

# System.Collections.Comparer Class

```
[ILAsm]
.class public sealed serializable Comparer extends System.Object
implements System.Collections.IComparer

[C#]
public sealed class Comparer: IComparer
```

## 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)

## Implements:

- **System.Collections.IComparer**

## Summary

Provides the default implementation of the System.Collections.IComparer interface.

## Inherits From: System.Object

**Library:** BCL

**Thread Safety:** All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

# 1 Comparer.Default Field

```
2 [ILAsm]  
3 .field public static initOnly class System.Collections.Comparer Default  
  
4 [C#]  
5 public static readonly Comparer Default
```

## 6 Summary

7 Returns a new System.Collections.Comparer instance containing the default  
8 implementation of the System.Collections.IComparer interface.

## 9 Description

10 This field is read-only.

# Comparer.Compare(System.Object, System.Object) Method

```
[ILAsm]  
.method public final hidebysig virtual int32 Compare(object a, object b)  
  
[C#]  
public int Compare(object a, object b)
```

## Summary

Returns the sort order of two *System.Object* instances.

## Parameters

Parameter	Description
<i>a</i>	The first <i>System.Object</i> to compare.
<i>b</i>	The second <i>System.Object</i> to compare.

## Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of *a* as compared to *b*. For non-zero return values, the exact value returned by this method is unspecified. The following table defines the return value:

Value	Condition
A negative number	$a < b$ .
Zero	$a == b$ .
A positive number	$a > b$ .

[*Note:* A null reference is considered to compare less than any other non-null object, and equal to any other null reference, independent of the underlying *System.Type* of either object.]

## Description

The behavior of this method is as follows:

- 1 • If *a* implements the `System.IComparable` interface, returns *a.CompareTo(b)*.
- 2 • If *a* does not implement the `System.IComparable` interface but *b* does, returns the
- 3 negated result of *b.CompareTo(a)*.
- 4 • If *a* and *b* both are not null and do not implement the `System.IComparable`
- 5 interface, `System.ArgumentException` is thrown.

## 6 Exceptions

Exception	Condition
<b>System.ArgumentException</b>	Both <i>a</i> and <i>b</i> are not null and do not implement the <code>System.IComparable</code> interface.  -or-  Both <i>a</i> and <i>b</i> are not null and are not assignment-compatible types.

7  
8