System.Collections.Generic.IList<T> Interface

```
[ILAsm]
.class interface public abstract IList`1<T> implements
System.Collections.Generic.ICollection`1<T>,
System.Collections.Generic.IEnumerable`1<T>

[C#]
public interface IList<T>: ICollection<T>, IEnumerable<T>
```

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)

Implements:

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

Summary

Represents a collection of objects that can be individually accessed by index.

Library: BCL

Description

This interface is a descendant of the `System.Collections.Generic.ICollection<T>` interface and is the base interface of all generic lists.

IList<T>.IndexOf(T) Method

```
[ILAsm]
.method public hidebysig virtual abstract int32 IndexOf(!0 value)

[C#]
int IndexOf(T value)
```

Summary

Determines the index of a specific item in the current instance.

Parameters

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

Return Value

The index of *value* if found in the current instance; otherwise, -1.

Description

Implementations can vary in how they determine equality of objects; for example, `System.Collections.Generic.List<T>` uses the default comparer, whereas, `System.Collections.Generic.Dictionary<T,U>` allows the user to specify the `System.Collections.Generic.IComparer<T>` implementation to use for comparing keys.

IList<T>.Insert(System.Int32, T) Method

[ILAsm]
```
.method public hidebysig virtual abstract void Insert(int32 index, !0
value)
```

[C#]
```
void Insert(int index, T value)
```

Summary

Inserts an item into the current instance at the specified position.

Parameters

Parameter	Description
index	A `System.Int32` that specifies the zero-based index at which value is inserted.
value	The `T` to insert into the current instance.

Description

In collections of contiguous elements, such as lists, the elements that follow the insertion point have indices one more than previously, to accommodate the new element. If the collection is indexed, the indexes of the elements that are moved are also updated.

Behaviors

If *index* equals the number of items in the `System.Collections.Generic.IList<T>`, then value is required to be appended to the end of the current instance.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* is not a valid index in the current instance (i.e. is negative or greater than the number of elements in the current instance).
System.NotSupportedException	The current instance is read-only.

1

IList<T>.RemoveAt(System.Int32) Method

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

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

Summary

Removes the item at the specified index of the current instance.

Parameters

Parameter	Description
index	A `System.Int32` that specifies the zero-based index of the item to remove.

Description

In collections of contiguous elements, such as lists, the elements that follow the removed element have indices one less than previously. If the collection is indexed, the indexes of the elements that are moved are also updated.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* is not a valid index in the current instance.
System.NotSupportedException	The current instance is read-only.

IList<T>.Item Property

[ILAsm]
```
.property !0 Item[int32 index] { public hidebysig virtual abstract
specialname !0 get_Item(int32 index) public hidebysig virtual abstract
specialname void set_Item(int32 index, !0 value) }
```

[C#]
```
T this[int index] { get; set; }
```

Summary

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

Parameters

Parameter	Description
index	The zero-based index of the element to get or set.

Property Value

The element at the specified index in the current instance.

Description

This property provides the ability to access a specific element in the collection by using some language-specific syntax.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*index* is not a valid index in the current instance.
System.NotSupportedException	The property is being set and the current instance is read-only.