ArrayList.Count** of the current instance, the  
26 elements of *c* are added to the end of the current instance.

27  
28 The order of the elements in the **System.Collections.ICollection** *c* is  
29 preserved in the current instance.

1 **Exceptions**  
2  
3

<b>Exception</b>	<b>Condition</b>
<b>System.ArgumentNullException</b>	<i>c</i> is <b>null</b> .
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. <i>index</i> > <b>System.Collections.ArrayList.Count</b> of the current instance.
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

4  
5  
6

# 1 ArrayList.LastIndexOf(System.Object, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 LastIndexOf(object  
5 value, int32 startIndex, int32 count)  
  
6 [C#]  
7 public virtual int LastIndexOf(object value, int  
8 startIndex, int count)
```

## 9 Summary

10 Searches the current instance, returning the index of the last  
11 occurrence of the specified **System.Object** in the specified range.

## 12 Parameters

13  
14

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in the current instance.
<i>startIndex</i>	A <b>System.Int32</b> that specifies the index at which searching starts.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to search, beginning with <i>startIndex</i> .

15  
16  
17

## 16 Return Value

18 A **System.Int32** that specifies the index of the last occurrence of  
19 value in the current instance, within the range *startIndex* + *count*  
20 through *startIndex*, if found; otherwise, -1.

21  
22  
23

[Note: This provides the caller with a standard code for a failed search.]

## 24 Description

## 25 Behaviors

26 As described above.

## 27 Default

28 This method uses **System.Array.LastIndexOf** to search the current  
29 instance for *value*.

30  
31

[Note: For the default implementation, this method performs a linear

1 search. On average, this is an  $O(count/2)$  operation. The longest  
2 search is an  $O(count)$  operation.]

3 **Exceptions**

4  
5

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>startIndex</i> < 0.  -or-  <i>count</i> < 0.  -or-  <i>startIndex</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.  -or-  <i>count</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.  -or-  <i>count</i> >= <i>startIndex</i> .

6  
7  
8

# 1 ArrayList.LastIndexOf(System.Object, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 LastIndexOf(object  
5 value, int32 startIndex)  
  
6 [C#]  
7 public virtual int LastIndexOf(object value, int  
8 startIndex)
```

## 9 Summary

10 Searches the current instance, returning the index of the last  
11 occurrence of the specified **System.Object** in the specified range of  
12 the current instance.

## 13 Parameters

14  
15

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in the current instance.
<i>startIndex</i>	A <b>System.Int32</b> that specifies the index at which searching starts. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance, inclusive.

16  
17  
18

## Return Value

19 A **System.Int32** that specifies the index of the last occurrence of  
20 *value* in the range of *startIndex* through the first element of the  
21 current instance, if found; otherwise, -1.

22  
23  
24

[Note: This provides the caller with a standard code for a failed search.]

## 25 Description

## 26 Behaviors

27 As described above.

## 28 Default

29 This method uses **System.Array.LastIndexOf** to search the current  
30 instance for *value*.

31  
32

[Note: For the default implementation, this method performs a linear

1 search. On average, this is an  $O(count/2)$  operation. The longest  
2 search is an  $O(count)$  operation.]

3 **Exceptions**

4  
5

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>startIndex</i> < 0. -or- <i>startIndex</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.

6  
7  
8

# 1 ArrayList.LastIndexOf(System.Object) 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual int32 LastIndexOf(object  
5 value)  
6 [C#]  
7 public virtual int LastIndexOf(object value)
```

## 8 Summary

9 Searches the current instance, returning the index of the last  
10 occurrence of the specified **System.Object**.

## 11 Parameters

12  
13

Parameter	Description
<i>value</i>	The <b>System.Object</b> to locate in the current instance.

14  
15  
16

## Return Value

17 A **System.Int32** that specifies the index of the last occurrence of  
18 *value* in the current instance, if found; otherwise, -1.

19  
20  
21

[*Note:* This provides the caller with a standard code for a failed search.]

## 22 Description

## 23 Behaviors

24 As described above.

## 25 Default

26 This method uses **System.Array.LastIndexOf** to search the current  
27 instance for *value*.

28  
29  
30  
31  
32

[*Note:* For the default implementation, this method performs a linear search. On average, this is an  $O(n/2)$  operation, where  $n$  is **System.Collections.ArrayList.Count** of the current instance. The longest search is an  $O(n)$  operation.]

33

# 1 ArrayList.ReadOnly(System.Collections.Array 2 rayList) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.ArrayList ReadOnly(class  
6 System.Collections.ArrayList list)  
  
7 [C#]  
8 public static ArrayList ReadOnly(ArrayList list)
```

## 9 Summary

10 Returns a read-only **System.Collections.ArrayList** wrapper.

## 11 Parameters

12  
13

Parameter	Description
<i>list</i>	The <b>System.Collections.ArrayList</b> to wrap.

14

## 15 Return Value

16

17 A read-only **System.Collections.ArrayList** wrapper around *list*.

## 18 Description

19 This method returns a read-only **System.Collections.ArrayList** that  
20 contains a reference to *list*. Operations that attempt add to, delete  
21 from, or modify the elements of this new list will throw  
22 **System.NotSupportedException**. Any modifications of the elements  
23 *list* will be reflected in the new list.

24

25 [Note: The **System.Collections.ArrayList.IsReadOnly** property of  
26 the new list is **true**. Every other property value of the new list  
27 references the same property value of *list*.

28

29 By performing operations on the new list, this wrapper can be used to  
30 prevent additions to, deletions from, and modifications of the  
31 **System.Collections.ArrayList** *list*.]

## 32 Exceptions

33

34

Exception	Condition
<b>System.ArgumentNullException</b>	<i>list</i> is <b>null</b> .

1  
2  
3

# ArrayList.Remove(System.Object) Method

```
[ILASM]
.method public hidebysig virtual void Remove(object obj)
[C#]
public virtual void Remove(object obj)
```

## Summary

Removes the first occurrence of the specified **System.Object** from the current instance.

## Parameters

Parameter	Description
<i>obj</i>	The <b>System.Object</b> to remove from the current instance.

## Description

[*Note:* This method is implemented to support the **System.Collections.IList** interface.]

## Behaviors

As described above.

## Default

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

If and only if *obj* is found in the current instance, *obj* is removed from the current instance, the rest of the elements are shifted down to fill the position vacated by *obj*, the **System.Collections.ArrayList.Count** of the current instance is decreased by one, and the position that was previously the last element in the current instance is set to **null**. If *obj* is not found in the current instance, the current instance remains unchanged.

[*Note:* For the default implementation, this method performs a linear search. On average, this is an  $O(n/2)$  operation, where  $n$  is **System.Collections.ArrayList.Count** of the current instance. The longest search is an  $O(n)$  operation.]

## Exceptions

1  
2  
3

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

# 1 ArrayList.RemoveAt(System.Int32)

## 2 Method

```
3 [ILASM]  
4 .method public hidebysig virtual void RemoveAt(int32 index)  
5 [C#]  
6 public virtual void RemoveAt(int index)
```

### 7 Summary

8 Removes the element at the specified index from the current instance.

### 9 Parameters

10  
11

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index of the element to remove from the current instance. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance, inclusive.

12  
13

### Description

14 [Note: This method is implemented to support the  
15 **System.Collections.IList** interface.]

### 16 Behaviors

17 As described above.

### 18 Default

19 The element at position *index* is removed from the current instance,  
20 the rest of the elements are shifted down to fill the position vacated by  
21 that element, the **System.Collections.ArrayList.Count** of the  
22 current instance is decreased by one, and the position that was  
23 previously the last element in the current instance is set to **null**.

### 24 Exceptions

25  
26

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or-

1  
2  
3

	<i>index</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

# 1 ArrayList.RemoveRange(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void RemoveRange(int32  
5 index, int32 count)  
  
6 [C#]  
7 public virtual void RemoveRange(int index, int count)
```

## 8 Summary

9 Removes the specified range of elements from the current instance.

## 10 Parameters

11  
12

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index of the first element of the range of elements in the current instance to remove. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>count</i> , inclusive.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to remove. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> , inclusive.

13  
14

## 15 Behaviors

16 As described above.

## 17 Default

18 The elements in the range of *index* to *index + count* are removed from  
19 the current instance, the rest of the elements are shifted down to fill  
20 the position vacated by those elements, the  
21 **System.Collections.ArrayList.Count** of the current instance is  
22 decreased by *count*, and the *count* positions that were previously the  
23 last elements in the current instance are set to **null**.

## 24 Exceptions

25  
26

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0.

	-or- <i>count</i> < 0.
<b>System.ArgumentException</b>	<b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> < <i>count</i> .
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

- 1
- 2
- 3

# 1 ArrayList.Repeat(System.Object, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.ArrayList Repeat(object value, int32  
6 count)  
  
7 [C#]  
8 public static ArrayList Repeat(object value, int count)
```

## 9 Summary

10 Returns a new **System.Collections.ArrayList** whose elements are  
11 copies of the specified **System.Object**.

## 12 Parameters

13  
14

Parameter	Description
<i>value</i>	The <b>System.Object</b> used to initialize the new <b>System.Collections.ArrayList</b> instance.
<i>count</i>	A <b>System.Int32</b> that specifies the number of times <i>value</i> is copied into the new <b>System.Collections.ArrayList</b> instance.

15  
16  
17

## 16 Return Value

18 A new **System.Collections.ArrayList** with *count* number of elements,  
19 all of which are copies of *value*.

## 20 Description

21 If *count* is less than the default initial capacity, 16, the  
22 **System.Collections.ArrayList.Capacity** of the new  
23 **System.Collections.ArrayList** instance is set to the default initial  
24 capacity. Else, the capacity is set to *count*. The  
25 **System.Collections.ArrayList.Count** of the new instance is set to  
26 *count*.

## 27 Exceptions

28  
29

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>count</i> < 0.

1  
2  
3

# 1 ArrayList.Reverse(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void Reverse(int32 index,  
5 int32 count)  
  
6 [C#]  
7 public virtual void Reverse(int index, int count)
```

## 8 Summary

9 Reverses the sequence of the elements in the specified range of the  
10 current instance.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index in the current instance at which reversing starts. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>count</i> , inclusive.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to reverse. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> , inclusive.

14  
15

## 16 Behaviors

17 As described above.

## 18 Default

19 This method uses **System.Array.Reverse** to modify the ordering of  
20 the current instance.

## 21 Exceptions

22  
23

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or-

	<i>count</i> < 0.
<b>System.ArgumentException</b>	<b>System.Collections.ArrayList.Count</b> of the current instance - <i>index</i> < <i>count</i> .
<b>System.NotSupportedException</b>	The current instance is read-only.

- 1
- 2
- 3

# 1 ArrayList.Reverse() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void Reverse()  
4 [C#]  
5 public virtual void Reverse()
```

## 6 Summary

7 Reverses the sequence of the elements in the current instance.

## 8 Behaviors

9 As described above.

## 10 Default

11 This method uses **System.Array.Reverse** to modify the ordering of  
12 the elements in the current instance.

## 13 Exceptions

14  
15

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only.

16  
17  
18

# 1 ArrayList.SetRange(System.Int32, 2 System.Collections.ICollection) Method

```
3 [ILASM]  
4 .method public hidebysig virtual void SetRange(int32 index,  
5 class System.Collections.ICollection c)  
  
6 [C#]  
7 public virtual void SetRange(int index, ICollection c)
```

## 8 Summary

9 Copies the elements of the specified **System.Collections.ICollection**  
10 to a range in the current instance.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index in the current instance at which to start copying the elements of <i>c</i> . This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>c.Count</i> , inclusive.
<i>c</i>	The <b>System.Collections.ICollection</b> whose elements to copy to the current instance.

14  
15

## 16 Behaviors

17 As described above.

## 18 Default

19 This method uses the **System.Collections.ICollection.CopyTo**  
20 implementation of **System.Collections.ICollection c**.

## 21 Exceptions

22  
23

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or- <b>Svstem.Collections.ArrayvList.Count</b>

- 1
- 2
- 3

	of the current instance - <i>index</i> < <i>c.Count</i> .
<b>System.ArgumentNullException</b>	<i>c</i> is <b>null</b> .
<b>System.NotSupportedException</b>	The current instance is read-only.

# 1 ArrayList.Sort(System.Int32, 2 System.Int32, 3 System.Collections.IComparer) Method

```
4 [ILASM]  
5 .method public hidebysig virtual void Sort(int32 index,  
6 int32 count, class System.Collections.IComparer comparer)  
  
7 [C#]  
8 public virtual void Sort(int index, int count, IComparer  
9 comparer)
```

## 10 Summary

11 Sorts the elements in the specified range of the current instance using  
12 the specified **System.Collections.IComparer** implementation.

## 13 Parameters

14  
15

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index at which sorting starts. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>count</i> , inclusive.
<i>count</i>	A <b>System.Int32</b> that specifies the number of elements to sort. This value is between 0 and the <b>System.Collections.ArrayList.Count</b> of the current instance minus <i>index</i> , inclusive.
<i>comparer</i>	The <b>System.Collections.IComparer</b> implementation to use when comparing elements. Specify <b>null</b> to use the <b>System.IComparable</b> implementation of each element in the current instance.

16

## 17 Description

## 18 Behaviors

19 As described above.

## 20 Default

21 If *comparer* is **null**, the **System.IComparable** implementation of  
22 each element in the current instance is used to make the sorting  
23 comparisons. If the sort is not successfully completed, the results are  
24 unspecified.

25

26 [Note: For the default implementation, this method uses

1 **System.Array.Sort**, which uses the Quicksort algorithm. This is an  
2  $O(n \log n)$  operation, where  $n$  is the number of elements to sort.]

3 **Exceptions**

4  
5

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	$index < 0$ . -or- $count < 0$ .
<b>System.ArgumentException</b>	<b>System.Collections.ArrayList.Count</b> of the current instance - $index < count$ .
<b>System.InvalidCastException</b>	<i>comparer</i> is <b>null</b> , and one or more elements in the current instance do not implement the <b>System.IComparable</b> interface.
<b>System.NotSupportedException</b>	The current instance is read-only.

6  
7  
8

# 1 ArrayList.Sort(System.Collections.IComparer) Method

```
3 [ILASM]
4 .method public hidebysig virtual void Sort(class
5 System.Collections.IComparer comparer)
6
7 [C#]
8 public virtual void Sort(IComparer comparer)
```

## 8 Summary

9 Sorts the elements of current instance using the specified  
10 **System.Collections.IComparer**.

## 11 Parameters

12  
13

Parameter	Description
<i>comparer</i>	The <b>System.Collections.IComparer</b> implementation to use when comparing elements. Specify <b>null</b> to use the <b>System.IComparable</b> implementation of each element in the current instance.

14  
15

## 15 Description

## 16 Behaviors

17 As described above.

## 18 Default

19 If *comparer* is **null**, the **System.IComparable** implementation of  
20 each element in the current instance is used to make the sorting  
21 comparisons. If the sort is not successfully completed, the results are  
22 unspecified.

23  
24  
25  
26

[Note: For the default implementation, this method uses **System.Array.Sort**, which uses the Quicksort algorithm. This is an  $O(n \log n)$  operation, where  $n$  is the number of elements to sort.]

## 27 Exceptions

28  
29

Exception	Condition
<b>System.InvalidCastException</b>	<i>comparer</i> is <b>null</b> , and one or more elements in the current instance do not implement the

1  
2  
3

	<b>System.IComparable</b> interface.
<b>System.NotSupportedException</b>	The current instance is read-only.

# 1 ArrayList.Sort() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void Sort()  
4 [C#]  
5 public virtual void Sort()
```

## 6 Summary

7 Sorts the elements of the current instance.

## 8 Description

## 9 Behaviors

10 As described above.

## 11 Default

12 The **System.IComparable** implementation of each element in the  
13 current instance is used to make the sorting comparisons. If the sort is  
14 not successfully completed, the results are unspecified.

15  
16 [Note: For the default implementation, this method uses  
17 **System.Array.Sort**, which uses the Quicksort algorithm. This is an  
18  $O(n \log n)$  operation, where  $n$  is the number of elements to sort.]

## 19 Exceptions

20  
21

Exception	Condition
<b>System.InvalidCastException</b>	One or more elements in the current instance do not implement the <b>System.IComparable</b> interface.
<b>System.NotSupportedException</b>	The current instance read-only.

22  
23  
24

# 1 ArrayList.Synchronized(System.Collections 2 s.ArrayList) Method

```
3 [ILASM]  
4 .method public hidebysig static class  
5 System.Collections.ArrayList Synchronized(class  
6 System.Collections.ArrayList list)  
  
7 [C#]  
8 public static ArrayList Synchronized(ArrayList list)
```

## 9 Summary

10 Returns a **System.Collections.ArrayList** wrapper around the  
11 specified **System.Collections.ArrayList** that is synchronized (thread-  
12 safe).

## 13 Parameters

14  
15

Parameter	Description
<i>list</i>	The <b>System.Collections.ArrayList</b> to synchronize.

16  
17  
18

## 17 Return Value

19 A **System.Collections.ArrayList** wrapper that is synchronized  
20 (thread-safe).

## 21 Description

22 This method returns a thread-safe **System.Collections.ArrayList**  
23 that contains a reference to *list*. Any modifications of the elements in  
24 either the returned list or *list* will be reflected in the other.

25  
26 [Note: The **System.Collections.ArrayList.IsSynchronized** property  
27 of the new list is **true**. Every other property value of the new list  
28 references the same property value of *list*.

29  
30 By performing operations on the new list, this wrapper can be used to  
31 guarantee thread-safe access to the **System.Collections.ArrayList**  
32 *list*.]

## 33 Exceptions

34  
35

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

1  
2  
3

<b>System.ArgumentNullException</b>	<i>list</i> is <b>null</b> .
-------------------------------------	------------------------------

# 1 ArrayList.ToArray(System.Type) Method

```
2 [ILASM]  
3 .method public hidebysig virtual class System.Array  
4 ToArray(class System.Type type)  
  
5 [C#]  
6 public virtual Array ToArray(Type type)
```

## 7 Summary

8 Copies the elements of the current instance to a new array of the  
9 specified **System.Type**.

## 10 Parameters

Parameter	Description
<i>type</i>	The <b>System.Type</b> of the <b>System.Array</b> to create and copy the elements of the current instance.

## 14 Return Value

16 An array of **System.Type** *type* containing copies of the elements of  
17 the current instance.

## 18 Description

## 19 Behaviors

20 As described above.

## 21 Default

22 The elements are copied using **System.Array.Copy**.

23  
24 [Note: For the default implementation, this method is an  $O(n)$   
25 operation, where  $n$  is the **System.Collections.ArrayList.Count** of  
26 the current instance.]

## 27 Exceptions

Exception	Condition
<b>System.ArgumentNullException</b>	<i>type</i> is <b>null</b> .
<b>System.InvalidCastException</b>	At least one element of the current instance

1  
2  
3

cannot be cast to the **System.Type** type.

# 1 ArrayList.ToArray() Method

```
2 [ILASM]
3 .method public hidebysig virtual class System.Object[]
4 ToArray()
5
6 [C#]
7 public virtual object[] ToArray()
```

## 7 Summary

8 Copies the elements of the current instance to a new **System.Object**  
9 array.

## 10 Return Value

11

12 A **System.Object** array containing copies of the elements of the  
13 current instance.

## 14 Description

## 15 Behaviors

16 As described above.

## 17 Default

18 The elements are copied using **System.Array.Copy**.

19

20 [*Note:* For the default implementation, this method is an  $O(n)$   
21 operation, where  $n$  is the **System.Collections.ArrayList.Count** of  
22 the current instance.]

23

# 1 ArrayList.TrimToSize() Method

```
2 [ILASM]  
3 .method public hidebysig virtual void TrimToSize()  
4 [C#]  
5 public virtual void TrimToSize()
```

## 6 Summary

7 Sets the **System.Collections.ArrayList.Capacity** of the current  
8 instance to the **System.Collections.ArrayList.Count** of the current  
9 instance.

## 10 Description

11 [Note: This method can be used to minimize the memory overhead of  
12 the current instance if no new elements will be added to it.

13

14 To completely clear all elements from the current instance, call the  
15 **System.Collections.ArrayList.Clear** method before calling  
16 **System.Collections.ArrayList.TrimToSize.**]

## 17 Behaviors

18 As described above.

## 19 Default

20 If the **System.Collections.ArrayList.Count** of the current instance is  
21 zero, the **System.Collections.ArrayList.Capacity** of the current  
22 instance is set to the default initial capacity of 16.

## 23 Exceptions

24

25

Exception	Condition
<b>System.NotSupportedException</b>	The current instance is read-only or has a fixed size.

26

27

28

# 1 ArrayList.Capacity Property

```
2 [ILASM]
3 .property int32 Capacity { public hidebysig virtual
4 specialname int32 get_Capacity() public hidebysig virtual
5 specialname void set_Capacity(int32 value) }
6
7 [C#]
8 public virtual int Capacity { get; set; }
```

## 8 Summary

9 Gets or sets the number of elements that the current instance is  
10 capable of storing.

## 11 Property Value

12

13 A **System.Int32** that specifies the number of elements that the  
14 current instance is capable of storing.

## 15 Description

16 [*Note:* The **System.Collections.ArrayList.Capacity** of a  
17 **System.Collections.ArrayList** is the size of the internal array used  
18 to hold the elements of that list. When it is set, the internal array is  
19 reallocated to the specified value.]

## 20 Behaviors

21 As described above.

## 22 Default

23 If an attempt is made to set **System.Collections.ArrayList.Capacity**  
24 to a value less or equal to 0, it is set to the default capacity of 16.

25

26 If the **System.Collections.ArrayList.Count** of the current instance  
27 exceeds the **System.Collections.ArrayList.Capacity** of the current  
28 instance while adding elements to the current instance, the capacity of  
29 the list is doubled by automatically reallocating the internal array  
30 before copying the old elements and adding the new elements.

## 31 Exceptions

32

33

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<b>System.Collections.ArrayList.Capacity</b> is set to a value that is less than the

1  
2  
3

	<b>System.Collections.ArrayList.Count</b> of the current instance.
--	--

# 1 ArrayList.Count Property

```
2 [ILASM]
3 .property int32 ICollection.Count { public hidebysig
4 virtual abstract specialname int32 get_ICollection.Count()
5 }
6
7 [C#]
8 int ICollection.Count { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.Count.**]

12

# 1 ArrayList.Count Property

```
2 [ILASM]  
3 .property int32 Count { public hidebysig virtual  
4 specialname int32 get_Count() }  
5 [C#]  
6 public virtual int Count { get; }
```

## 7 Summary

8 Gets the number of elements contained in the current instance.

## 9 Property Value

10

11 A **System.Int32** that specifies the number of elements contained in  
12 the current instance.

## 13 Description

14 This property is read-only.

15

16 **System.Collections.ArrayList.Count** is the number of elements that  
17 are contained by the **System.Collections.ArrayList**. The count of a  
18 list is always less than or equal to  
19 **System.Collections.ArrayList.Capacity** of that list.

20

21 [*Note:* This property is implemented to support the  
22 **System.Collections.IList** interface.]

## 23 Behaviors

24 As described above.

## 25 Default

26 If the **System.Collections.ArrayList.Count** exceeds the  
27 **System.Collections.ArrayList.Capacity** of the current instance  
28 while adding elements to the current instance, the capacity of the list  
29 is doubled by automatically reallocating the internal array before  
30 copying the old elements and adding the new elements.

31

# 1 ArrayList.IsFixedSize Property

```
2 [ILASM]  
3 .property bool IList.IsFixedSize { public hidebysig virtual  
4 abstract specialname bool get_IList.IsFixedSize() }  
5  
6 [C#]  
7 bool IList.IsFixedSize { get; }
```

## 7 Summary

8 Implemented to support the **System.Collections.IList** interface.  
9 [Note: For more information, see  
10 **System.Collections.IList.IsFixedSize.**]

11

# 1 ArrayList.IsFixedSize Property

```
2 [ILASM]
3 .property bool IsFixedSize { public hidebysig virtual
4 specialname bool get_IsFixedSize() }
5
6 [C#]
7 public virtual bool IsFixedSize { get; }
```

## 7 Summary

8 Gets a **System.Boolean** indicating whether the  
9 **System.Collections.ArrayList.Capacity** of the current instance  
10 cannot be changed.

## 11 Property Value

12

13 **true** if the **System.Collections.ArrayList.Capacity** of the current  
14 instance cannot be changed; otherwise, **false**.

## 15 Description

16 This property is read-only.

17

18 [*Note:* Elements cannot be added or removed from a  
19 **System.Collections.ArrayList** with a fixed size, while existing  
20 elements can be modified.

21

22 This property is implemented to support the  
23 **System.Collections.IList** interface.]

## 24 Behaviors

25 As described above.

## 26 Default

27 The default value for this property is **false**.

28

# 1 ArrayList.IsReadOnly Property

```
2 [ILASM]
3 .property bool IList.IsReadOnly { public hidebysig virtual
4 abstract specialname bool get_IList.IsReadOnly() }

5 [C#]
6 bool IList.IsReadOnly { get; }
```

## 7 Summary

8 Implemented to support the **System.Collections.IList** interface.  
9 [Note: For more information, see  
10 **System.Collections.IList.IsReadOnly**.]

11

# 1 ArrayList.IsReadOnly Property

```
2 [ILASM]
3 .property bool IsReadOnly { public hidebysig virtual
4 specialname bool get_IsReadOnly() }
5
6 [C#]
7 public virtual bool IsReadOnly { get; }
```

## 7 Summary

8 Gets a value indicating whether the current instance is read-only.

## 9 Property Value

10

11 **true** if the current instance is read-only; otherwise, **false**.

## 12 Description

13 This property is read-only.

14

15 [*Note:* The elements of a **System.Collections.ArrayList** that is read-  
16 only cannot be modified, nor can elements be added to or removed  
17 from that list.

18

19 This property is implemented to support the  
20 **System.Collections.IList** interface.]

## 21 Behaviors

22 As described above.

## 23 Default

24 The default value of this property is **false**.

25

# 1 ArrayList.IsSynchronized Property

```
2 [ILASM]
3 .property bool ICollection.IsSynchronized { public
4 hidebysig virtual abstract specialname bool
5 get_ICollection.IsSynchronized() }
6
7 [C#]
8 bool ICollection.IsSynchronized { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.IsSynchronized.**]

12

# 1 ArrayList.IsSynchronized Property

```
2 [ILASM]
3 .property bool IsSynchronized { public hidebysig virtual
4 specialname bool get_IsSynchronized() }
5
6 [C#]
7 public virtual bool IsSynchronized { get; }
```

## 7 Summary

8 Gets a value indicating whether access to the current instance is  
9 synchronized (thread-safe).

## 10 Property Value

11

12 **true** if access to the current instance is synchronized (thread-safe);  
13 otherwise, **false**.

## 14 Description

15 This property is read-only.

16

17 To guarantee the thread safety of the **System.Collections.ArrayList**,  
18 all operations must be done through the wrapper returned by the  
19 **System.Collections.ArrayList.Synchronized** method.

20

21 [*Note:* This property is implemented to support the  
22 **System.Collections.IList** interface.]

## 23 Behaviors

24 As described above.

## 25 Default

26 The default value of this property is **false**.

27

# 1 ArrayList.Item Property

```
2 [ILASM]
3 .property object Item[int32 index] { public hidebysig
4 virtual specialname object get_Item(int32 index) public
5 hidebysig virtual specialname void set_Item(int32 index,
6 object value) }
7
8 [C#]
9 public virtual object this[int index] { get; set; }
```

## 9 Summary

10 Gets or sets the element at the specified index of the current instance.

## 11 Parameters

12  
13

Parameter	Description
<i>index</i>	A <b>System.Int32</b> that specifies the zero-based index of the element in the current instance to get or set. This value is greater than or equal to 0, and less than the <b>System.Collections.ArrayList.Count</b> of the current instance.

14  
15  
16

## 15 Property Value

17 The element at the specified index of the current instance.

## 18 Description

19 [Note: This property provides the ability to access a specific element in  
20 the collection by using the following syntax: myCollection[index].

21  
22 This property is implemented to support the  
23 **System.Collections.IList** interface.]

## 24 Behaviors

25 As described above.

## 26 Exceptions

27  
28

Exception	Condition
<b>System.ArgumentOutOfRangeException</b>	<i>index</i> < 0. -or-

1  
2  
3

	<i>index</i> >= <b>System.Collections.ArrayList.Count</b> of the current instance.
--	--

# 1 ArrayList.SyncRoot Property

```
2 [ILASM]
3 .property object ICollection.SyncRoot { public hideby sig
4 virtual abstract specialname object
5 get_ICollection.SyncRoot() }
6
7 [C#]
8 object ICollection.SyncRoot { get; }
```

## 8 Summary

9 Implemented to support the **System.Collections.ICollection**  
10 interface. [Note: For more information, see  
11 **System.Collections.ICollection.SyncRoot.**]

12

# 1 ArrayList.SyncRoot Property

```
2 [ILASM]
3 .property object SyncRoot { public hidebysig virtual
4 specialname object get_SyncRoot() }
5
6 [C#]
7 public virtual object SyncRoot { get; }
```

## 7 Summary

8 Gets an object that can be used to synchronize access to the current  
9 instance.

## 10 Property Value

11

12 A **System.Object** that can be used to synchronize access to the  
13 current instance.

## 14 Description

15 This property is read-only.

16

17 Program code must perform synchronized operations on the  
18 **System.Collections.ArrayList.SyncRoot** of the current instance, not  
19 directly on the current instance. This ensures proper operation of  
20 collections that are derived from other objects. Specifically, it  
21 maintains proper synchronization with other threads that might be  
22 simultaneously modifying the current instance.

## 23 Behaviors

24 As described above.

## 25 Default

26 This method returns a reference to the current instance.

27

28 [*Note:* This property is implemented to support the  
29 **System.Collections.IList** interface.]

30