System.Collections.Generic.Comparer<T> Class

```
[ILAsm]
.class public abstract serializable beforefieldinit
System.Collections.Generic.Comparer`1<T> extends System.Object implements
System.Collections.IComparer, class
System.Collections.Generic.IComparer`1<!0>

[C#]
public abstract class Comparer<T>:
System.Collections.Generic.IComparer<T>, System.Collections.IComparer
```

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)

Implements:

- **System.Collections.Generic.IComparer<T>**
- **System.Collections.IComparer**

Summary

Provides a base class for implementations of the `System.Collections.Generic.IComparer`1<T>` generic interface.

Inherits From: System.Object

Library: BCL

Description

Derive from this class to provide a custom implementation of the `System.Collections.Generic.IComparer`1<T>` interface for use with collection classes such as the `System.Collections.Generic.SortedList`2<T1,T2>` and `System.Collections.Generic.SortedDictionary`2<T1,T2>` generic classes.

The difference between deriving from the `System.Collections.Generic.Comparer`1<T>` class and implementing the `System.IComparable` interface is as follows:

- To specify how two objects should be compared by default, implement the `System.IComparable` interface in your class. This ensures that sort operations will use the default comparison code that you provided.

- To define a comparer to use instead of the default comparer, derive from the `System.Collections.Generic.Comparer`1<T>` class. You can then use this comparer in sort operations that take a comparer as a parameter.

The object returned by the `System.Collections.Generic.Comparer`1<T>.Default` property uses the `System.IComparable`1<T>` generic interface (IComparable<T> in C#) to compare two objects. If type *T* does not implement the `System.IComparable`1<T>` generic interface, the `System.Collections.Generic.Comparer`1<T>.Default` property returns a `System.Collections.Generic.Comparer`1<T>` that uses the `System.IComparable` interface.

Behaviors

`System.Collections.Generic.Comparer`1<T>.Compare` and `System.Collections.Generic.EqualityComparer`1<T>.Equals` may behave differently in terms of culture-sensitivity and case-sensitivity.

For string comparisons, the `System.StringComparer` class is recommended over Comparer<String>. Properties of the `System.StringComparer` class return predefined instances that perform string comparisons with different combinations of culture-sensitivity and case-sensitivity. The case-sensitivity and culture-sensitivity are consistent among the members of the same `System.StringComparer` instance.

For more information on culture-specific comparisons, see the `System.Globalization` namespace.

Comparer<T>() Constructor

[ILAsm]
```
.method family hidebysig specialname rtspecialname instance void .ctor()
cil managed
```

[C#]
```
protected Comparer ()
```

Summary

Initializes a new instance of the ``System.Collections.Generic.Comparer`1<T>`` class.

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

[ILAsm]
```
.method public hidebysig newslot abstract virtual instance int32
Compare(!0 x, !0 y) cil managed
```

[C#]
```
public abstract int Compare (T x, T y)
```

Summary

When overridden in a derived class, performs a comparison of two objects of the same type and returns a value indicating whether one object is less than, equal to, or greater than the other.

Parameters

Parameter	Description
x	The first object to compare.
y	The second object to compare.

Return Value

A signed integer that indicates the relative values of *x* and *y*, as shown in the following table.

Value	Meaning
Less than zero	*x* is less than *y*.
Zero	*x* equals *y*.
Greater than zero	*x* is greater than *y*.

Description

Implement this method to provide a customized sort order comparison for type *T*.

Behaviors

Comparing `null` with any reference type is allowed and does not generate an exception. A null reference is considered to be less than any reference that is not null.

For information on culture-specific comparisons, see the `System.Globalization` namespace.

Exceptions

Exception	Condition
System.ArgumentException	Type *T* does not implement either the ``System.IComparable`1<T>`` generic interface or the `System.IComparable` interface.

Comparer<T>.System.Collections.IComparer.Compare(System.Object, System.Object) Method

```
[ILAsm]
.method private hidebysig newslot virtual final instance int32
System.Collections.IComparer.Compare(object x, object y) cil managed

[C#]
int IComparer.Compare (object x, object y)
```

Summary

Compares two objects and returns a value indicating whether one is less than, equal to, or greater than the other.

Parameters

Parameter	Description
x	The first object to compare.
y	The second object to compare.

Return Value

A signed integer that indicates the relative values of *x* and *y*, as shown in the following table.

Value	Meaning
Less than zero	*x* is less than *y*.
Zero	*x* equals *y*.
Greater than zero	*x* is greater than *y*.

Description

This method is a wrapper for the `System.Collections.Generic.Comparer`1<T>.Compare` method, so *obj* must be cast to the type specified by the generic argument *T* of the current instance. If it cannot be cast to *T*, an `System.ArgumentException` is thrown.

Comparing `null` with any reference type is allowed and does not generate an exception. When sorting, `null` is considered to be less than any other object.

Usage

`System.Collections.Generic.Comparer`1<T>.Compare` and `System.Collections.Generic.EqualityComparer`1<T>.Equals` behave differently in terms of culture-sensitivity and case-sensitivity.

For string comparisons, the `System.StringComparer` class is recommended over Comparer<String>. Properties of the `System.StringComparer` class return predefined instances that perform string comparisons with different combinations of culture-sensitivity and case-sensitivity. The case-sensitivity and culture-sensitivity are consistent among the members of the same `System.StringComparer` instance.

For more information on culture-specific comparisons, see the `System.Globalization` namespace.

Exceptions

Exception	Condition
System.ArgumentException	*x* or *y* is of a type that cannot be cast to type *T*. -or- *x* and *y* do not implement either the `System.IComparable`1<T>` generic interface or the `System.IComparable` interface.

Comparer<T>.Default Property

```
[ILAsm]
.property class System.Collections.Generic.Comparer`1<!0> Default() {.get
class System.Collections.Generic.Comparer`1<!0>
System.Collections.Generic.Comparer`1::get_Default() }

[C#]
public static System.Collections.Generic.Comparer<T> Default { get; }
```

Summary

Returns a default sort order comparer for the type specified by the generic argument.

Property Value

An object that inherits `System.Collections.Generic.Comparer`1<T>` and serves as a sort order comparer for type *T*.

Description

The `System.Collections.Generic.Comparer`1<T>` returned by this property uses the `System.IComparable`1<T>` generic interface (IComparable<T> in C#) to compare two objects. If type *T* does not implement the `System.IComparable`1<T>` generic interface, this property returns a `System.Collections.Generic.Comparer`1<T>` that uses the `System.IComparable` interface.

Usage

For string comparisons, the `System.StringComparer` class is recommended over Comparer<String>. Properties of the `System.StringComparer` class return predefined instances that perform string comparisons with different combinations of culture-sensitivity and case-sensitivity. The case-sensitivity and culture-sensitivity are consistent among the members of the same `System.StringComparer` instance.

For more information on culture-specific comparisons, see the `System.Globalization` namespace.