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 04 00 00 00 00 00 00 00]
- *Version:* 2.0.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 `System.Collections.ArrayList.Capacity`, which is the allocated length of the internal array. The total number of elements contained by a list is its `System.Collections.ArrayList.Count`. As elements are added to a list, its capacity is automatically increased as required by reallocating the internal array.

Example

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

[C#]

```
using System;
using System.Collections;

public class SamplesArrayList {

  public static void Main() {

  // Create and initialize a new ArrayList.
  ArrayList myAL = new ArrayList();
  myAL.Add("Hello");
  myAL.Add("World");
  myAL.Add("!");

  // Display the properties and values of the ArrayList.
  Console.WriteLine( "myAL" );
  Console.WriteLine( "Count: {0}", myAL.Count );
  Console.WriteLine( "Capacity: {0}", myAL.Capacity );
  Console.Write( "Values:" );
  PrintValues( myAL );
  }

public static void PrintValues( IEnumerable myList ) {

  IEnumerator myEnumerator = myList.GetEnumerator();
  while ( myEnumerator.MoveNext() )
    Console.Write( " {0}", myEnumerator.Current );
    Console.WriteLine();
  }
}
```

The output is

```
myAL

Count: 3

Capacity: 16

Values: Hello World !
```

ArrayList() Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor()

[C#]
public ArrayList()
```

Summary

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

Description

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

ArrayList(System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(int32 capacity)

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

Summary

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

Parameters

Parameter	Description
capacity	A System.Int32 that specifies the number of elements that the new instance is initially capable of storing.

Description

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

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*capacity* < 0.

ArrayList(System.Collections.ICollection) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.Collections.ICollection c)

[C#]
public ArrayList(ICollection c)
```

Summary

Constructs and initializes a new instance of the `System.Collections.ArrayList` class with the elements from the specified `System.Collections.ICollection`. The initial `System.Collections.ArrayList.Count` and `System.Collections.ArrayList.Capacity` of the new list are both equal to the number of elements in the specified collection.

Parameters

Parameter	Description
c	The `System.Collections.ICollection` whose elements are copied to the new list.

Description

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

Exceptions

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

ArrayList.Adapter(System.Collections.IList) Method

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

[C#]
public static ArrayList Adapter(IList list)
```

Summary

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

Parameters

Parameter	Description
list	The `System.Collections.IList` to wrap.

Return Value

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

Description

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

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

Exceptions

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

ArrayList.Add(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual int32 Add(object value)

[C#]
public virtual int Add(object value)
```

Summary

Adds the specified `System.Object` to the end of the current instance.

Parameters

Parameter	Description
value	The `System.Object` to be added to the end of the current instance.

Return Value

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

Behaviors

As described above.

Exceptions

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

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
c	The `System.Collections.ICollection` 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 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`.]

Exceptions

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

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

```
[ILAsm]
.method public hidebysig virtual int32 BinarySearch(object value, class
System.Collections.IComparer comparer)

[C#]
public virtual int BinarySearch(object value, IComparer comparer)
```

Summary

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

Parameters

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

Return Value

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

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

[*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.]

Description

This method performs a binary search.

[*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.]

Behaviors

As described above.

Default

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

value is compared to elements in a binary search of the current instance until an element with a value greater than or equal to *value* is found. 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, `System.InvalidOperationException` 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($\log_2 n$) operation where *n* is equal to the `System.Collections.ArrayList.Count` of the current instance.]

Exceptions

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

	interface.

ArrayList.BinarySearch(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual int32 BinarySearch(object value)

[C#]
public virtual int BinarySearch(object value)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` for which to search.

Return Value

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

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

[*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.]

Description

This method performs a binary search.

[*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.]

Behaviors

As described above.

Default

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

value is compared to elements in a binary search of the current instance until an element with a value greater than or equal to *value* is found. 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, `System.InvalidOperationException` 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($\log_2 n$) operation where *n* is equal to the `System.Collections.ArrayList.Count` of the current instance.]

Exceptions

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

ArrayList.BinarySearch(System.Int32, System.Int32, System.Object, System.Collections.IComparer) Method

```
[ILAsm]
.method public hidebysig virtual int32 BinarySearch(int32 index, int32
count, object value, class System.Collections.IComparer comparer)

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

Summary

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

Parameters

Parameter	Description
index	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.
count	A `System.Int32` that specifies the number of elements to search, beginning with *index*. 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 *index*.
value	The `System.Object` for which to search.
comparer	The `System.Collections.IComparer` implementation to use when comparing elements. Specify `null` to use the `System.IComparable` implementation of each element.

Return Value

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

Return Value	Description
The index of *value* in the current	*value* was found.

instance.	
The bitwise complement of the index of the first element that is greater than *value*.	*value* was not found, and at least one element in the range of *index* to *index* + *count* - 1 in the current instance is greater than *value*.
The bitwise complement of (*index* + *count*)	*value* was not found, and *value* is greater than all elements in the range of *index* to *index* + *count* - 1 in the current instance.

[*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.]

Description

This method performs a binary search.

[*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.]

Behaviors

As described above.

Default

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* - 1 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, `System.InvalidOperationException` 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($\log_2$*count*) operation.]

Exceptions

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

ArrayList.Clear() Method

```
[ILAsm]
.method public hidebysig virtual void Clear()

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

Summary

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

Description

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

Behaviors

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

Default

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

Usage

To reset the `System.Collections.ArrayList.Capacity` of the current instance, call `System.Collections.ArrayList.TrimToSize` or set the `System.Collections.ArrayList.Capacity` property directly.

Exceptions

Exception	Condition

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

ArrayList.Clone() Method

[ILAsm]
```
.method public hidebysig virtual object Clone()
```

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

Summary

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

Return Value

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

Description

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

Behaviors

As described above.

Default

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

ArrayList.Contains(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual bool Contains(object item)

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

Summary

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

Parameters

Parameter	Description
item	The `System.Object` to locate in the current instance.

Return Value

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

Description

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

Behaviors

As described above.

Default

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

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

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

```
[ILAsm]
.method public hidebysig virtual void CopyTo(int32 index, class
System.Array array, int32 arrayIndex, int32 count)

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

Summary

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

Parameters

Parameter	Description
index	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.
array	The one-dimensional `System.Array` that is the destination of the elements copied from the current instance.
arrayIndex	A `System.Int32` that specifies the first index of *array* to which the elements of the current instance are copied. This value is greater than or equal to zero, and less than *array*.Length minus *count*.
count	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 *index* and *array*.Length minus *arrayIndex*.

Behaviors

As described above.

Default

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

Exceptions

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

ArrayList.CopyTo(System.Array) Method

```
[ILAsm]
.method public hidebysig virtual void CopyTo(class System.Array array)

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

Summary

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

Parameters

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

Behaviors

As described above.

Default

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

Exceptions

Exception	Condition
System.ArgumentNullException	*array* is `null`.
System.ArgumentException	*array* has more than one dimension. -or- `System.Collections.ArrayList.Count` of the current

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

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

```
[ILAsm]
.method public hidebysig virtual void CopyTo(class System.Array array,
int32 arrayIndex)

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

Summary

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

Parameters

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

Description

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

Behaviors

As described above.

Default

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

1 **Exceptions**

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

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
list	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	*list* is `null`.

ArrayList.GetEnumerator() Method

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

[C#]
public virtual IEnumerator GetEnumerator()
```

Summary

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

Return Value

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

Description

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

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

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

]

Behaviors

As described above.

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

```
[ILAsm]
.method public hidebysig virtual class System.Collections.IEnumerator
GetEnumerator(int32 index, int32 count)

[C#]
public virtual IEnumerator GetEnumerator(int index, int count)
```

Summary

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

Parameters

Parameter	Description
index	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.
count	A `System.Int32` that specifies the number of elements, beginning with *index*, 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 *index*.

Return Value

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

Description

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

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

Behaviors

As described above.

Default

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

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

Exceptions

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

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

```
[ILAsm]
.method public hidebysig virtual class System.Collections.ArrayList
GetRange(int32 index, int32 count)

[C#]
public virtual ArrayList GetRange(int index, int count)
```

Summary

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

Parameters

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

Return Value

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

Behaviors

As described above.

Default

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

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* < 0. -or- *count* < 0.
System.ArgumentException	`System.Collections.ArrayList.Count` of the current instance - *index* < *count*.

ArrayList.IndexOf(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual int32 IndexOf(object value)

[C#]
public virtual int IndexOf(object value)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` to locate in the current instance.

Return Value

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

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

Description

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

Behaviors

As described above.

Default

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

[*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.]

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

```
[ILAsm]
.method public hidebysig virtual int32 IndexOf(object value, int32
startIndex, int32 count)

[C#]
public virtual int IndexOf(object value, int startIndex, int count)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` to locate in current instance.
startIndex	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.
count	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 *startIndex*, inclusive.

Return Value

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

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

Description

Behaviors

As described above.

Default

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

[*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**
System.ArgumentOutOfRangeException	*startIndex*>= `System.Collections.ArrayList.Count` of the current instance. -or- *count* < 0. -or- *count* >`System.Collections.ArrayList.Count` of the current instance - *startIndex*.

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

```
[ILAsm]
.method public hidebysig virtual int32 IndexOf(object value, int32
startIndex)

[C#]
public virtual int IndexOf(object value, int startIndex)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` to locate in current instance.
startIndex	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.

Return Value

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

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

Description

Behaviors

As described above.

Default

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

[*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**
System.ArgumentOutOfRangeException	*startIndex* < 0. -or- *startIndex* >= `System.Collections.ArrayList.Count` of the current instance.

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

```
[ILAsm]
.method public hidebysig virtual void Insert(int32 index, object value)

[C#]
public virtual void Insert(int index, object value)
```

Summary

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

Parameters

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

Description

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

Behaviors

As described above.

Default

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

Exceptions

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

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
index	A `System.Int32` that specifies the index in the current instance at which the new elements are inserted. This value is between 0 and the `System.Collections.ArrayList.Count` of the current instance, inclusive.
c	The `System.Collections.ICollection` 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	*c* is `null`.
System.ArgumentOutOfRangeException	*index* < 0. *index* > `System.Collections.ArrayList.Count` of the current instance.
System.NotSupportedException	The current instance is read-only or has a fixed size.

ArrayList.LastIndexOf(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual int32 LastIndexOf(object value)

[C#]
public virtual int LastIndexOf(object value)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` to locate in the current instance.

Return Value

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

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

Description

Behaviors

As described above.

Default

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

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

[*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.]

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

```
[ILAsm]
.method public hidebysig virtual int32 LastIndexOf(object value, int32
startIndex)

[C#]
public virtual int LastIndexOf(object value, int startIndex)
```

Summary

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

Parameters

Parameter	Description
value	The `System.Object` to locate in the current instance.
startIndex	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.

Return Value

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

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

Description

Behaviors

As described above.

Default

The ArrayList is searched backward starting at *startIndex*.

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

[*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.]

Exceptions

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

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

```
[ILAsm]
.method public hidebysig virtual int32 LastIndexOf(object value, int32
startIndex, int32 count)

[C#]
public virtual int LastIndexOf(object value, int startIndex, int count)
```

Summary

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

Parameters

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

Return Value

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

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

Description

Behaviors

As described above.

Default

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

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

[*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.]

Exceptions

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

ArrayList.ReadOnly(System.Collections.ArrayList) Method

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

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

Summary

Returns a read-only `System.Collections.ArrayList` wrapper.

Parameters

Parameter	Description
list	The `System.Collections.ArrayList` to wrap.

Return Value

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

Description

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

[*Note:* The `System.Collections.ArrayList.IsReadOnly` and `System.Collections.ArrayList.IsFixedSize` properties of the new list are `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 prevent additions to, deletions from, and modifications of the `System.Collections.ArrayList`*list*.

]

Exceptions

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

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
obj	The `System.Object` 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 *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

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

ArrayList.RemoveAt(System.Int32) Method

```
[ILAsm]
.method public hidebysig virtual void RemoveAt(int32 index)

[C#]
public virtual void RemoveAt(int index)
```

Summary

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

Parameters

Parameter	Description
index	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.

Description

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

Behaviors

As described above.

Default

The element at position *index* is removed from the current instance, the rest of the elements are shifted down to fill the position vacated by that element, 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`.

Exceptions

Exception	Condition

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

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

```
[ILAsm]
.method public hidebysig virtual void RemoveRange(int32 index, int32
count)

[C#]
public virtual void RemoveRange(int index, int count)
```

Summary

Removes the specified range of elements from the current instance.

Parameters

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

Behaviors

As described above.

Default

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

Exceptions

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

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

```
[ILAsm]
.method public hidebysig static class System.Collections.ArrayList
Repeat(object value, int32 count)

[C#]
public static ArrayList Repeat(object value, int count)
```

Summary

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

Parameters

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

Return Value

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

Description

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

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*count* < 0.

ArrayList.Reverse() Method

[ILAsm]
```
.method public hidebysig virtual void Reverse()
```

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

Summary

Reverses the sequence of the elements in the current instance.

Behaviors

As described above.

Default

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

Exceptions

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

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

[ILAsm]
```
.method public hidebysig virtual void Reverse(int32 index, int32 count)
```

[C#]
```
public virtual void Reverse(int index, int count)
```

Summary

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

Parameters

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

Behaviors

As described above.

Default

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

Exceptions

Exception	Condition

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

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

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

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

Summary

Copies the elements of the specified `System.Collections.ICollection` to a range in the current instance.

Parameters

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

Behaviors

As described above.

Default

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

Exceptions

Exception	Condition

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

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

```
[ILAsm]
.method public hidebysig virtual void Sort(int32 index, int32 count, class
System.Collections.IComparer comparer)

[C#]
public virtual void Sort(int index, int count, IComparer comparer)
```

Summary

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

Parameters

Parameter	Description
index	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 *count*, inclusive.
count	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 *index*, inclusive.
comparer	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.

Description

Behaviors

As described above.

Default

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

[*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.]

Exceptions

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

ArrayList.Sort(System.Collections.IComparer) Method

```
[ILAsm]
.method public hidebysig virtual void Sort(class
System.Collections.IComparer comparer)

[C#]
public virtual void Sort(IComparer comparer)
```

Summary

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

Parameters

Parameter	Description
comparer	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.

Description

Behaviors

As described above.

Default

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

[*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.]

Exceptions

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

ArrayList.Sort() Method

[ILAsm]
```
.method public hidebysig virtual void Sort()
```

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

Summary

Sorts the elements of the current instance.

Description

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

Behaviors

As described above.

Default

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

[*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.]

Exceptions

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

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
list	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	*list* is `null`.

ArrayList.ToArray() Method

```
[ILAsm]
.method public hidebysig virtual object[] ToArray()

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

Summary

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

Return Value

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

Description

Behaviors

As described above.

Default

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

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

ArrayList.ToArray(System.Type) Method

```
[ILAsm]
.method public hidebysig virtual class System.Array ToArray(class
System.Type type)

[C#]
public virtual Array ToArray(Type type)
```

Summary

Copies the elements of the current instance to a new array of the specified `System.Type`.

Parameters

Parameter	Description
type	The `System.Type` of the `System.Array` to create and copy the elements of the current instance.

Return Value

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

Description

Behaviors

As described above.

Default

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

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

Exceptions

Exception	**Condition**
System.ArgumentNullException	*type* is `null`.
System.InvalidCastException	At least one element of the current instance cannot be cast to the `System.Type`*type*.

ArrayList.TrimToSize() Method

```
[ILAsm]
.method public hidebysig virtual void TrimToSize()

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

Summary

Sets the `System.Collections.ArrayList.Capacity` of the current instance to the `System.Collections.ArrayList.Count` of the current instance.

Description

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

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

]

Behaviors

As described above.

Default

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

Exceptions

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

ArrayList.Capacity Property

```
[ILAsm]
.property int32 Capacity { public hidebysig virtual specialname int32
get_Capacity() public hidebysig virtual specialname void
set_Capacity(int32 value) }

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

Summary

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

Property Value

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

Description

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

Behaviors

As described above.

Default

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

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

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.

ArrayList.Count Property

[ILAsm]
```
.property int32 ICollection.Count { public hidebysig virtual abstract
specialname int32 get_ICollection.Count() }
```

[C#]
```
int ICollection.Count { get; }
```

Summary

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

ArrayList.Count Property

[ILAsm]
```
.property int32 Count { public hidebysig virtual specialname int32
get_Count() }
```

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

Summary

Gets the number of elements contained in the current instance.

Property Value

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

Description

This property is read-only.

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

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

Behaviors

As described above.

Default

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

ArrayList.IsFixedSize Property

[ILAsm]
```
.property bool IsFixedSize { public hidebysig virtual specialname bool
get_IsFixedSize() }
```

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

Summary

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

Property Value

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

Description

This property is read-only.

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

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

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

]

Behaviors

As described above.

Default

The default value for this property is `false`.

ArrayList.IsFixedSize Property

[ILAsm]
```
.property bool IList.IsFixedSize { public hidebysig virtual abstract
specialname bool get_IList.IsFixedSize() }
```

[C#]
```
bool IList.IsFixedSize { get; }
```

Summary

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

ArrayList.IsReadOnly Property

[ILAsm]
```
.property bool IsReadOnly { public hidebysig virtual specialname bool
get_IsReadOnly() }
```

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

Summary

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

Property Value

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

Description

This property is read-only.

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

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

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

]

Behaviors

As described above.

Default

The default value of this property is `false`.

ArrayList.IsReadOnly Property

[ILAsm]
```
.property bool IList.IsReadOnly { public hidebysig virtual abstract
specialname bool get_IList.IsReadOnly() }
```

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

Summary

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

ArrayList.IsSynchronized Property

```
[ILAsm]
.property bool IsSynchronized { public hidebysig virtual specialname bool
get_IsSynchronized() }

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

Summary

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

Property Value

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

Description

This property is read-only.

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

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

Behaviors

As described above.

Default

The default value of this property is `false`.

ArrayList.IsSynchronized Property

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

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

Summary

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

ArrayList.Item Property

[ILAsm]
```
.property object Item[int32 index] { public hidebysig virtual specialname
object get_Item(int32 index) public hidebysig virtual specialname void
set_Item(int32 index, object value) }
```

[C#]
```
public virtual object this[int index] { get; set; }
```

Summary

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

Parameters

Parameter	Description
index	A `System.Int32` 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 `System.Collections.ArrayList.Count` of the current instance.

Property Value

The element at the specified index of the current instance.

Description

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

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

]

Behaviors

As described above.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* < 0.

	-or- *index* >= `System.Collections.ArrayList.Count` of the current instance.

ArrayList.SyncRoot Property

```
[ILAsm]
.property object SyncRoot { public hidebysig virtual specialname object
get_SyncRoot() }

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

Summary

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

Property Value

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

Description

This property is read-only.

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

Behaviors

As described above.

Default

This method returns a reference to the current instance.

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

ArrayList.SyncRoot Property

[ILAsm]
```
.property object ICollection.SyncRoot { public hidebysig virtual abstract
specialname object get_ICollection.SyncRoot() }
```

[C#]
```
object ICollection.SyncRoot { get; }
```

Summary

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