

System.Collections.Generic.IComparer<T> Interface

```
[ILAsm]
.class interface public abstract IComparer`1<T>

[C#]
public interface IComparer<T>
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
 - CLSCompliantAttribute(true)

Summary

Provides a mechanism to customize comparison in sort ordering of a generic collection.

Library: BCL

IComparer<T>.Compare(T, T) Method

```
[ILAsm]  
.method public hidebysig virtual abstract int32 Compare(!0 x, !0 y)  
  
[C#]  
int Compare(T x, T y)
```

Summary

Returns the sort order of two T instances.

Parameters

Parameter	Description
x	First T to compare.
y	Second T to compare.

Return Value

A `System.Int32` containing a value that reflects the sort order of x as compared to y . The following table defines the conditions under which the returned value is a negative number, zero, or a positive number.

Value	Condition
A negative number	$x < y$.
Zero	$x == y$.
A positive number	$x > y$.

Behaviors

For any objects A , B and C , the following are required to be true:

`Compare(A,A)` is required to return zero.

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

If `Compare(A,B)` is zero, then `Compare(B,C)` and `Compare(A,C)` must have the same sign (negative, zero or positive).

If `Compare(B,C)` is zero, then `Compare(A,B)` and `Compare(A,C)` must have the same sign (negative, zero or positive).

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

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

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

The exact behavior of this method is unspecified. The intent of this method is to provide a mechanism that orders instances of a class in a manner that is consistent with the mathematical definitions of the relational operators (`<`, `>`, and `==`), without regard for class-specific definitions of the operators.