System.IComparable<-T> Interface

```
[ILAsm]
.class interface public abstract System.IComparable`1<-T>

[C#]
public interface IComparable<in T>
```

Assembly Info:

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

Summary

Defines a generalized comparison method that a value type or class implements to create a type-specific comparison method for ordering instances.

Library: BCL

Description

This interface is implemented by types whose values can be ordered; for example, the numeric and string classes. A value type or class implements the System.IComparable`1<T>.CompareTo method to create a type-specific comparison method suitable for purposes such as sorting.

The System.IComparable`1<T> interface defines the System.IComparable`1<T>.CompareTo method, which determines the sort order of instances of the implementing type. The System.IEquatable`1<T> interface defines the System.IEquatable`1<T>.Equals method, which determines the equality of instances of the implementing type.

The implementation of the System.IComparable`1<T>.CompareTo method must return an System.Int32 that has one of three values, as shown in the following table.

Value	Meaning
Less than zero	This object is less than the object specified by the System.IComparable`1<T>.CompareTo method.
Zero	This object is equal to the method parameter.
Greater than	This object is greater than the method parameter.

zero	

The ``System.IComparable`1<T>`` interface provides a strongly typed comparison method for ordering members of a generic collection object. Because of this, it is usually not called directly from developer code. Instead, it is called automatically by methods such as ``System.Collections.Generic.List`1<T>.Sort`` and ``System.Collections.Generic.SortedList`2<T1,T2>.Add``.

Behaviors

Replace the type parameter of the ``System.IComparable`1<T>`` interface with the type that is implementing this interface.

IComparable<-T>.CompareTo(T) Method

```
[ILAsm]
.method public hidebysig newslot abstract virtual instance int32
CompareTo(!0 other) cil managed

[C#]
public int CompareTo (T other)
```

Summary

Compares the current object with another object of the same type.

Parameters

Parameter	Description
other	An object to compare with this object.

Return Value

A value that indicates the relative order of the objects being compared. The return value has the following meanings:

Value	Meaning
Less than zero	This object is less than the *other* parameter.
Zero	This object is equal to *other*.
Greater than zero	This object is greater than *other*.

Description

``System.IComparable`1<T>.CompareTo`` provides a strongly typed comparison method for ordering members of a generic collection object. Because of this, it is usually not called directly from developer code. Instead, it is called automatically by methods such as ``System.Collections.Generic.List`1<T>.Sort`` and ``System.Collections.Generic.SortedList`2<T1,T2>.Add``.

This method is only a definition and must be implemented by a specific class or value type to have effect. The meaning of the comparisons, "less than," "equal to," and "greater than," depends on the particular implementation.

By definition, any object compares greater than `null`, and two null references compare equal to each other.

Behaviors

For objects A, B, and C, the following must be true:

`A.CompareTo(A)` is required to return zero.

If `A.CompareTo(B)` returns zero, then `B.CompareTo(A)` is required to return zero.

If `A.CompareTo(B)` returns zero and `B.CompareTo(C)` returns zero, then `A.CompareTo(C)` is required to return zero.

If `A.CompareTo(B)` returns a value other than zero, then `B.CompareTo(A)` is required to return a value of the opposite sign.

If `A.CompareTo(B)` returns a value *x* that is not equal to zero, and `B.CompareTo(C)` returns a value *y* of the same sign as *x*, then `A.CompareTo(C)` is required to return a value of the same sign as *x* and *y*.

Usage

Use the `` System.IComparable`1<T>.CompareTo `` method to determine the ordering of instances of a class.