

1 System.Collections.ArrayList Class

```
2 [ILAsm]  
3 .class public serializable ArrayList extends System.Object implements  
4 System.ICloneable, System.Collections.ICollection,  
5 System.Collections.IEnumerable, System.Collections.IList  
  
6 [C#]  
7 public class ArrayList: ICloneable, ICollection, IEnumerable, IList
```

8 Assembly Info:

- 9 • *Name:* mscorlib
- 10 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 11 • *Version:* 2.0.x.x
- 12 • *Attributes:*
 - 13 ○ CLSCompliantAttribute(true)

14 Type Attributes:

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

17 Implements:

- 18 • System.Collections.IList
- 19 • System.Collections.ICollection
- 20 • System.Collections.IEnumerable
- 21 • System.ICloneable

22 Summary

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

25 Inherits From: System.Object

26

27 **Library:** BCL

28

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

30

31 Description

32 System.Collections.ArrayList implements a variable-size
33 System.Collections.IList that uses an array of objects to store the elements. A
34 System.Collections.ArrayList has a System.Collections.ArrayList.Capacity,
35 which is the allocated length of the internal array. The total number of elements
36 contained by a list is its System.Collections.ArrayList.Count. As elements are added

1 to a list, its capacity is automatically increased as required by reallocating the internal
2 array.

3 **Example**

4 The following example shows how to create, initialize, and display the values of a
5 `System.Collections.ArrayList`.

6 [C#]

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

37 The output is

```
38
39 myAL
40
41 Count: 3
42
43 Capacity: 16
44
45 Values: Hello World !
```

46

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 `System.Collections.ArrayList` class
8 that is empty and has the default initial `System.Collections.ArrayList.Capacity`.

9 Description

10 The default initial `System.Collections.ArrayList.Capacity` of a
11 `System.Collections.ArrayList` is 16.

12

1 ArrayList(System.Int32) Constructor

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

6 Summary

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

9 Parameters

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

10 11 Description

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

14 Exceptions

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

15
16

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 `System.Collections.ArrayList` class
10 with the elements from the specified `System.Collections.ICollection`. The initial
11 `System.Collections.ArrayList.Count` and
12 `System.Collections.ArrayList.Capacity` of the new list are both equal to the number
13 of elements in the specified collection.

14 Parameters

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

15 Description

17 The elements in the new `System.Collections.ArrayList` instance are in the same
18 order as contained in *c*.

19 Exceptions

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

20
21

1 ArrayList.Adapter(System.Collections.IList)

2 Method

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

8 Summary

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

11 Parameters

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

12 Return Value

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

15 Description

16 This method returns a System.Collections.ArrayList that contains a reference to the
17 System.Collections.IList *list*. Any modifications to the elements in either the
18 returned list or *list* are reflected in the other.

19
20 [Note: The System.Collections.ArrayList class provides generic
21 System.Collections.ArrayList.Reverse,
22 System.Collections.ArrayList.BinarySearch and
23 System.Collections.ArrayList.Sort methods. This wrapper provides a means to use
24 those methods on System.Collections.IList *list* without implementing the methods
25 for the list. Performing these operations through the wrapper might be less efficient than
26 operations applied directly to the list.]
27
28

29 Exceptions

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

1

2

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

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

9

10 Return Value

11 A `System.Int32` that specifies the index of the current instance at which *value* has been
12 added.

13 Behaviors

14 As described above.

15

16 Exceptions

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

17

18

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30

ArrayList.AddRange(System.Collections.ICollection) Method

```
[ILAsm]  
.method public hidebysig virtual void AddRange(class  
System.Collections.ICollection c)  
  
[C#]  
public virtual void AddRange(ICollection c)
```

Summary

Adds the elements of the specified `System.Collections.ICollection` to the end of the current instance.

Parameters

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

Description

Behaviors

As described above.

Default

If the `System.Collections.ArrayList.Count` of the current instance plus the size of the collection being added is greater than the `System.Collections.ArrayList.Capacity` of the current instance, the capacity of the current instance is either doubled or increased to the new `System.Collections.ArrayList.Count`, whichever is greater. The internal array is reallocated to accommodate the new elements and the existing elements are copied to the new array before the new elements are added.

[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

1 be increased to accommodate the new elements, this method becomes an $O(n+m)$
2 operation, where n is the number of elements to be added and m is
3 `System.Collections.ArrayList.Count.`]

4
5

6 Exceptions

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

7
8

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

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

8 Summary

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

11 Parameters

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

12 13 Return Value

14 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 System.Collections.ArrayList.Count of the current instance.	<i>value</i> was not found, and <i>value</i> is greater than all elements in the current instance.

15
16 [Note: If *value* is not found and the current instance is already sorted, the bitwise
17 complement of the return value indicates the index in the current instance where *value*

1 would be found.]

2

3

4 **Description**

5 This method performs a binary search.

6

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

10

11

12 **Behaviors**

13 As described above.

14

15 **Default**

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

17

18 *value* is compared to elements in a binary search of the current instance until an
19 element with a value greater than or equal to *value* is found. If *comparer* is null, the
20 `System.IComparable` implementation of the element being compared -- or of *value* if
21 the element being compared does not implement the interface -- is used to make the
22 comparison. If *value* does not implement the `System.IComparable` interface and is
23 compared to an element that does not implement the interface,
24 `System.InvalidOperationException` is thrown. If the current instance is not already
25 sorted, correct results are not guaranteed.

26 [Note: For the default implementation, this method is an $O(\log_2 n)$ operation where *n* is
27 equal to the `System.Collections.ArrayList.Count` of the current instance.]

28

29

30 **Exceptions**

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

	interface.
--	------------

1

2

1 ArrayList.BinarySearch(System.Object)

2 Method

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

7 Summary

8 Searches the current instance for the specified System.Object.

9 Parameters

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

10 Return Value

12 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 System.Collections.ArrayList.Count of the current instance.	<i>value</i> was not found, and <i>value</i> is greater than all elements in the current instance.

13
14 [Note: If *value* is not found and the current instance is already sorted, the bitwise
15 complement of the return value indicates the index in the current instance where *value*
16 would be found.]

19 Description

1 This method performs a binary search.

2

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

6

7

8 Behaviors

9 As described above.

10

11 Default

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

13

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

21 [Note: For the default implementation, this method is an $O(\log_2 n)$ operation where n is
22 equal to the `System.Collections.ArrayList.Count` of the current instance.]

23

24

25 Exceptions

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

26

27

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 index, int32  
6 count, object value, class System.Collections.IComparer comparer)  
  
7 [C#]  
8 public virtual int BinarySearch(int index, int count, object value,  
9 IComparer comparer)
```

10 Summary

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

13 Parameters

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

14

15 Return Value

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	<i>value</i> was found.

instance.	
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 range of <i>index</i> to <i>index</i> + <i>count</i> - 1 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> - 1 in the current instance.

1
2
3
4
5
6

[*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* - 1.]

7 **Description**

8 This method performs a binary search.

9

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

13

14

15 **Behaviors**

16 As described above.

17

18 **Default**

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

20

21 *value* is compared to elements in a binary search of the range of *index* to *index* + *count*
22 - 1 in the current instance until an element with a value greater than or equal to *value* is
23 found or the end of the range is reached. If *comparer* is null, the `System.IComparable`
24 implementation of the element being compared -- or of *value* if the element being
25 compared does not implement the interface -- is used to make the comparison. If *value*
26 does not implement the `System.IComparable` interface and is compared to an element
27 that does not implement the interface, `System.InvalidOperationException` is thrown.
28 If the current instance is not already sorted, correct results are not guaranteed.

1 [Note: For the default implementation, this method is an $O(\log_2 count)$ operation.]

2

3

4 **Exceptions**

Exception	Condition
System.ArgumentException	System.Collections.ArrayList.Count of the current instance - $index < count$. -or- <i>comparer</i> is null, and <i>value</i> is not assignment-compatible with at least one element in the current instance.
System.ArgumentOutOfRangeException	$index < 0$. -or- $count < 0$.
System.InvalidOperationException	<i>comparer</i> is null, and both <i>value</i> and at least one element involved in the search of the current instance do not implement the System.IComparable interface.

5

6

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, depending upon the
8 element type.

9 Description

10 [Note: This method is implemented to support the System.Collections.IList
11 interface.]
12
13

14 Behaviors

15 Reference-type elements are set to null. Value-type elements are set to zero, except
16 for System.Boolean elements, which are set to false.
17

18 Default

19 This method uses System.Array.Clear to reset the values of the current instance.
20 System.Collections.ArrayList.Count is set to zero.
21 System.Collections.ArrayList.Capacity is not changed.
22

23 Usage

24 To reset the System.Collections.ArrayList.Capacity of the current instance, call
25 System.Collections.ArrayList.TrimToSize or set the
26 System.Collections.ArrayList.Capacity property directly.
27

28 Exceptions

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

System.NotSupportedException

The current instance is read-only or has a fixed size.

1

2

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

10 Description

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

14 Behaviors

15 As described above.
16

17 Default

18 This method uses `System.Array.Copy` to clone the current instance.
19

1 ArrayList.Contains(System.Object) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual bool Contains(object item)  
4 [C#]  
5 public virtual bool Contains(object item)
```

6 Summary

7 Determines whether the specified `System.Object` is contained in the current instance.

8 Parameters

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

9

10 Return Value

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

12 Description

13 [Note: This method is implemented to support the `System.Collections.IList`
14 interface.]
15
16

17 Behaviors

18 As described above.
19

20 Default

21 This method determines equality by calling the `System.Object.Equals` implementation
22 of the type of *item*.

23

24

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

- 1
- 2
- 3

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

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

10 Summary

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

13 Parameters

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

14

15 Behaviors

16 As described above.

17

18 Default

1 This method uses `System.Array.Copy` to copy the current instance to `array`.

2

3 **Exceptions**

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

4

5

1 ArrayList.CopyTo(System.Array) Method

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

6 Summary

7 Copies the elements from the current instance to the specified `System.Array`.

8 Parameters

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

9

10 Behaviors

11 As described above.

12

13 Default

14 This method uses `System.Array.Copy` to copy the current instance to *array*.

15

16 Exceptions

Exception	Condition
System.ArgumentNullException	<i>array</i> is null.
System.ArgumentException	<i>array</i> has more than one dimension.
	-or- <code>System.Collections.ArrayList.Count</code> of the current

	instance > <i>array</i> .Length.
System.InvalidCastException	At least one element in the current instance is not assignment-compatible with the type of <i>array</i> .

1

2

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

```
3 [ILAsm]  
4 .method public hidebysig virtual void CopyTo(class System.Array array,  
5 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 `System.Array`, starting
10 at the specified index of the array.

11 Parameters

Parameter	Description
<i>array</i>	The one-dimensional <code>System.Array</code> that is the destination of the elements copied from the current instance. The <code>System.Array.Length</code> of this array is greater than or equal to the sum of <i>arrayIndex</i> and the <code>System.Collections.ArrayList.Count</code> of the current instance.
<i>arrayIndex</i>	A <code>System.Int32</code> 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> .

12 13 Description

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

18 Behaviors

19 As described above.
20

21 Default

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

1 Exceptions

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

2

3

ArrayList.FixedSize(System.Collections.ArrayList) Method

```
[ILAsm]  
.method public hidebysig static class System.Collections.ArrayList  
FixedSize(class System.Collections.ArrayList list)  
  
[C#]  
public static ArrayList FixedSize(ArrayList list)
```

Summary

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

Parameters

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

Return Value

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

Description

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

[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*.

Adding to or removing from *list* will not throw an exception and is reflected in the returned list.

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

]

Exceptions

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

1

2

1 ArrayList.GetEnumerator() Method

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

7 Summary

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

9 Return Value

10 A System.Collections.IEnumerator for the current instance.

11 Description

12 If the the current instance is modified while an enumeration is in progress, a call to
13 System.Collections.IEnumerator.MoveNext or
14 System.Collections.IEnumerator.Reset throws
15 System.InvalidOperationException.
16

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

20 This property is implemented to support the System.Collections.IList interface.
21

22]

23 Behaviors

24 As described above.
25
26

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

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

8 Summary

9 Returns a System.Collections.IEnumerator for the specified section of the current
10 instance.

11 Parameters

Parameter	Description
<i>index</i>	A System.Int32 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 System.Collections.ArrayList.Count of the current instance.
<i>count</i>	A System.Int32 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 System.Collections.ArrayList.Count of the current instance minus <i>index</i> .

12 13 Return Value

14 A System.Collections.IEnumerator that can iterate over the range of *index* to *index* +
15 *count* - 1 in the current instance.

16 Description

17 The enumerator only enumerates over the range of the current instance from *index* to
18 *index* + *count* - 1. If the current instance is modified while an enumeration is in
19 progress, a call to System.Collections.IEnumerator.MoveNext or
20 System.Collections.IEnumerator.Reset will throw
21 System.InvalidOperationException.

22
23 [Note: For detailed information regarding the use of an enumerator, see
24 System.Collections.IEnumerator.]
25
26

27 Behaviors

1 As described above.

2

3 **Default**

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

7

8 If the elements of the current instance have not been modified while the enumeration
9 was in progress, a call to `System.Collections.IEnumerator.MoveNext` either returns
10 `true` and advances the enumerator one element in the current instance, or returns
11 `false` indicating the enumerator is at the end of the specified range.

12 **Exceptions**

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

13

14

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

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

8 Summary

9 Returns a `System.Collections.ArrayList` that represents the specified range of the
10 current instance.

11 Parameters

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

12

13 Return Value

14 A `System.Collections.ArrayList` that represents the range in the current instance
15 from *index* to *index* + *count* - 1.

16 Behaviors

17 As described above.

18

19 Default

20 This method does not create copies of the elements: the new
21 `System.Collections.ArrayList` instance is a view window into the source list.
22 Therefore, all subsequent changes to the source list must be done through this view
23 window `System.Collections.ArrayList`. If changes are made directly to the source

1 list, the view window list is invalidated and any operations on it throw
2 `System.InvalidOperationException`.

3

4 Exceptions

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

5

6

1 ArrayList.IndexOf(System.Object) Method

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

6 Summary

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

9 Parameters

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

11 Return Value

12 A System.Int32 that specifies the index of the first occurrence of *value* in the current
13 instance, if found; otherwise, -1.

14
15 [*Note:* This provides the caller with a standard code for a failed search.]
16
17

18 Description

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

23 Behaviors

24 As described above.
25

26 Default

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

1 [Note: For the default implementation, this method performs a linear search. On average,
2 this is an $O(n/2)$ operation, where n is *count*. The longest search is an $O(n)$ operation.]
3
4
5

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

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

8 Summary

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

11 Parameters

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

12 13 Return Value

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

16 [Note: This provides the caller with a standard code for a failed search.]
17
18
19

20 Description

21 Behaviors

22 As described above.
23

1 **Default**

2 This method uses `System.Array.IndexOf` to search the current instance for *value*.

3

4

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

7

8

9 **Exceptions**

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

10

11

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

```
3 [ILAsm]  
4 .method public hidebysig virtual int32 IndexOf(object value, int32  
5 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 occurrence of the specified
10 System.Object in the specified range.

11 Parameters

Parameter	Description
<i>value</i>	The System.Object to locate in current instance.
<i>startIndex</i>	A System.Int32 that specifies the index at which searching begins. This value is between 0 and the System.Collections.ArrayList.Count of the current instance minus 1, inclusive.

12

13 Return Value

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

17
18 [Note: This provides the caller with a standard code for a failed search.]
19
20

21 Description

22 Behaviors

23 As described above.
24

25 Default

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

1
2
3
4
5
6

[*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.]

7 **Exceptions**

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

8
9

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

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

7 Summary

8 Inserts the specified `System.Object` into the current instance at the specified index.

9 Parameters

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

10

11 Description

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

14

15

16 Behaviors

17 As described above.

18

19 Default

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

25

1 Exceptions

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

2

3

ArrayList.InsertRange(System.Int32, System.Collections.ICollection) Method

```
[ILAsm]
.method public hidebysig virtual void InsertRange(int32 index, class
System.Collections.ICollection c)

[C#]
public virtual void InsertRange(int index, ICollection c)
```

Summary

Inserts the elements of the specified `System.Collections.ICollection` at the specified index of the current instance.

Parameters

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

Behaviors

As described above.

Default

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

The order of the elements in the `System.Collections.ICollection c` is preserved in the current instance.

Exceptions

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

1

2

1 ArrayList.LastIndexOf(System.Object)

2 Method

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

7 Summary

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

10 Parameters

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

11 Return Value

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

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

19 Description

20 Behaviors

21 As described above.

23 Default

24 The ArrayList is searched backward starting at the last element and ending at the first
25 element.

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

28
29
30 [Note: For the default implementation, this method performs a linear search. On average,

1 this is an $O(n/2)$ operation, where n is `System.Collections.ArrayList.Count` of the
2 current instance. The longest search is an $O(n)$ operation.]

3

4

5

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

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

8 Summary

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

11 Parameters

Parameter	Description
<i>value</i>	The System.Object to locate in the current instance.
<i>startIndex</i>	A System.Int32 that specifies the index at which searching starts. This value is between 0 and the System.Collections.ArrayList.Count of the current instance - 1, inclusive.

12

13 Return Value

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

16

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

18

19

20 Description

21 Behaviors

22 As described above.

23

24 Default

25 The ArrayList is searched backward starting at *startIndex*.

26

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

1
2
3
4
5
6

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

7 **Exceptions**

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

8
9

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

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

8 Summary

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

11 Parameters

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

12 13 Return Value

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

17
18 [Note: This provides the caller with a standard code for a failed search.]
19
20

21 Description

22 Behaviors

23 As described above.
24

25 Default

1 The ArrayList is searched backward starting at *startIndex* and ending at *startIndex* -
2 *count* + 1.

3
4 This method uses `System.Array.LastIndexOf` to search the current instance for *value*.

5
6
7 [Note: For the default implementation, this method performs a linear search. On average,
8 this is an $O(count/2)$ operation. The longest search is an $O(count)$ operation.]

9
10

11 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> < 0. -or- <i>count</i> < 0. -or- <i>startIndex</i> >= System.Collections.ArrayList.Count of the current instance. -or- <i>count</i> >= System.Collections.ArrayList.Count of the current instance. -or- <i>count</i> > <i>startIndex</i> + 1.

12

13

1
2 **ArrayList.ReadOnly(System.Collections.Array**
3 **List) Method**

```
4 [ILAsm]  
5 .method public hidebysig static class System.Collections.ArrayList  
6 ReadOnly(class 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**

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

12
13 **Return Value**

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

15 **Description**

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

20
21 [Note: The System.Collections.ArrayList.IsReadOnly and
22 System.Collections.ArrayList.IsFixedSize properties of the new list are true.
23 Every other property value of the new list references the same property value of *list*.

24
25 By performing operations on the new list, this wrapper can be used to prevent additions
26 to, deletions from, and modifications of the System.Collections.ArrayList/*list*.

27]
28]

29 **Exceptions**

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

1

2

1 ArrayList.Remove(System.Object) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual void Remove(object obj)  
4 [C#]  
5 public virtual void Remove(object obj)
```

6 Summary

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

8 Parameters

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

9 10 Description

11 [Note: This method is implemented to support the `System.Collections.IList`
12 interface.]
13
14

15 Behaviors

16 As described above.

17

18 Default

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

20
21 If *obj* is found in the current instance, *obj* is removed from the current instance, the rest
22 of the elements are shifted down to fill the position vacated by *obj*, the
23 `System.Collections.ArrayList.Count` of the current instance is decreased by one,
24 and the position that was previously the last element in the current instance is set to
25 null. If *obj* is not found in the current instance, the current instance remains
26 unchanged.

27 [Note: For the default implementation, this method performs a linear search. On average,
28 this is an $O(n/2)$ operation, where n is `System.Collections.ArrayList.Count` of the
29 current instance. The longest search is an $O(n)$ operation.]
30
31

1 **Exceptions**

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

2

3

1 ArrayList.RemoveAt(System.Int32) Method

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

6 Summary

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

8 Parameters

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

9 10 Description

11 [Note: This method is implemented to support the System.Collections.IList
12 interface.]
13
14

15 Behaviors

16 As described above.
17

18 Default

19 The element at position *index* is removed from the current instance, the rest of the
20 elements are shifted down to fill the position vacated by that element, the
21 System.Collections.ArrayList.Count of the current instance is decreased by one,
22 and the position that was previously the last element in the current instance is set to
23 null.
24

25 Exceptions

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

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

1

2

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

```
3 [ILAsm]  
4 .method public hidebysig virtual void RemoveRange(int32 index, int32  
5 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

Parameter	Description
<i>index</i>	A <code>System.Int32</code> 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 <code>System.Collections.ArrayList.Count</code> of the current instance minus <i>count</i> , inclusive.
<i>count</i>	A <code>System.Int32</code> that specifies the number of elements to remove. This value is between 0 and the <code>System.Collections.ArrayList.Count</code> of the current instance minus <i>index</i> , inclusive.

11

12 Behaviors

13 As described above.

14

15 Default

16 The elements in the range of *index* to *index + count - 1* are removed from the current
17 instance, the rest of the elements are shifted down to fill the position vacated by those
18 elements, the `System.Collections.ArrayList.Count` of the current instance is
19 decreased by *count*, and the *count* positions that were previously the last elements in
20 the current instance are set to null.

21

22 Exceptions

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

1

2

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

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

8 Summary

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

11 Parameters

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

12 13 Return Value

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

16 Description

17 If *count* is less than the default initial capacity, 16, the
18 System.Collections.ArrayList.Capacity of the new
19 System.Collections.ArrayList instance is set to the default initial capacity.
20 Otherwise, the capacity is set to *count*. The System.Collections.ArrayList.Count of
21 the new instance is set to *count*.

22 Exceptions

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

23
24

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

11 Default

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

14

15 Exceptions

Exception	Condition
System.NotSupportedException	The current instance is read-only.

16

17

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

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

7 Summary

8 Reverses the sequence of the elements in the specified range of the current instance.

9 Parameters

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

10

11 Behaviors

12 As described above.

13

14 Default

15 This method uses `System.Array.Reverse` to modify the ordering of the current
16 instance.

17

18 Exceptions

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

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

1

2

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

```
3 [ILAsm]  
4 .method public hidebysig virtual void SetRange(int32 index, class  
5 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 to a range in
10 the current instance.

11 Parameters

Parameter	Description
<i>index</i>	A System.Int32 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 System.Collections.ArrayList.Count of the current instance minus <i>c.Count</i> , inclusive.
<i>c</i>	The System.Collections.ICollection whose elements to copy to the current instance.

12

13 Behaviors

14 As described above.

15

16 Default

17 This method uses the System.Collections.ICollection.CopyTo implementation of
18 System.Collections.ICollectionC.

19

20 Exceptions

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

System.ArgumentOutOfRangeException	<i>index</i> < 0. -or- System.Collections.ArrayList.Count of the current instance - <i>index</i> < <i>c.Count</i> .
System.ArgumentNullException	<i>c</i> is null.
System.NotSupportedException	The current instance is read-only.

1

2

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

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

8 Summary

9 Sorts the elements in the specified range of the current instance using the specified
10 System.Collections.IComparer implementation.

11 Parameters

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

12 13 Description

14 Behaviors

15 As described above.

16 17 Default

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

1
2
3
4
5
6
7

[*Note:* For the default implementation, this method uses `System.Array.Sort`, which uses the Quicksort algorithm. This is an $O(n \log_2 n)$ operation, where n is the number of elements to sort.]

8 Exceptions

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

9
10

1
2 **ArrayList.Sort(System.Collections.IComparer)**
3 **Method**

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

9 **Summary**

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

12 **Parameters**

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

13
14 **Description**

15 **Behaviors**

16 As described above.

17
18 **Default**

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

22
23
24 [Note: For the default implementation, this method uses System.Array.Sort, which uses
25 the Quicksort algorithm. This is an $O(n \log_2 n)$ operation, where n is the number of elements
26 to sort.]

27
28

1 Exceptions

Exception	Condition
System.InvalidOperationException	<i>comparer</i> is null, and one or more elements in the current instance do not implement the <code>System.IComparable</code> interface.
System.NotSupportedException	The current instance is read-only.

2

3

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 The `System.IComparable` implementation of each element in the current instance is
10 used to make the sorting comparisons.

11 Behaviors

12 As described above.

13

14 Default

15 If the sort is not successfully completed, the results are unspecified.

16

17

18 [*Note:* For the default implementation, this method uses `System.Array.Sort`, which uses
19 the Quicksort algorithm. This is an $O(n \log_2 n)$ operation, where n is the number of elements
20 to sort.]

21

22

23 Exceptions

Exception	Condition
System.NotSupportedException	The current instance is read-only.

24

25

ArrayList.Synchronized(System.Collections.ArrayList) Method

```
[ILAsm]
.method public hidebysig static class System.Collections.ArrayList
Synchronized(class System.Collections.ArrayList list)

[C#]
public static ArrayList Synchronized(ArrayList list)
```

Summary

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

Parameters

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

Return Value

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

Description

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

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

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

]

Exceptions

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

1

2

1 ArrayList.ToArray() Method

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

6 Summary

7 Copies the elements of the current instance to a new `System.Object` array.

8 Return Value

9 A `System.Object` array containing copies of the elements of the current instance.

10 Description

11 Behaviors

12 As described above.

13

14 Default

15 The elements are copied using `System.Array.Copy`.

16

17

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

20

21

22

1 ArrayList.ToArray(System.Type) Method

```
2 [ILAsm]  
3 .method public hidebysig virtual class System.Array ToArray(class  
4 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 specified
9 System.Type.

10 Parameters

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

11 Return Value

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

14 Description

15 Behaviors

16 As described above.

18 Default

19 The elements are copied using System.Array.Copy.

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

26 Exceptions

Exception	Condition
System.ArgumentNullException	<i>type</i> is null.
System.InvalidCastException	At least one element of the current instance cannot be cast to the <code>System.Type</code> <i>type</i> .

1

2

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 instance to the
8 `System.Collections.ArrayList.Count` of the current instance.

9 Description

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

12
13 To completely clear all elements from the current instance, call the
14 `System.Collections.ArrayList.Clear` method before calling
15 `System.Collections.ArrayList.TrimToSize`.

16
17]

18 Behaviors

19 As described above.

20

21 Default

22 If the `System.Collections.ArrayList.Count` of the current instance is zero, the
23 `System.Collections.ArrayList.Capacity` of the current instance is set to the default
24 initial capacity of 16.

25

26 Exceptions

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

27

28

1 ArrayList.Capacity Property

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

8 Summary

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

10 Property Value

11 A System.Int32 that specifies the number of elements that the current instance is
12 capable of storing.

13 Description

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

20 Behaviors

21 As described above.
22

23 Default

24 If an attempt is made to set System.Collections.ArrayList.Capacity to a value less
25 or equal to 0, it is set to the default capacity of 16.
26

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

32 Exceptions

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

System.ArgumentOutOfRangeException

System.Collections.ArrayList.Capacity is set to a value that is less than the System.Collections.ArrayList.Count of the current instance.

1

2

1 ArrayList.Count Property

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

7 Summary

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

10

1 ArrayList.Count Property

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

7 Summary

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

9 Property Value

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

12 Description

13 This property is read-only.

14
15 System.Collections.ArrayList.Count is the number of elements that are contained
16 by the System.Collections.ArrayList. The count of a list is always less than or equal
17 to System.Collections.ArrayList.Capacity of that list.

18
19 *[Note: This property is implemented to support the System.Collections.IList*
20 *interface.]*
21
22

23 Behaviors

24 As described above.

26 Default

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

32

1 **ArrayList.IsFixedSize Property**

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

7 **Summary**

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

10 **Property Value**

11 true if the `System.Collections.ArrayList.Capacity` of the current instance cannot
12 be changed; otherwise, false.

13 **Description**

14 This property is read-only.

15
16 [*Note:* Elements cannot be added or removed from a `System.Collections.ArrayList`
17 with a fixed size, while existing elements can be modified.

18
19 An attempt to add to or remove from a fixed size `ArrayList` will throw
20 `System.NotSupportedException`. However, the size of a fixed size `ArrayList` will change
21 to reflect the additions or removals from the `ArrayList` that was used to create the fixed
22 size `ArrayList`.

23
24 This property is implemented to support the `System.Collections.IList` interface.

25
26]

27 **Behaviors**

28 As described above.

29

30 **Default**

31 The default value for this property is `false`.

32

33

1 ArrayList.IsFixedSize Property

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

7 Summary

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

10

1 ArrayList.IsReadOnly Property

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

7 Summary

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

9 Property Value

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

11 Description

12 This property is read-only.

13
14 [*Note:* The elements of a `System.Collections.ArrayList` that is read-only cannot be
15 modified, nor can elements be added to or removed from that list.

16
17 An attempt to add to, remove from, or modify a read-only `ArrayList` will throw
18 `System.NotSupportedException`. However, changes to the `ArrayList` that was used to
19 create the read-only `ArrayList` are reflected in the read-only `ArrayList`.

20
21 This property is implemented to support the `System.Collections.IList` interface.

22
23]

24 Behaviors

25 As described above.

26

27 Default

28 The default value of this property is `false`.

29

30

1 ArrayList.IsReadOnly Property

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

7 Summary

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

10

1 ArrayList.IsSynchronized Property

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

7 Summary

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

10 Property Value

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

12 Description

13 This property is read-only.

14
15 To guarantee the thread safety of the `System.Collections.ArrayList`, all operations
16 must be done through the wrapper returned by the
17 `System.Collections.ArrayList.Synchronized` method.

18
19 [*Note:* This property is implemented to support the `System.Collections.IList`
20 interface.]
21
22

23 Behaviors

24 As described above.

25

26 Default

27 The default value of this property is false.

28

29

1 ArrayList.IsSynchronized Property

```
2 [ILAsm]  
3 .property bool ICollection.IsSynchronized { public hidebysig virtual  
4 abstract specialname bool get_ICollection.IsSynchronized() }
```

```
5 [C#]  
6 bool ICollection.IsSynchronized { get; }
```

7 Summary

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

10

1 ArrayList.Item Property

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

8 Summary

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

10 Parameters

Parameter	Description
<i>index</i>	A <code>System.Int32</code> 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 <code>System.Collections.ArrayList.Count</code> of the current instance.

11 12 Property Value

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

14 Description

15 [*Note:* This property provides the ability to access a specific element in the collection by
16 using the following syntax: `myCollection[index]`.
17

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

21 Behaviors

22 As described above.
23

24 Exceptions

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

-or-

index > =
System.Collections.ArrayList.Count of the
current instance.

1

2

1 ArrayList.SyncRoot Property

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

7 Summary

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

9 Property Value

10 A `System.Object` that can be used to synchronize access to the current instance.

11 Description

12 This property is read-only.

13
14 Program code must perform synchronized operations on the
15 `System.Collections.ArrayList.SyncRoot` of the current instance, not directly on the
16 current instance. This ensures proper operation of collections that are derived from
17 other objects. Specifically, it maintains proper synchronization with other threads that
18 might be simultaneously modifying the current instance.

19 Behaviors

20 As described above.

21

22 Default

23 This method returns a reference to the current instance.

24

25

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

27

28

29

1 ArrayList.SyncRoot Property

```
2 [ILAsm]  
3 .property object ICollection.SyncRoot { public hidebysig virtual abstract  
4 specialname object get_ICollection.SyncRoot() }  
  
5 [C#]  
6 object ICollection.SyncRoot { get; }
```

7 Summary

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

10