System.String Class

```
[ILAsm]
.class public sealed serializable String extends System.Object
implements System.ICloneable, System.IComparable,
System.Collections.IEnumerable, System.IComparable`1<string>,
System.IEquatable`1<string>,
System.Collections.Generic.IEnumerable`1<char>

[C#]
public sealed class String: ICloneable, IComparable, IEnumerable,
IComparable<String>, IEquatable<String>, IEnumerable<Char>
```

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)

Type Attributes:

- DefaultMemberAttribute("Chars") [*Note:* This attribute requires the RuntimeInfrastructure library.]

Implements:

- **System.IComparable**
- **System.ICloneable**
- **System.IComparable<System.String>**
- **System.IEquatable<System.String>**
- **System.Collections.IEnumerable**
- **System.Collections.Generic.IEnumerable<System.Char>**

Summary

Represents an immutable series of characters.

Inherits From: System.Object

Library: BCL

Thread Safety: This type is safe for multithreaded operations.

Description

An *index* is the position of a character within a string. The first character in the string is at index 0. The length of a string is the number of characters it is made

up of. The last accessible *index* of a string instance is `System.String.Length` - 1.

Strings are immutable; once created, the contents of a `System.String` do not change. Combining operations, such as `System.String.Replace`, cannot alter existing strings. Instead, such operations return a new string that contains the results of the operation, an unchanged string, or the null value. To perform modifications to a `System.String` use the `System.Text.StringBuilder`.

Implementations of `System.String` are required to contain a variable-length character buffer positioned a fixed number of bytes after the beginning of the String object. [*Note:* The `System.Runtime.CompilerServices.RuntimeHelpers.OffsetToStringData` method returns the number of bytes between the start of the String object and the character buffer. This information is intended primarily for use by compilers, not application programmers. For additional information, see `System.Runtime.CompilerServices.RuntimeHelpers.OffsetToStringData`.]

[*Note:* Comparisons and searches are case-sensitive by default, and unless otherwise specified, use the culture defined (if any) for the current thread to determine the order of the alphabet used by the strings. This information is then used to compare the two strings on a character-by-character basis. Upper case letters evaluate greater than their lowercase equivalents.

The following characters are considered white space when present in a `System.String` instance: 0x9, 0xA, 0xB, 0xC, 0xD, 0x20, 0xA0, 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005, 0x2006, 0x2007, 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF. The null character is defined as hexadecimal 0x00.

The `System.String`(`System.String`) constructor is omitted for performance reasons. If you need a copy of a `System.String`, consider using `System.String.Copy` or the `System.Text.StringBuilder` class.

To insert a formatted string representation of an object into a string, use the `System.String.Format` methods. These methods take one or more arguments to be formatted, and a format string. The format string contains literals and zero or more format specifications of the form { *N* [, *M*][: *formatSpecifier*]}, where:

- *N* is a zero-based integer indicating the argument to be formatted. If the actual argument is a null reference, then an empty string is used in its place.

- *M* is an optional integer indicating the minimum width of the region to contain the formatted value of argument *N*. If the length of the string representation of the value is less than *M*, then the region is padded with spaces. If *M* is negative, the formatted value is left justified in the region; if *M* is positive, then the value is right justified. If *M* is not specified, it is assumed to be zero indicating that neither padding nor alignment is customized. Note that if the length of the formatted value is greater than *M*, then *M* is ignored.

- *formatSpecifier* is an optional string that determines the representation used for arguments. For example, an integer can be represented in hexadecimal or decimal format, or as a monetary value. If *formatSpecifier* is omitted and an argument implements the `System.IFormattable` interface, then a null reference is used as the `System.IFormattable.ToString` format specifier. Therefore, all implementations of `System.IFormattable.ToString` are required to allow a null reference as a format specifier, and return a string containing the default representation of the object as determined by the object type. For additional information on format specifiers, see `System.IFormattable`.

If an object referenced in the format string implements `System.IFormattable`, then the `System.IFormattable.ToString` method of the object provides the formatting. If the argument does not implement `System.IFormattable`, then the `System.Object.ToString` method of the object provides default formatting, and *formatSpecifier*, if present, is ignored. For an example that demonstrates this, see Example 2.

To include a curly bracket in a formatted string, specify the bracket twice; for example, specify "{{" to include "{" in the formatted string. See Example 1.

The `System.Console` class exposes the same functionality as the `System.String.Format` methods via `System.Console.Write` and `System.Console.WriteLine`. The primary difference is that the `System.String.Format` methods return the formatted string, while the System.Console methods write the formatted string to a stream.

]

When a non-empty string is searched for the first or last occurrence of an empty string, the empty string is found at the search start position.

Example

Example 1

The following example demonstrates formatting numeric data types and inserting literal curly brackets into strings.

[C#]

```
using System;
class StringFormatTest {
    public static void Main() {
        decimal dec = 1.99999m;
        double doub = 1.0000000001;

        string somenums = String.Format("Some formatted numbers:
dec={0,15:E} doub={1,20}", dec, doub);
        Console.WriteLine(somenums);

        string curlies = "Literal curly brackets: {{ and }} and {{0}}";
```

```
        Console.WriteLine(curlies);

        object nullObject = null;
        string embeddedNull = String.Format("A null argument looks
like: {0}", nullObject);
        Console.WriteLine(embeddedNull);
    }
}
```

The output is

```
Some formatted numbers: dec=  1.999990E+000 doub=        1.0000000001
Literal curly brackets: {{ and }} and {{0}}
A null argument looks like:
```

Example 2

The following example demonstrates how formatting works if `System.IFormattable` is or is not implemented by an argument to the `System.String.Format` method. Note that the format specifier is ignored if the argument does not implement `System.IFormattable`.

[C#]

```
using System;
class StringFormatTest {
    public class DefaultFormatEleven {
        public override string ToString() {
            return "11 string";
        }
    }
    public class FormattableEleven:IFormattable {
        // The IFormattable ToString implementation.
        public string ToString(string format, IFormatProvider
formatProvider) {
            Console.Write("[IFormattable called] ");
            return 11.ToString(format, formatProvider);
        }
        // Override Object.ToString to show that it is not called.
        public override string ToString() {
            return "Formatted 11 string";
        }
    }

    public static void Main() {
        DefaultFormatEleven def11 = new DefaultFormatEleven ();
         FormattableEleven for11 = new  FormattableEleven();
        string def11string = String.Format("{0}",def11);
        Console.WriteLine(def11string);
        // The format specifier x is ignored.
        def11string = String.Format("{0,15:x}", def11);
        Console.WriteLine(def11string);

        string form11string = String.Format("{0}",for11);
        Console.WriteLine(form11string );
        form11string = String.Format("{0,15:x}",for11);
```

```
        Console.WriteLine(form11string);
    }
}
```
The output is

```
11 string
      11 string
[IFormattable called] 11
[IFormattable called]               b
```

Example 3

The following example demonstrates searching for an empty string in a non-empty string.

```
[C#]
using System;
class EmptyStringSearch {
        public static void Main()      {
                Console.WriteLine("ABCDEF".IndexOf(""));
                Console.WriteLine("ABCDEF".IndexOf("", 2));
                Console.WriteLine("ABCDEF".IndexOf("", 3, 2));
                Console.WriteLine("ABCDEF".LastIndexOf(""));
                Console.WriteLine("ABCDEF".LastIndexOf("", 1));
                Console.WriteLine("ABCDEF".LastIndexOf("", 4, 2));
        }
}
```
The output is

```
0
2
3
5
1
4
```

String(System.Char, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(valuetype
System.Char c, int32 count)

[C#]
public String(char c, int count)
```

Summary

Constructs and initializes a new instance of `System.String`.

Parameters

Parameter	Description
c	A `System.Char`.
count	A `System.Int32` containing the number of occurrences of *c*.

Description

If the specified number is 0, `System.String.Empty` is created.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*count* is less than zero.

Example

The following example demonstrates using this constructor.

[C#]

```
using System;

public class StringExample {
 public static void Main() {

 string s = new String('a', 10);

 Console.WriteLine(s);
 }
```

```
}
```

The output is

```
aaaaaaaaaa
```

String(System.Char*) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(char* value)

[C#]
unsafe public String(char* value)
```

Summary

Constructs and initializes a new instance of `System.String` using a specified pointer to a sequence of Unicode characters.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	A pointer to a null-terminated array of Unicode characters. If *value* is a null pointer, `System.String.Empty` is created.

Description

This member is not CLS-compliant. For a CLS-compliant alternative, use the `System.String`(`System.Char`[]) constructor.

This constructor copies the sequence of Unicode characters at the specified pointer until a null character (hexadecimal 0x00) is reached.

If the specified array is not null-terminated, the behavior of this constructor is system dependent. For example, such a situation might cause an access violation.

[*Note:* In C# this constructor is defined only in the context of unmanaged code.]

String(System.Char[]) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(char[] value)

[C#]
public String(char[] value)
```

Summary

Constructs and initializes a new instance of `System.String` by copying the specified array of Unicode characters.

Parameters

Parameter	Description
value	An array of Unicode characters.

Description

If the specified array is a null reference or contains no elements, `System.String.Empty` is created.

String(System.Char*, System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(char* value,
int32 startIndex, int32 length)

[C#]
unsafe public String(char* value, int startIndex, int length)
```

Summary

Constructs and initializes a new instance of `System.String` using a specified pointer to a sequence of Unicode characters, the index within that sequence at which to start copying characters, and the number of characters to be copied to construct the `System.String`.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	A pointer to an array of Unicode characters.
startIndex	A `System.Int32` containing the index within the array referenced by *value* from which to start copying.
length	A `System.Int32` containing the number of characters to copy from *value* to the new `System.String`. If *length* is zero, `System.String.Empty` is created.

Description

This member is not CLS-compliant. For a CLS-compliant alternative, use the `System.String`(`System.Char`, `System.Int32`, `System.Int32`) constructor.

This constructor copies Unicode characters from *value*, starting at *startIndex* and ending at (*startIndex* + *length* - 1).

If the specified range is outside of the memory allocated for the sequence of characters, the behavior of this constructor is system dependent. For example, such a situation might cause an access violation.

[*Note:* In C# this constructor is defined only in the context of unmanaged code.]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* or *length* is less than zero. -or- *value* is a null pointer and *length* is not zero.

The following member must be implemented if the RuntimeInfrastructure library is present in the implementation.

String(System.SByte*, System.Int32, System.Int32, System.Text.Encoding) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(class
System.SByte* value, int32 startIndex, int32 length, class
System.Text.Encoding enc)

[C#]
unsafe public String(sbyte* value, int startIndex, int length,
Encoding enc)
```

Summary

Constructs and initializes a new instance of the `String` class to the value indicated by a specified pointer to an array of 8-bit signed integers, a starting character position within that array, a length, and an `Encoding` object.

Type Attributes:

- CLSCompliantAttribute(false)

Parameters

Parameter	Description
value	A pointer to a `System.SByte` array.
startIndex	A `System.Int32` containing the starting position within *value*.
length	A `System.Int32` containing the number of characters within *value* to use. If *length* is zero, `System.String.Empty` is created.
enc	A `System.Text.Encoding` object that specifies how the array referenced by *value* is encoded.

Description

If *value* is a `null` pointer, a `System.String.Empty` instance is constructed.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* or *length* is less than zero.

	-or- *value* is a null pointer and *length* is not zero.

String(System.Char[], System.Int32, System.Int32) Constructor

```
[ILAsm]
public rtspecialname specialname instance void .ctor(char[] value,
int32 startIndex, int32 length)

[C#]
public String(char[] value, int startIndex, int length)
```

Summary

Constructs and initializes a new instance of `System.String` using an array of Unicode characters, the index within array at which to start copying characters, and the number of characters to be copied.

Parameters

Parameter	Description
value	An array of Unicode characters.
startIndex	A `System.Int32` containing the index within the array referenced by *value* from which to start copying.
length	A `System.Int32` containing the number of characters to copy from the *value* array. If *length* is zero, `System.String.Empty` is created.

Description

This constructor copies the sequence Unicode characters found at *value* between indexes *startIndex* and *startIndex* + *length* - 1.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *length* is less than zero. -or- The sum of *startIndex* and *length* is greater than the number of elements in *value*.

String.Empty Field

```
[ILAsm]
.field public static initOnly string Empty

[C#]
public static readonly string Empty
```

Summary

A constant string representing the empty string.

Description

This field is read-only.

This field is a string of length zero, "".

String.Clone() Method

```
[ILAsm]
.method public final hidebysig virtual object Clone()

[C#]
public object Clone()
```

Summary

Returns a reference to the current instance of `System.String`.

Return Value

A reference to the current instance of `System.String`.

Description

[*Note:* `System.String.Clone` does not generate a new `System.String` instance. Use the `System.String.Copy` or `System.String.CopyTo` method to create a separate `System.String` object with the same value as the current instance.

This method is implemented to support the `System.ICloneable` interface.

]

String.Compare(System.String, System.Int32, System.String, System.Int32, System.Int32, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static int32 Compare(string strA, int32
indexA, string strB, int32 indexB, int32 length, bool ignoreCase)

[C#]
public static int Compare(string strA, int indexA, string strB, int
indexB, int length, bool ignoreCase)
```

Summary

Compares substrings of two strings, ignoring or honoring their case.

Parameters

Parameter	Description
strA	The first `System.String` containing a substring to compare. Can be a null reference.
indexA	A `System.Int32` containing the starting index of the substring within *strA*.
strB	The second `System.String` containing a substring to compare. Can be a null reference.
indexB	A `System.Int32` containing the starting index of the substring within *strB*.
length	A `System.Int32` containing the maximum number of characters in the substrings to compare. If *length* is zero, then zero is returned.
ignoreCase	A `System.Boolean` indicating if the comparison is case-insensitive. If *ignoreCase* is `true`, the comparison is case-insensitive. If *ignoreCase* is `false`, the comparison is case-sensitive, and uppercase letters evaluate greater than their lowercase equivalents.

Return Value

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

Value Type	Condition

A negative number	The substring in *strA* is < the substring in *strB*.
Zero	The substring in *strA* == the substring in *strB*, or *length* is zero.
A positive number	The substring in *strA* is > the substring in *strB*.

Description

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Uppercase letters evaluate greater than their lower case equivalents.

The maximum number of characters compared is the lesser of the length of *strA* less *indexA*, the length of *strB* less *indexB*, and *length*.

When a culture is available, the method uses the culture of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*indexA* is greater than *strA*.Length -or- *indexB* is greater than *strB*.Length -or- *indexA*, *indexB*, or *length* is negative.

Example

The following example demonstrates comparing substrings with and without case sensitivity.

[C#]

```
using System;
public class StringCompareExample {
 public static void Main() {
 string strA = "STRING A";
 string strB = "string b";
```

```
 int first = String.Compare( strA, strB, true );
 int second = String.Compare( strA, 0, strB, 0, 4, true );
 int third = String.Compare( strA, 0, strB, 0, 4, false );
 Console.WriteLine( "When the string 'STRING A' is compared to the
string 'string b' in a case-insensitive manner, the return value is
{0}.", first );
 Console.WriteLine( "When the substring 'STRI' of 'STRING A' is
compared to the substring 'stri' of 'string b' in a case-insensitive
manner, the return value is {0}.", second );
 Console.WriteLine( "When the substring 'STRI' of 'STRING A' is
compared to the substring 'stri' of 'string b' in a case-sensitive
manner, the return value is {0}.", third );
 }
}
```

The output is

```
When the string 'STRING A' is compared to the string 'string b' in a
case-insensitive manner, the return value is -1.


When the substring 'STRI' of 'STRING A' is compared to the substring
'stri' of 'string b' in a case-insensitive manner, the return value is
0.


When the substring 'STRI' of 'STRING A' is compared to the substring
'stri' of 'string b' in a case-sensitive manner, the return value is 1.
```

String.Compare(System.String, System.Int32, System.String, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int32 Compare(string strA, int32
indexA, string strB, int32 indexB, int32 length)

[C#]
public static int Compare(string strA, int indexA, string strB, int
indexB, int length)
```

Summary

Compares substrings of two strings.

Parameters

Parameter	Description
strA	The first `System.String` to compare. Can be a null reference.
indexA	A `System.Int32` containing the starting index of the substring within *strA*.
strB	The second `System.String` to compare. Can be a null reference.
indexB	A `System.Int32` containing the starting index of the substring within *strB*.
length	A `System.Int32` containing the maximum number of characters in the substrings to compare. If *length* is zero, then zero is returned.

Return Value

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

Value	Meaning
A negative number	The substring in *strA* is < the substring in *strB*.
Zero	The substring in *strA* == the substring in *strB*, or *length* is zero.
A positive number	The substring in *strA* is > the substring in *strB*.

Description

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Uppercase letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	The sum of *indexA* and *length* is greater than *strA*.Length. -or- The sum of *indexB* and *length* is greater than *strB*.Length. -or- *indexA*, *indexB*, or *length* is negative.

Example

The following example demonstrates comparing substrings.

[C#]

```
using System;
public class StringCompareExample {
 public static void Main() {
 string strA = "A string";
 string strB = "B ring";
 int first = String.Compare( strA, 4, strB, 2, 3 );
 int second = String.Compare( strA, 3, strB, 3, 3 );
 Console.WriteLine( "When the substring 'rin' of 'A string' is compared
to the substring 'rin' of 'B ring', the return value is {0}.", first );
 Console.WriteLine( "When the substring 'tri' of 'A string' is compared
to the substring 'ing' of 'B ring', the return value is {0}.", second
);
 }
}
```

The output is

When the substring 'rin' of 'A string' is compared to the substring 'rin' of 'B ring', the return value is 0.

When the substring 'tri' of 'A string' is compared to the substring 'ing' of 'B ring', the return value is 1.

String.Compare(System.String, System.String, System.Boolean) Method

```
[ILAsm]
.method public hidebysig static int32 Compare(string strA, string
strB, bool ignoreCase)

[C#]
public static int Compare(string strA, string strB, bool ignoreCase)
```

Summary

Returns sort order of two `System.String` objects, ignoring or honoring their case.

Parameters

Parameter	Description
strA	The first `System.String` to compare. Can be a null reference.
strB	The second `System.String` to compare. Can be a null reference.
ignoreCase	A `System.Boolean` indicating whether the comparison is case-insensitive. If *ignoreCase* is `true`, the comparison is case-insensitive. If *ignoreCase* is `false`, the comparison is case-sensitive, and uppercase letters evaluate greater than their lowercase equivalents.

Return Value

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

Value	Meaning
A negative number	*strA* is < *strB*.
Zero	*strA* == *strB*.
A positive number	*strA* is > *strB*.

Description

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Uppercase letters evaluate greater than their lowercase

equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

`String.Compare`(*strA*, *strB*, `false`) is equivalent to `String.Compare`(*strA*, *strB*).

]

Example

The following example demonstrates comparing strings with and without case sensitivity.

[C#]

```
using System;
public class StringCompareExample {
 public static void Main() {
 string strA = "A STRING";
 string strB = "a string";
 int first = String.Compare( strA, strB, true );
 int second = String.Compare( strA, strB, false );
 Console.WriteLine( "When 'A STRING' is compared to 'a string' in a
case-insensitive manner, the return value is {0}.", first );
 Console.WriteLine( "When 'A STRING' is compared to 'a string' in a
case-sensitive manner, the return value is {0}.", second );
 }
}
```

The output is

```
When 'A STRING' is compared to 'a string' in a case-insensitive manner,
the return value is 0.


When 'A STRING' is compared to 'a string' in a case-sensitive manner,
the return value is 1.
```

String.Compare(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static int32 Compare(string strA, string
strB)

[C#]
public static int Compare(string strA, string strB)
```

Summary

Compares two `System.String` objects in a case-sensitive manner.

Parameters

Parameter	Description
strA	The first `System.String` to compare. Can be a null reference.
strB	The second `System.String` to compare. Can be a null reference.

Return Value

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

Value	Meaning
A negative number	*strA* is lexicographically < *strB*.
Zero	*strA* is lexicographically == *strB*.
A positive number	*strA* is lexicographically > *strB*.

Description

This method performs a case-sensitive operation.

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Uppercase letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-

by-character basis.

]

String.CompareOrdinal(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static int32 CompareOrdinal(string strA,
string strB)

[C#]
public static int CompareOrdinal(string strA, string strB)
```

Summary

Compares two specified `System.String` objects based on the code points of the contained Unicode characters.

Parameters

Parameter	Description
strA	The first `System.String` to compare.
strB	The second `System.String` to compare.

Return Value

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

Value	Description
A negative number	*strA* is < *strB*, or *strA* is a null reference.
Zero	*strA* == *strB*, or both *strA* and *strB* are null references.
A positive number	*strA* is > *strB*, or *strB* is a null reference.

Description

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Uppercase letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

]

String.CompareOrdinal(System.String, System.Int32, System.String, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static int32 CompareOrdinal(string strA,
int32 indexA, string strB, int32 indexB, int32 length)

[C#]
public static int CompareOrdinal(string strA, int indexA, string
strB, int indexB, int length)
```

Summary

Compares substrings of two specified `System.String` objects based on the code points of the contained Unicode characters.

Parameters

Parameter	Description
strA	The first `System.String` to compare.
indexA	A `System.Int32` containing the starting index of the substring in *strA*.
strB	The second `System.String` to compare.
indexB	A `System.Int32` containing the starting index of the substring in *strB*.
length	A `System.Int32` containing the number of characters in the substrings to compare.

Return Value

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

Value Type	Condition
A negative number	The substring in *strA* is < the substring in *strB*, or *strA* is a null reference.
Zero	The substring in *strA* == the substring in *strB*, or both *strA* and *strB* are null references.
A positive number	The substring in *strA* is > the substring in *strB*, or *strB* is a null reference.

Description

When either of the String arguments is the null reference an `System.ArgumentOutOfRangeException` shall be thrown if the corresponding index is non-zero.

[*Note:* The maximum number of characters compared is the lesser of the length of *strA* less *indexA*, the length of *strB* less *indexB*, and *length*.

The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. The result of comparing two null references is zero. Upper case letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

]

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*indexA* is greater than *strA*.Length -or- *indexB* is greater than *strB*.Length -or- *indexA*, *indexB*, or *length*is negative.

String.CompareTo(System.Object) Method

```
[ILAsm]
.method public final hidebysig virtual int32 CompareTo(object value)

[C#]
public int CompareTo(object value)
```

Summary

Returns the sort order of the current instance compared to the specified object.

Parameters

Parameter	Description
value	The `System.Object` to compare to the current instance.

Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *value*. 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	The current instance is lexicographically < *value*.
Zero	The current instance is lexicographically == *value*.
A positive number	The current instance is lexicographically > *value*, or *value* is a null reference.

Description

value is required to be a `System.String` object.

[*Note:* The result of comparing any `System.String` (including the empty string) to a null reference is greater than zero. Uppercase letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the

ordering of individual characters. The two strings are compared on a character-by-character basis.

This method is implemented to support the `System.IComparable` interface.

]

Exceptions

Exception	Condition
System.ArgumentException	*value* is not a `System.String`.

String.CompareTo(System.String) Method

```
[ILAsm]
.method public final hidebysig virtual int32 CompareTo(string strB)

[C#]
public int CompareTo(string strB)
```

Summary

Returns the sort order of the current instance compared to the specified string.

Parameters

Parameter	Description
strB	The `System.String` to compare to the current instance.

Return Value

The return value is a negative number, zero, or a positive number reflecting the sort order of the current instance as compared to *strB*. 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	The current instance is lexicographically < *strB*.
Zero	The current instance is lexicographically == *strB*.
A positive number	The current instance is lexicographically > *strB*, or *strB* is a null reference.

Description

[*Note:* Uppercase letters evaluate greater than their lowercase equivalents.

The method uses the culture (if any) of the current thread to determine the ordering of individual characters. The two strings are compared on a character-by-character basis.

This method is implemented to support the `System.IComparable<System.String>` interface.

]

String.Concat(System.Object, System.Object) Method

```
[ILAsm]
.method public hidebysig static string Concat(object arg0, object
arg1)

[C#]
public static string Concat(object arg0, object arg1)
```

Summary

Concatenates the `System.String` representations of two specified objects.

Parameters

Parameter	Description
arg0	The first `System.Object` to concatenate.
arg1	The second `System.Object` to concatenate.

Return Value

The concatenated `System.String` representation of the values of *arg0* and *arg1*.

Description

`System.String.Empty` is used in place of any null argument.

This version of `System.String.Concat` is equivalent to `System.String.Concat`(*arg0*.ToString(), *arg1*.ToString ()).

[*Note:* If either of the arguments is an array reference, the method concatenates a string representing that array, instead of its members (for example, `System.String`)[].]

Example

The following example demonstrates concatenating two objects.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
```

```
 string str = String.Concat( 'c', 32 );
 Console.WriteLine( "The concatenated Objects are: {0}", str );
 }
}
```

The output is

```
The concatenated Objects are: c32
```

String.Concat(System.Object, System.Object, System.Object) Method

```
[ILAsm]
.method public hidebysig static string Concat(object arg0, object
arg1, object arg2)

[C#]
public static string Concat(object arg0, object arg1, object arg2)
```

Summary

Concatenates the `System.String` representations of three specified objects, in order provided.

Parameters

Parameter	Description
arg0	The first `System.Object` to concatenate.
arg1	The second `System.Object` to concatenate.
arg2	The third `System.Object` to concatenate.

Return Value

The concatenated `System.String` representations of the values of *arg0*, *arg1*, and *arg2*.

Description

This method concatenates the values returned by the `System.String.ToString` methods on every argument. `System.String.Empty` is used in place of any null argument.

This version of `System.String.Concat` is equivalent to `String.Concat(` *arg0*`.ToString(), ` *arg1*`.ToString(), ` *arg2*`.ToString () )`.

Example

The following example demonstrates concatenating three objects.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
 string str = String.Concat( 'c', 32, "String" );
```

```
 Console.WriteLine( "The concatenated Objects are: {0}", str );
 }
}
```

The output is

```
The concatenated Objects are: c32String
```

String.Concat(System.Object[]) Method

```
[ILAsm]
.method public hidebysig static string Concat(object[] args)

[C#]
public static string Concat(params object[] args)
```

Summary

Concatenates the `System.String` representations of the elements in an array of `System.Object` instances.

Parameters

Parameter	Description
args	An array of `System.Object` instances to concatenate.

Return Value

The concatenated `System.String` representations of the values of the elements in *args*.

Description

This method concatenates the values returned by the `System.String.ToString` methods on every object in the *args* array. `System.String.Empty` is used in place of any null reference in the array.

Exceptions

Exception	Condition
System.ArgumentNullException	*args* is a null reference.

Example

The following example demonstrates concatenating an array of objects.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
 string str = String.Concat( 'c', 32, "String" );
 Console.WriteLine( "The concatenated Object array is: {0}", str );
```

```
 }
}
```

The output is

```
The concatenated Object array is: c32String
```

String.Concat(System.String, System.String) Method

[ILAsm]
```
.method public hidebysig static string Concat(string str0, string str1)
```

[C#]
```
public static string Concat(string str0, string str1)
```

Summary

Concatenates two specified instances of `System.String`.

Parameters

Parameter	Description
str0	The first `System.String` to concatenate.
str1	The second `System.String` to concatenate.

Return Value

A `System.String` containing the concatenation of *str0* and *str1*.

Description

`System.String.Empty` is used in place of any null argument.

Example

The following example demonstrates concatenating two strings.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
 string str = String.Concat( "one", "two" );
 Console.WriteLine( "The concatenated strings are: {0}", str );
 }
}
```

The output is

```
The concatenated strings are: onetwo
```

String.Concat(System.String, System.String, System.String) Method

[ILAsm]
.method public hidebysig static string Concat(string str0, string str1, string str2)

[C#]
public static string Concat(string str0, string str1, string str2)

Summary

Concatenates three specified instances of `System.String`.

Parameters

Parameter	Description
str0	The first `System.String` to concatenate.
str1	The second `System.String` to concatenate.
str2	The third `System.String` to concatenate.

Return Value

A `System.String` containing the concatenation of *str0*, *str1*, and *str2*.

Description

`System.String.Empty` is used in place of any null argument.

Example

The following example demonstrates concatenating three strings.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
 string str = String.Concat( "one", "two", "three" );
 Console.WriteLine( "The concatenated strings are: {0}", str );
 }
}
```

The output is

```
The concatenated strings are: onetwothree
```

String.Concat(System.String[]) Method

```
[ILAsm]
.method public hidebysig static string Concat(string[] values)

[C#]
public static string Concat(params string[] values)
```

Summary

Concatenates the elements of a specified array.

Parameters

Parameter	Description
values	An array of `System.String` instances to concatenate.

Return Value

A `System.String` containing the concatenated elements of *values*.

Description

`System.String.Empty` is used in place of any null reference in the array.

Exceptions

Exception	Condition
System.ArgumentNullException	*values* is a null reference.

Example

The following example demonstrates concatenating an array of strings.

[C#]

```
using System;
public class StringConcatExample {
 public static void Main() {
 string str = String.Concat( "one", "two", "three", "four", "five" );
 Console.WriteLine( "The concatenated String array is: {0}", str );
 }
}
```

The output is

```
The concatenated String array is: onetwothreefourfive
```

String.Copy(System.String) Method

[ILAsm]
.method public hidebysig static string Copy(string str)

[C#]
public static string Copy(string str)

Summary

Creates a new instance of `System.String` with the same value as a specified instance of `System.String`.

Parameters

Parameter	Description
str	The `System.String` to be copied.

Return Value

A new `System.String` with the same value as *str*.

Exceptions

Exception	Condition
System.ArgumentNullException	*str* is a null reference.

Example

The following example demonstrates copying strings.

[C#]

```
using System;
public class StringCopyExample {
 public static void Main() {
 string strA = "string";
 Console.WriteLine( "The initial string, strA, is '{0}'.", strA );
 string strB = String.Copy( strA );
 strA = strA.ToUpper();
 Console.WriteLine( "The copied string, strB, before strA.ToUpper, is
'{0}'.", strB );
 Console.WriteLine( "The initial string after StringCopy and ToUpper,
is '{0}'.", strA );
 Console.WriteLine( "The copied string, strB, after strA.ToUpper, is
'{0}'.", strB );
```

```
 }
}
```

The output is

```
The initial string, strA, is 'string'.

The copied string, strB, before strA.ToUpper, is 'string'.

The initial string after StringCopy and ToUpper, is 'STRING'.

The copied string, strB, after strA.ToUpper, is 'string'.
```

String.CopyTo(System.Int32, System.Char[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance void CopyTo(int32 sourceIndex,
char[] destination, int32 destinationIndex, int32 count)

[C#]
public void CopyTo(int sourceIndex, char[] destination, int
destinationIndex, int count)
```

Summary

Copies a specified number of characters from a specified position in the current `System.String` instance to a specified position in a specified array of Unicode characters.

Parameters

Parameter	Description
sourceIndex	A `System.Int32` containing the index of the current instance from which to copy.
destination	An array of Unicode characters.
destinationIndex	A `System.Int32` containing the index of an array element in *destination* to copy.
count	A `System.Int32` containing the number of characters in the current instance to copy to *destination*.

Exceptions

Exception	Condition
System.ArgumentNullException	*destination* is a null reference.
System.ArgumentOutOfRangeException	*sourceIndex*, *destinationIndex*, or *count* is negative -or- *count* is greater than the length of the substring from *startIndex* to the end of the current instance -or-

	count is greater than the length of the subarray from *destinationIndex* to the end of *destination*

Example

The following example demonstrates copying characters from a string to a Unicode character array.

[C#]

```
using System;
public class StringCopyToExample {
 public static void Main() {
 string str = "this is the new string";
 Char[] cAry = {'t','h','e',' ','o','l','d'};
 Console.WriteLine( "The initial string is '{0}'", str );
 Console.Write( "The initial character array is: '" );
 foreach( Char c in cAry)
 Console.Write( c );
 Console.WriteLine( "'" );
 str.CopyTo( 12, cAry, 4, 3 );
 Console.Write( "The character array after CopyTo is: '" );
 foreach( Char c in cAry)
 Console.Write( c );
 Console.WriteLine("'");
 }
}
```

The output is

```
The initial string is 'this is the new string'


The initial character array is: 'the old'


The character array after CopyTo is: 'the new'
```

String.EndsWith(System.String) Method

```
[ILAsm]
.method public hidebysig instance bool EndsWith(string value)

[C#]
public bool EndsWith(string value)
```

Summary

Returns a `System.Boolean` value that indicates whether the ending characters of the current instance match the specified `System.String`.

Parameters

Parameter	Description
value	A `System.String` to match.

Return Value

`true` if the end of the current instance is equal to *value*; `false` if *value* is not equal to the end of the current instance or is longer than the current instance.

Description

This method compares *value* with the substring at the end of the current instance that has a same length as *value.*

The comparison is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

Example

The following example demonstrates determining whether the current instance ends with a specified string.

[C#]

```
using System;
public class StringEndsWithExample {
 public static void Main() {
```

```
 string str = "One string to compare";
 Console.WriteLine( "The given string is '{0}'", str );
 Console.Write( "The given string ends with 'compare'? " );
 Console.WriteLine( str.EndsWith( "compare" ) );
 Console.Write( "The given string ends with 'Compare'? " );
 Console.WriteLine( str.EndsWith( "Compare" ) );
 }
}
```

The output is

```
The given string is 'One string to compare'


The given string ends with 'compare'? True


The given string ends with 'Compare'? False
```

String.Equals(System.Object) Method

```
[ILAsm]
.method public hidebysig virtual bool Equals(object obj)

[C#]
public override bool Equals(object obj)
```

Summary

Determines whether the current instance and the specified object have the same value.

Parameters

Parameter	Description
obj	A `System.Object`.

Return Value

`true` if *obj* is a `System.String` and its value is the same as the value of the current instance; otherwise, `false`.

Description

This method checks for value equality. This comparison is case-sensitive.

[*Note:* This method overrides `System.Object.Equals`.]

Exceptions

Exception	Condition
System.NullReferenceException	The current instance is a null reference.

Example

The following example demonstrates checking to see if an object is equal to the current instance.

[C#]

```
using System;
public class StringEqualsExample {
```

```
 public static void Main() {
 string str = "A string";
 Console.WriteLine( "The given string is '{0}'", str );
 Console.Write( "The given string is equal to 'A string'? " );
 Console.WriteLine( str.Equals( "A string" ) );
 Console.Write( "The given string is equal to 'A String'? " );
 Console.WriteLine( str.Equals( "A String" ) );
 }
}
```

The output is

```
The given string is 'A string'


The given string is equal to 'A string'? True


The given string is equal to 'A String'? False
```

String.Equals(System.String) Method

```
[ILAsm]
.method public hidebysig virtual bool Equals(string value)

[C#]
public override bool Equals(string value)
```

Summary

Determines whether the current instance and the specified string have the same value.

Parameters

Parameter	Description
value	A `System.String`.

Return Value

`true` if the value of *value* is the same as the value of the current instance; otherwise, `false`.

Description

This method checks for value equality. This comparison is case-sensitive.

[*Note:* This method is implemented to support the `System.IEquatable<System.String>` interface.]

String.Equals(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static bool Equals(string a, string b)

[C#]
public static bool Equals(string a, string b)
```

Summary

Determines whether two specified `System.String` objects have the same value.

Parameters

Parameter	Description
a	A `System.String` or a null reference.
b	A `System.String` or a null reference.

Return Value

`true` if the value of *a* is the same as the value of *b*; otherwise, `false`.

Description

The comparison is case-sensitive.

Example

The following example demonstrates checking to see if two strings are equal.

[C#]

```
using System;
public class StringEqualsExample {
 public static void Main() {
 string strA = "A string";
 string strB = "a string";
 string strC = "a string";
 Console.Write( "The string '{0}' is equal to the string '{1}'? ",
strA, strB );
 Console.WriteLine( String.Equals( strA, strB ) );
 Console.Write( "The string '{0}' is equal to the string '{1}'? ",
strC, strB );
 Console.WriteLine( String.Equals( strC, strB ) );
 }
}
```

The output is

```
The string 'A string' is equal to the string 'a string'? False

The string 'a string' is equal to the string 'a string'? True
```

String.Format(System.String, System.Object[]) Method

```
[ILAsm]
.method public hidebysig static string Format(string format,
object[] args)

[C#]
public static string Format(string format, params object[] args)
```

Summary

Replaces the format specification in a specified `System.String` with the textual equivalent of the value of a corresponding `System.Object` instance in a specified array.

Parameters

Parameter	Description
format	A `System.String` containing zero or more format specifications.
args	A `System.Object` array containing the objects to be formatted.

Return Value

A `System.String` containing a copy of *format* in which the format specifications have been replaced by the `System.String` equivalent of the corresponding instances of `System.Object` in *args*.

Description

If an object referenced in the format string is a null reference, an empty string is used in its place.

[*Note:* This version of `System.String.Format` is equivalent to `System.String.Format`(null, *format*, *args*). For more information on the format specification see the `System.String` class overview.]

Exceptions

Exception	Condition
System.ArgumentNullException	*format* or *args* is a null reference.
System.FormatException	*format* is invalid.

	-or- The number indicating an argument to be formatted is less than zero, or greater than or equal to the length of the *args* array.

Example

The following example demonstrates the `System.String.Format` method.

[C#]

```
using System;
public class StringFormat {
   public static void Main() {
      Console.WriteLine( String.Format("The winning numbers were
{0:000} {1:000} {2:000} {3:000} {4:000} today.", 5, 10, 11, 37, 42) );
      Console.WriteLine( "The winning numbers were {0, -6}{1, -6}{2, -
6}{3, -6}{4, -6} today.", 5, 10, 11, 37, 42 );
 }
}
```

The output is

```
The winning numbers were 005 010 011 037 042 today.
The winning numbers were 5     10    11    37    42     today.
```

String.Format(System.String, System.Object) Method

```
[ILAsm]
.method public hidebysig static string Format(string format, object
arg0)

[C#]
public static string Format(string format, object arg0)
```

Summary

Replaces the format specification in a provided `System.String` with a specified textual equivalent of the value of a specified `System.Object` instance.

Parameters

Parameter	Description
format	A `System.String` containing zero or more format specifications.
arg0	A `System.Object` to be formatted. Can be a null reference.

Return Value

A copy of *format* in which the first format specification has been replaced by the formatted `System.String` equivalent of the *arg0*.

Description

If an object referenced in the format string is a null reference, an empty string is used in its place.

[*Note:* This version of `System.String.Format` is equivalent to `String.Format(` `null`, *format*, `new Object`[] {*arg0*}). For more information on the format specification see the `System.String` class overview.]

Exceptions

Exception	Condition
System.ArgumentNullException	*format* is a null reference.
System.FormatException	The format specification in *format* is invalid.

	-or- The number indicating an argument to be formatted is less than zero, or greater than or equal to the number of provided objects to be formatted (1).

Example

The following example demonstrates the `System.String.Format` method.

[C#]

```
using System;
public class StringFormat {
 public static void Main() {
 Console.WriteLine(String.Format("The high temperature today was
{0:###} degrees.", 88));
 Console.WriteLine("The museum had {0,-6} visitors today.", 88);
 }
}
```

The output is

```
The high temperature today was 88 degrees.
The museum had 88     visitors today.
```

String.Format(System.String, System.Object, System.Object) Method

```
[ILAsm]
.method public hidebysig static string Format(string format, object
arg0, object arg1)

[C#]
public static string Format(string format, object arg0, object arg1)
```

Summary

Replaces the format specification in a specified `System.String` with the textual equivalent of the value of two specified `System.Object` instances.

Parameters

Parameter	Description
format	A `System.String` containing zero or more format specifications.
arg0	A `System.Object` to be formatted. Can be a null reference.
arg1	A `System.Object` to be formatted. Can be a null reference.

Return Value

A `System.String` containing a copy of *format* in which the format specifications have been replaced by the `System.String` equivalent of *arg0* and *arg1*.

Description

If an object referenced in the format string is a null reference, an empty string is used in its place.

[*Note:* This version of `System.String.Format` is equivalent to `String.Format(` `null`, *format*, `new Object[]` {*arg0*, *arg1*}). For more information on the format specification see the `System.String` class overview.]

Exceptions

Exception	Condition
System.ArgumentNullException	*format* is a null reference.
System.FormatException	*format* is invalid.

	-or- The number indicating an argument to be formatted is less than zero, or greater than or equal to the number of provided objects to be formatted (2).

Example

The following example demonstrates the `System.String.Format` method.

[C#]

```
using System;
public class StringFormat {
  public static void Main() {
  Console.WriteLine( String.Format("The temperature today oscillated
between {0:####} and {1:####} degrees.", 78, 100) );
  Console.WriteLine( String.Format("The temperature today oscillated
between {0:0000} and {1:0000} degrees.", 78, 100) );
  Console.WriteLine( "The temperature today oscillated between {0, -4}
and {1, -4} degrees.", 78, 100 );
   }
}
```

The output is

```
The temperature today oscillated between 78 and 100 degrees.
The temperature today oscillated between 0078 and 0100 degrees.
The temperature today oscillated between 78   and 100  degrees.
```

String.Format(System.String, System.Object, System.Object, System.Object) Method

```
[ILAsm]
.method public hidebysig static string Format(string format, object arg0, object arg1, object arg2)

[C#]
public static string Format(string format, object arg0, object arg1, object arg2)
```

Summary

Replaces the format specification in a specified `System.String` with the textual equivalent of the value of three specified `System.Object` instances.

Parameters

Parameter	Description
format	A `System.String` containing zero or more format specifications.
arg0	The first `System.Object` to be formatted. Can be a null reference.
arg1	The second `System.Object` to be formatted. Can be a null reference.
arg2	The third `System.Object` to be formatted. Can be a null reference.

Return Value

A `System.String` containing a copy of *format* in which the first, second, and third format specifications have been replaced by the `System.String` equivalent of *arg0*, *arg1*, and *arg2*.

Description

If an object referenced in the format string is a null reference, an empty string is used in its place.

[*Note:* This version of `System.String.Format` is equivalent to `String.Format(` `null`, *format*, `new Object[]` {*arg0*, *arg1*, *arg2*}). For more information on the format specification see the `System.String` class overview.]

Exceptions

Exception	Condition
System.ArgumentNullException	*format* is a null reference.
System.FormatException	*format* is invalid. -or- The number indicating an argument to be formatted is less than zero, or greater than or equal to the number of provided objects to be formatted (3).

Example

The following example demonstrates the `System.String.Format` method.

[C#]

```
using System;
public class StringFormat {
   public static void Main() {
      Console.WriteLine(String.Format("The temperature today oscillated
between {0:###} and {1:###} degrees. The average temperature  was
{2:000} degrees.", 78, 100, 91));
      Console.WriteLine("The temperature today oscillated between {0,
4} and {1, 4} degrees. The average temperature was {2, 4}  degrees.",
78, 100, 91);
   }
}
```

The output is

```
The temperature today oscillated between 78 and 100 degrees. The
average temperature was 091 degrees.
The temperature today oscillated between   78 and  100 degrees. The
average temperature was   91 degrees.
```

String.Format(System.IFormatProvider, System.String, System.Object[]) Method

```
[ILAsm]
.method public hidebysig static string Format(class
System.IFormatProvider provider, string format, object[] args)

[C#]
public static string Format(IFormatProvider provider, string format,
params object[] args)
```

Summary

Replaces the format specification in a specified `System.String` with the culture-specific textual equivalent of the value of a corresponding `System.Object` instance in a specified array.

Parameters

Parameter	Description
provider	A `System.IFormatProvider` interface that supplies an object that provides culture-specific formatting information. Can be a null reference.
format	A `System.String` containing zero or more format specifications.
args	A `System.Object` array to be formatted.

Return Value

A `System.String` containing a copy of *format* in which the format specifications have been replaced by the `System.String` equivalent of the corresponding instances of `System.Object` in *args*.

Description

If an object referenced in the format string is a null reference, an empty string is used in its place.

The *format* parameter string is embedded with zero or more format specifications of the form, {*N* [, *M*][: *formatString*]}, where *N* is a zero-based integer indicating the argument to be formatted, *M* is an optional integer indicating the width of the region to contain the formatted value, and *formatString* is an optional string of formatting codes. [*Note:* For more information on the format specification see the `System.String` class overview.]

Exceptions

Exception	Condition
System.ArgumentNullException	*format* or *args* is a null reference.
System.FormatException	*format* is invalid. -or- The number indicating an argument to be formatted (*N*) is less than zero, or greater than or equal to the length of the *args* array.

String.GetEnumerator() Method

```
[ILAsm]
.method public hidebysig instance CharEnumerator GetEnumerator()

[C#]
public CharEnumerator GetEnumerator()
```

Summary

Retrieves an object that can iterate through the individual characters in the current instance.

Return Value

A `System.CharEnumerator` object.

Description

This method is required by programming languages that support the `System.Collections.IEnumerator` interface to iterate through members of a collection.

String.GetHashCode() Method

```
[ILAsm]
.method public hidebysig virtual int32 GetHashCode()

[C#]
public override int GetHashCode()
```

Summary

Generates a hash code for the current instance.

Return Value

A `System.Int32` containing the hash code for this instance.

Description

The algorithm used to generate the hash code is unspecified.

[*Note:* This method overrides `System.Object.GetHashCode`.]

String.IndexOf(System.Char) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(valuetype
System.Char value)

[C#]
public int IndexOf(char value)
```

Summary

Returns the index of the first occurrence of a specified Unicode character in the current instance.

Parameters

Parameter	Description
value	A Unicode character.

Return Value

A `System.Int32` containing the zero-based index of the first occurrence of *value* character in the current instance; otherwise, -1 if *value* was not found.

Description

This method is case-sensitive.

String.IndexOf(System.Char, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(valuetype
System.Char value, int32 startIndex)

[C#]
public int IndexOf(char value, int startIndex)
```

Summary

Returns the index of the first occurrence of a specified Unicode character in the current instance, with the search starting from a specified index.

Parameters

Parameter	Description
value	A Unicode character.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.

Return Value

A `System.Int32` containing the zero-based index of the first occurrence of *value* in the current instance starting from the specified index; otherwise, -1 if *value* was not found.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* is less than zero or greater than the length of the current instance.

Example

The following example demonstrates the `System.String.IndexOf` method.

[C#]

```
using System;
public class StringIndexOf {
 public static void Main() {
 String str = "This is the string";
 Console.WriteLine( "Searching for the index of 'h' starting from index
0 yields {0}.", str.IndexOf( 'h', 0 ) );
 Console.WriteLine( "Searching for the index of 'h' starting from index
10 yields {0}.", str.IndexOf( 'h', 10 ) );
 }
}
```

The output is

```
Searching for the index of 'h' starting from index 0 yields 1.


Searching for the index of 'h' starting from index 10 yields -1.
```

String.IndexOf(System.Char, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(valuetype
System.Char value, int32 startIndex, int32 count)

[C#]
public int IndexOf(char value, int startIndex, int count)
```

Summary

Returns the index of the first occurrence of a specified Unicode character in the current instance, with the search over the specified range starting at the provided index.

Parameters

Parameter	Description
value	A Unicode character.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the number of consecutive elements of the current instance to be searched starting at *startIndex*.

Return Value

A `System.Int32` containing the zero-based index of the first occurrence of *value* in the current instance in the specified range of indexes; otherwise, -1 if *value* was not found.

Description

The search begins at *startIndex* and continues until *startIndex* + *count* - 1 is reached. The character at *startIndex* + *count* is not included in the search.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* or *count* is negative -or-

	startIndex + *count* is greater than the length of the current instance.

String.IndexOf(System.String) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(string value)

[C#]
public int IndexOf(string value)
```

Summary

Returns the index of the first occurrence of a specified `System.String` in the current instance.

Parameters

Parameter	Description
value	The `System.String` for which to search.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the first substring in the current instance that is equal to *value*.	*value* was found starting at the index returned.
-1	*value* was not found.

Description

The search begins at the first character of the current instance. The search is case-sensitive, culture-sensitive, and the culture (if any) of the current thread is used.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

Example

The following example demonstrates the `System.String.IndexOf` method.

[C#]

```
using System;
public class StringIndexOf {
 public static void Main() {
 String str = "This is the string";
 Console.WriteLine( "Searching for the index of \"is\" yields {0,2}.",
str.IndexOf( "is" ) );
 Console.WriteLine( "Searching for the index of \"Is\" yields {0,2}.",
str.IndexOf( "Is" ) );
 Console.WriteLine( "Searching for the index of \"\" yields {0,2}.",
str.IndexOf( "" ) );
 }
}
```

The output is

```
Searching for the index of "is" yields 2.


Searching for the index of "Is" yields -1.


Searching for the index of "" yields 0.
```

String.IndexOf(System.String, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(string value, int32
startIndex)

[C#]
public int IndexOf(string value, int startIndex)
```

Summary

Returns the index of the first occurrence of a specified `System.String` in the current instance, with the search starting from a specified index.

Parameters

Parameter	Description
value	The `System.String` for which to search.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the first substring in the current instance that is equal to *value*.	*value* was found starting at the index returned.
-1	*value* was not found.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

System.ArgumentOutOfRangeException	*startIndex* is greater than the length of the current instance.

String.IndexOf(System.String, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOf(string value, int32
startIndex, int32 count)

[C#]
public int IndexOf(string value, int startIndex, int count)
```

Summary

Returns the index of the first occurrence of a specified `System.String` in the current instance, with the search over the specified range starting at the provided index.

Parameters

Parameter	Description
value	The `System.String` for which to search
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the number of consecutive elements of the current instance to be searched starting at *startIndex*.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the first substring in the current instance that is equal to *value*.	*value* was found starting at the index returned.
-1	*value* was not found.

Description

The search begins at *startIndex* and continues until *startIndex* + *count* - 1 is reached. The character at *startIndex* + *count* is not included in the search.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *count* is negative -or- *startIndex* + *count* is greater than the length of the current instance.

String.IndexOfAny(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOfAny(char[] anyOf)

[C#]
public int IndexOfAny(char[] anyOf)
```

Summary

Reports the index of the first occurrence in the current instance of any character in a specified array of Unicode characters.

Parameters

Parameter	Description
anyOf	An array of Unicode characters.

Return Value

The index of the first occurrence of an element of *anyOf* in the current instance; otherwise, -1 if no element of *anyOf* was found.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.

String.IndexOfAny(System.Char[], System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOfAny(char[] anyOf,
int32 startIndex)

[C#]
public int IndexOfAny(char[] anyOf, int startIndex)
```

Summary

Returns the index of the first occurrence of any element in a specified array of Unicode characters in the current instance, with the search starting from a specified index.

Parameters

Parameter	Description
anyOf	An array of Unicode characters.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.

Return Value

A `System.Int32` containing a positive value equal to the index of the first occurrence of an element of *anyOf* in the current instance; otherwise, -1 if no element of *anyOf* was found.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* is greater than the length of the current instance

String.IndexOfAny(System.Char[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 IndexOfAny(char[] anyOf,
int32 startIndex, int32 count)

[C#]
public int IndexOfAny(char[] anyOf, int startIndex, int count)
```

Summary

Returns the index of the first occurrence of any element in a specified Array of Unicode characters in the current instance, with the search over the specified range starting from the provided index.

Parameters

Parameter	Description
anyOf	An array containing the Unicode characters to seek.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the range of the current instance at which to end searching.

Return Value

A `System.Int32` containing a positive value equal to the index of the first occurrence of an element of *anyOf* in the current instance; otherwise, -1 if no element of *anyOf* was found.

Description

The search begins at *startIndex* and continues until *startIndex* + *count* - *1*. The character at *startIndex* + *count* is not included in the search.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *count* is negative.

	-or- *startIndex* + *count* is greater than the length of the current instance.

String.Insert(System.Int32, System.String) Method

```
[ILAsm]
.method public hidebysig instance string Insert(int32 startIndex,
string value)

[C#]
public string Insert(int startIndex, string value)
```

Summary

Returns a `System.String` equivalent to the current instance with a specified `System.String` inserted at the specified position.

Parameters

Parameter	Description
startIndex	A `System.Int32` containing the index of the insertion.
value	The `System.String` to insert.

Return Value

A new `System.String` that is equivalent to the current string with *value* inserted at index *startIndex.*

Description

In the new string returned by this method, the first character of *value* is at *startIndex,* and all characters in the current string from *startIndex* to the end are inserted in the new string after the last character of *value*.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* is greater than the length of the current instance.

String.Intern(System.String) Method

```
[ILAsm]
.method public hidebysig static string Intern(string str)

[C#]
public static string Intern(string str)
```

Summary

Retrieves the system's reference to a specified `System.String`.

Parameters

Parameter	Description
str	A `System.String`.

Return Value

The `System.String` reference to *str*.

Description

Instances of each unique literal string constant declared in a program, as well as any unique instance of `System.String` you add programmatically are kept in a table, called the "intern pool".

The intern pool conserves string storage. If a literal string constant is assigned to several variables, each variable is set to reference the same constant in the intern pool instead of referencing several different instances of `System.String` that have identical values.

This method looks up a specified string in the intern pool. If the string exists, a reference to it is returned. If it does not exist, an instance equal to the specified string is added to the intern pool and a reference that instance is returned.

Exceptions

Exception	Condition
System.ArgumentNullException	*str* is a null reference.

Example

The following example demonstrates the `System.String.Intern` method.

[C#]

```
using System;
using System.Text;
public class StringExample {
 public static void Main() {

     String s1 = "MyTest";
        String s2 = new
StringBuilder().Append("My").Append("Test").ToString();
        String s3 = String.Intern(s2);

        Console.WriteLine(Object.ReferenceEquals(s1, s2));
//different
        Console.WriteLine(Object.ReferenceEquals(s1, s3));    //the
same
    }
}
```

The output is

```
False


True
```

String.IsInterned(System.String) Method

```
[ILAsm]
.method public hidebysig static string IsInterned(string str)

[C#]
public static string IsInterned(string str)
```

Summary

Retrieves a reference to a specified `System.String`.

Parameters

Parameter	Description
str	A `System.String`.

Return Value

A `System.String` reference to *str* if it is in the system's intern pool; otherwise, a null reference.

Description

Instances of each unique literal string constant declared in a program, as well as any unique instance of `System.String` you add programmatically are kept in a table, called the "intern pool".

The intern pool conserves string storage. If a literal string constant is assigned to several variables, each variable is set to reference the same constant in the intern pool instead of referencing several different instances of `System.String` that have identical values.

[*Note:* This method does not return a `System.Boolean` value, but can still be used where a `System.Boolean` is needed.]

Exceptions

Exception	Condition
System.ArgumentNullException	*str* is a null reference.

Example

The following example demonstrates the `System.String.IsInterned` method.

[C#]

```
using System;
using System.Text;

public class StringExample {
    public static void Main() {

        String s1 = new
StringBuilder().Append("My").Append("Test").ToString();

        Console.WriteLine(String.IsInterned(s1) != null);
    }
}
```

The output is

```
True
```

String.Join(System.String, System.String[]) Method

```
[ILAsm]
.method public hidebysig static string Join(string separator,
string[] value)

[C#]
public static string Join(string separator, string[] value)
```

Summary

Concatenates the elements of a specified `System.String` array, inserting a separator string between each element pair and yielding a single concatenated string.

Parameters

Parameter	Description
separator	A `System.String`.
value	A `System.String` array.

Return Value

A `System.String` consisting of the elements of *value* separated by instances of the *separator* string.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

Example

The following example demonstrates the `System.String.Join` method.

[C#]

```
using System;
public class StringJoin {
 public static void Main() {
 String[] strAry = { "Red", "Green", "Blue" };
 Console.WriteLine( String.Join( ":: ", strAry ) );
 }
```

```
}
```
The output is

```
Red:: Green:: Blue
```

String.Join(System.String, System.String[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig static string Join(string separator,
string[] value, int32 startIndex, int32 count)

[C#]
public static string Join(string separator, string[] value, int
startIndex, int count)
```

Summary

Concatenates a specified separator `System.String` between the elements of a specified `System.String` array, yielding a single concatenated string.

Parameters

Parameter	Description
separator	A `System.String`.
value	A `System.String` array.
startIndex	A `System.Int32` containing the first array element in *value* to join.
count	A `System.Int32` containing the number of elements in *value* to join.

Return Value

A `System.String` consisting of the specified strings in *value* joined by *separator*. Returns `System.String.Empty` if *count* is zero, *value* has no elements, or *separator* and all the elements of *value* are `Empty`.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* plus *count* is greater than the number of elements in *value*.

Example

The following example demonstrates the `System.String.Join` method.

[C#]

```
using System;
public class StringJoin {
 public static void Main() {
 String[] strAry = { "Red", "Green", "Blue" };
 Console.WriteLine( String.Join( ":: ", strAry, 1, 2 ) );
 }
}
```

The output is

```
Green:: Blue
```

String.LastIndexOf(System.String, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(string value,
int32 startIndex)

[C#]
public int LastIndexOf(string value, int startIndex)
```

Summary

Returns the index of the last occurrence of a specified `System.String` within the current instance, starting at a given position.

Parameters

Parameter	Description
value	A `System.String`.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the last substring in the current instance that is equal to *value*.	*value* was found.
-1	*value* was not found.

Description

This method searches for the last occurrence of the specified `System.String` between the start of the string and the indicated index.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* is less than zero or greater than or equal to the length of the current instance.

String.LastIndexOf(System.String, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(string value,
int32 startIndex, int32 count)

[C#]
public int LastIndexOf(string value, int startIndex, int count)
```

Summary

Returns the index of the last occurrence of a specified `System.String` in the provided range of the current instance.

Parameters

Parameter	Description
value	The substring to search for.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the range of the current instance at which to end searching.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the last substring in the current instance that is equal to *value*.	*value* was found.
-1	*value* was not found.

Description

The search begins at *startIndex* and continues until *startIndex* - *count* + 1.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *count* is less than zero. -or- *startIndex* - *count* is smaller than -1.

String.LastIndexOf(System.String) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(string value)

[C#]
public int LastIndexOf(string value)
```

Summary

Returns the index of the last occurrence of a specified `System.String` within the current instance.

Parameters

Parameter	Description
value	A `System.String`.

Return Value

A `System.Int32` that indicates the result of the search for *value* in the current instance as follows:

Return Value	Description
A zero-based number equal to the index of the start of the last substring in the current instance that is equal to *value*.	*value* was found.
-1	*value* was not found.

Description

The search is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

String.LastIndexOf(System.Char, System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(valuetype
System.Char value, int32 startIndex, int32 count)

[C#]
public int LastIndexOf(char value, int startIndex, int count)
```

Summary

Returns the index of the last occurrence of a specified character in the provided range of the current instance.

Parameters

Parameter	Description
value	A Unicode character to locate.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the range of the current instance at which to end searching.

Return Value

A `System.Int32` containing the index of the last occurrence of *value* in the current instance if found between *startIndex* and (*startIndex* - *count* + 1); otherwise, -1.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *count* is less than zero. -or-

	startIndex - *count* is less than -1.

String.LastIndexOf(System.Char, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(valuetype
System.Char value, int32 startIndex)

[C#]
public int LastIndexOf(char value, int startIndex)
```

Summary

Returns the index of the last occurrence of a specified character within the current instance.

Parameters

Parameter	Description
value	A Unicode character to locate.
startIndex	A `System.Int32` containing the index in the current instance from which to begin searching.

Return Value

A `System.Int32` containing the index of the last occurrence of *value* in the current instance, if found; otherwise, -1.

Description

This method searches for the last occurrence of the specified character between the start of the string and the indicated index.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* is less than zero or greater than the length of the current instance.

Example

The following example demonstrates the `System.String.LastIndexOf` method.

[C#]

```
using System;
public class StringLastIndexOfTest {
   public static void Main() {
      String str = "aa bb cc dd";

      Console.WriteLine( str.LastIndexOf('d', 8) );
      Console.WriteLine( str.LastIndexOf('b', 8) );
   }
}
```

The output is

```
-1


4
```

String.LastIndexOf(System.Char) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOf(valuetype
System.Char value)

[C#]
public int LastIndexOf(char value)
```

Summary

Returns the index of the last occurrence of a specified character within the current instance.

Parameters

Parameter	Description
value	The Unicode character to locate.

Return Value

A `System.Int32` containing the index of the last occurrence of *value* in the current instance, if found; otherwise, -1.

Description

This method is case-sensitive.

String.LastIndexOfAny(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOfAny(char[] anyOf)

[C#]
public int LastIndexOfAny(char[] anyOf)
```

Summary

Returns the index of the last occurrence of any element of a specified array of characters in the current instance.

Parameters

Parameter	Description
anyOf	An array of Unicode characters.

Return Value

A `System.Int32` containing the index of the last occurrence of any element of *anyOf* in the current instance, if found; otherwise, -1.

Description

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.

String.LastIndexOfAny(System.Char[], System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOfAny(char[] anyOf,
int32 startIndex)

[C#]
public int LastIndexOfAny(char[] anyOf, int startIndex)
```

Summary

Returns the index of the last occurrence of any element of a specified array of characters in the current instance.

Parameters

Parameter	Description
anyOf	An array of Unicode characters.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.

Return Value

A `System.Int32` containing the index of the last occurrence of any element of *anyOf* in the current instance, if found; otherwise, -1.

Description

This method searches for the last occurrence of the specified characters between the start of the string and the indicated index.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* is less than zero or greater than or equal to the length of the current instance.

String.LastIndexOfAny(System.Char[], System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance int32 LastIndexOfAny(char[] anyOf,
int32 startIndex, int32 count)

[C#]
public int LastIndexOfAny(char[] anyOf, int startIndex, int count)
```

Summary

Returns the index of the last occurrence of any of specified characters in the provided range of the current instance.

Parameters

Parameter	Description
anyOf	An array of Unicode characters.
startIndex	A `System.Int32` containing the index of the current instance from which to start searching.
count	A `System.Int32` containing the range of the current instance at which to end searching.

Return Value

A `System.Int32` containing the index of the last occurrence of any element of *anyOf* if found between *startIndex* and (*startIndex* - *count* + 1); otherwise, -1.

Description

The search begins at *startIndex* and continues until *startIndex* - *count* + 1. The character at *startIndex* - *count* is not included in the search.

This method is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*anyOf* is a null reference.
System.ArgumentOutOfRangeException	*startIndex* or *count* is less than zero. -or-

	startIndex - *count* is smaller than -1.

String.op_Equality(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static specialname bool op_Equality(string
a, string b)

[C#]
public static bool operator ==(String a, String b)
```

Summary

Returns a `System.Boolean` value indicating whether the two specified string values are equal to each other.

Parameters

Parameter	Description
a	The first `System.String` to compare.
b	The second `System.String` to compare.

Return Value

`true` if *a* and *b* represent the same string value; otherwise, `false`.

String.op_Inequality(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig static specialname bool
op_Inequality(string a, string b)

[C#]
public static bool operator !=(String a, String b)
```

Summary

Returns a `System.Boolean` value indicating whether the two string values are not equal to each other.

Parameters

Parameter	Description
a	The first `System.String` to compare.
b	The second `System.String` to compare.

Return Value

`true` if *a* and *b* do not represent the same string value; otherwise, `false`.

String.PadLeft(System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string PadLeft(int32 totalWidth)

[C#]
public string PadLeft(int totalWidth)
```

Summary

Right-aligns the characters in the current instance, padding with spaces on the left, for a specified total length.

Parameters

Parameter	Description
totalWidth	A `System.Int32` containing the number of characters in the resulting string.

Return Value

A new `System.String` that is equivalent to the current instance right-aligned and padded on the left with as many spaces as needed to create a length of *totalWidth*. If *totalWidth* is less than the length of the current instance, returns a new `System.String` that is identical to the current instance.

Description

[*Note:* A space in Unicode format is defined as the hexadecimal value 0x20.]

Exceptions

Exception	Condition
System.ArgumentException	*totalWidth* is less than zero.

String.PadLeft(System.Int32, System.Char) Method

```
[ILAsm]
.method public hidebysig instance string PadLeft(int32 totalWidth,
valuetype System.Char paddingChar)

[C#]
public string PadLeft(int totalWidth, char paddingChar)
```

Summary

Right-aligns the characters in the current instance, padding on the left with a specified Unicode character, for a specified total length.

Parameters

Parameter	Description
totalWidth	A `System.Int32` containing the number of characters in the resulting string.
paddingChar	A `System.Char` that specifies the padding character to use.

Return Value

A new `System.String` that is equivalent to the current instance right-aligned and padded on the left with as many *paddingChar* characters as needed to create a length of *totalWidth*. If *totalWidth* is less than the length of the current instance, returns a new `System.String` that is identical to the current instance.

Exceptions

Exception	Condition
System.ArgumentException	*totalWidth* is less than zero.

String.PadRight(System.Int32, System.Char) Method

```
[ILAsm]
.method public hidebysig instance string PadRight(int32 totalWidth,
valuetype System.Char paddingChar)

[C#]
public string PadRight(int totalWidth, char paddingChar)
```

Summary

Left-aligns the characters in the current instance, padding on the right with a specified Unicode character, for a specified total number of characters.

Parameters

Parameter	Description
totalWidth	A `System.Int32` containing the number of characters in the resulting string.
paddingChar	A `System.Char` that specifies the padding character to use.

Return Value

A new `System.String` that is equivalent to the current instance left aligned and padded on the right with as many *paddingChar* characters as needed to create a length of *totalWidth*. If *totalWidth* is less than the length of the current instance, returns a new `System.String` that is identical to the current instance.

Exceptions

Exception	Condition
System.ArgumentException	*totalWidth* is less than zero.

String.PadRight(System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string PadRight(int32 totalWidth)

[C#]
public string PadRight(int totalWidth)
```

Summary

Left-aligns the characters in the current instance, padding with spaces on the right, for a specified total number of characters.

Parameters

Parameter	Description
totalWidth	A `System.Int32` containing the number of characters in the resulting string.

Return Value

A new `System.String` that is equivalent to this instance left aligned and padded on the right with as many spaces as needed to create a length of *totalWidth*. If *totalWidth* is less than the length of the current instance, returns a new `System.String` that is identical to the current instance.

Exceptions

Exception	Condition
System.ArgumentException	*totalWidth* is less than zero.

String.Remove(System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string Remove(int32 startIndex,
int32 count)

[C#]
public string Remove(int startIndex, int count)
```

Summary

Deletes a specified number of characters from the current instance beginning at a specified index.

Parameters

Parameter	Description
startIndex	A `System.Int32` containing the index of the current instance from which to start deleting characters.
count	A `System.Int32` containing the number of characters to delete.

Return Value

A new `System.String` that is equivalent to the current instance without the specified range characters.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* or *count* is less than zero. -or- *startIndex* plus *count* is greater than the length of the current instance.

String.Replace(System.String, System.String) Method

```
[ILAsm]
.method public hidebysig instance string Replace(string oldValue,
string newValue)

[C#]
public string Replace(string oldValue, string newValue)
```

Summary

Replaces all instances of a specified substring within the current instance with another specified string.

Parameters

Parameter	Description
oldValue	A `System.String` containing the string value to be replaced.
newValue	A `System.String` containing the string value to replace all occurrences of *oldValue*. Can be a null reference.

Return Value

A `System.String` equivalent to the current instance with all occurrences of *oldValue* replaced with *newValue*. If the replacement value is a null reference, the specified substring is removed.

String.Replace(System.Char, System.Char) Method

```
[ILAsm]
.method public hidebysig instance string Replace(valuetype
System.Char oldChar, valuetype System.Char newChar)

[C#]
public string Replace(char oldChar, char newChar)
```

Summary

Replaces all instances of a specified Unicode character with another specified Unicode character.

Parameters

Parameter	Description
oldChar	The Unicode character to be replaced.
newChar	The Unicode character to replace all occurrences of *oldChar*.

Return Value

A `System.String` equivalent to the current instance with all occurrences of *oldChar* replaced with *newChar*.

String.Split(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance string[] Split(char[] separator)

[C#]
public string[] Split(params char[] separator)
```

Summary

Returns substrings of the current instance that are delimited by the specified characters.

Parameters

Parameter	Description
separator	A `System.Char` array of delimiters. Can be a null reference.

Return Value

A `System.String` array containing the results of the split operation as follows:

Return Value	Description
A single-element array containing the current instance.	None of the elements of *separator* are contained in the current instance.
A multi-element `System.String` array, each element of which is a substring of the current instance that was delimited by one or more characters in *separator.*	At least one element of *separator* is contained in the current instance.
A multi-element `System.String` array, each element of which is a substring of the current instance that was delimited by white space characters.	The current instance contains white space characters and *separator* is a null reference or an empty array.

Description

`System.String.Empty` is returned for any substring where two delimiters are adjacent or a delimiter is found at the beginning or end of the current instance.

Delimiter characters are not included in the substrings.

String.Split(System.Char[], System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string[] Split(char[] separator,
int32 count)

[C#]
public string[] Split(char[] separator, int count)
```

Summary

Returns substrings of the current instance that are delimited by the specified characters.

Parameters

Parameter	Description
separator	An array of Unicode characters that delimit the substrings in the current instance, an empty array containing no delimiters, or a null reference.
count	A `System.Int32` containing the maximum number of array elements to return.

Return Value

A `System.String` array containing the results of the split operation as follows:

Return Value	Description
A single-element array containing the current instance.	None of the elements of *separator* are contained in the current instance.
A multi-element `System.String` array, each element of which is a substring of the current instance that was delimited by one or more characters in *separator*	At least one element of *separator* is contained in the current instance.
A multi-element `System.String` array, each element of which is a substring of the current instance that was delimited by white space characters.	The current instance contains white space characters and *separator* is a null reference or an empty array.

Description

`System.String.Empty` is returned for any substring where two delimiters are adjacent or a delimiter is found at the beginning or end of the current instance.

Delimiter characters are not included in the substrings.

If there are more substrings in the current instance than the maximum specified number, the first *count* -1 elements of the array contain the first *count* - 1 substrings. The remaining characters in the current instance are returned in the last element of the array.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*count* is negative.

String.StartsWith(System.String) Method

```
[ILAsm]
.method public hidebysig instance bool StartsWith(string value)

[C#]
public bool StartsWith(string value)
```

Summary

Returns a `System.Boolean` value that indicates whether the start of the current instance matches the specified `System.String`.

Parameters

Parameter	Description
value	A `System.String`.

Return Value

`true` if the start of the current instance is equal to *value*; `false` if *value* is not equal to the start of the current instance or is longer than the current instance.

Description

This method compares *value* with the substring at the start of the current instance that has a length of *value*.Length. If *value*.Length is greater than the length of the current instance or the relevant substring of the current instance is not equal to *value*, this method returns `false`; otherwise, this method returns `true`.

The comparison is case-sensitive.

Exceptions

Exception	Condition
System.ArgumentNullException	*value* is a null reference.

String.Substring(System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string Substring(int32 startIndex,
int32 length)

[C#]
public string Substring(int startIndex, int length)
```

Summary

Retrieves a substring from the current instance, starting from a specified index, continuing for a specified length.

Parameters

Parameter	Description
startIndex	A `System.Int32` containing the index of the start of the substring in the current instance.
length	A `System.Int32` containing the number of characters in the substring.

Return Value

A `System.String` containing the substring of the current instance with the specified length that begins at the specified position. Returns `System.String.Empty` if *startIndex* is equal to the length of the current instance and length is zero.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*length* is greater than the length of the current instance. -or- *startIndex* or *length* is less than zero.

String.Substring(System.Int32) Method

```
[ILAsm]
.method public hidebysig instance string Substring(int32 startIndex)

[C#]
public string Substring(int startIndex)
```

Summary

Retrieves a substring from the current instance, starting from a specified index.

Parameters

Parameter	Description
startIndex	A `System.Int32` containing the index of the start of the substring in the current instance.

Return Value

A `System.String` equivalent to the substring that begins at *startIndex* of the current instance. Returns `System.String.Empty` if *startIndex* is equal to the length of the current instance.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* is less than zero or greater than the length of the current instance.

String.System.Collections.Generic.IEnumerable<System.Char>.GetEnumerator() Method

```
[ILAsm]
.method private hidebysig virtual abstract class
System.Collections.Generic.IEnumerator<char>
System.Collections.Generic.IEnumerable<char>.GetEnumerator()

[C#]
IEnumerator<Char> IEnumerable<Char>.GetEnumerator()
```

Summary

This method is implemented to support the `System.Collections.Generics.IEnumerable<System.Char>` interface.

String.System.Collections.IEnumerable.GetEnumerator() Method

```
[ILAsm]
.method private final hidebysig virtual class
System.Collections.IEnumerator
System.Collections.IEnumerable.GetEnumerator()

[C#]
IEnumerator IEnumerable.GetEnumerator()
```

Summary

Implemented to support the `System.Collections.IEnumerable` interface. [Note: For more information, see `System.Collections.IEnumerable.GetEnumerator`.]

String.ToCharArray() Method

```
[ILAsm]
.method public hidebysig instance char[] ToCharArray()

[C#]
public char[] ToCharArray()
```

Summary

Copies the characters in the current instance to a Unicode character array.

Return Value

A `System.Char` array whose elements are the individual characters of the current instance. If the current instance is an empty string, the array returned by this method is empty and has a zero length.

String.ToCharArray(System.Int32, System.Int32) Method

```
[ILAsm]
.method public hidebysig instance char[] ToCharArray(int32
startIndex, int32 length)

[C#]
public char[] ToCharArray(int startIndex, int length)
```

Summary

Copies the characters in a specified substring in the current instance to a Unicode character array.

Parameters

Parameter	Description
startIndex	A `System.Int32` containing the index of the start of a substring in the current instance.
length	A `System.Int32` containing the length of the substring in the current instance.

Return Value

A `System.Char` array whose elements are the *length* number of characters in the current instance, starting from the index *startIndex* in the current instance. If the specified length is zero, the entire string is copied starting from the beginning of the current instance, and ignoring the value of *startIndex*. If the current instance is an empty string, the returned array is empty and has a zero length.

Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	*startIndex* or *length* is less than zero. -or- *startIndex* plus *length* is greater than the length of the current instance.

String.ToLower() Method

```
[ILAsm]
.method public hidebysig instance string ToLower()

[C#]
public string ToLower()
```

Summary

Returns a copy of this `System.String` in lowercase.

Return Value

A `System.String` in lowercase..

Description

This method takes into account the culture (if any) of the current thread.

String.ToString() Method

```
[ILAsm]
.method public hidebysig virtual string ToString()

[C#]
public override string ToString()
```

Summary

Returns a `System.String` representation of the value of the current instance.

Return Value

The current `System.String`.

Description

[*Note:* This method overrides `System.Object.ToString`.]

String.ToString(System.IFormatProvider) Method

```
[ILAsm]
.method public final hidebysig virtual string ToString(class
System.IFormatProvider provider)

[C#]
public string ToString(IFormatProvider provider)
```

Summary

Returns this instance of `String`; no actual conversion is performed.

Parameters

Parameter	Description
provider	(Reserved) A `System.IFormatProvider` interface implementation which supplies culture-specific formatting information.

Return Value

This `String`.

Description

provider is reserved, and does not currently participate in this operation.

String.ToUpper() Method

```
[ILAsm]
.method public hidebysig instance string ToUpper()

[C#]
public string ToUpper()
```

Summary

Returns a copy of the current instance with all elements converted to uppercase, using default properties.

Return Value

A new `System.String` in uppercase.

String.Trim(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance string Trim(char[] trimChars)

[C#]
public string Trim(params char[] trimChars)
```

Summary

Removes all occurrences of a set of characters provided in a character `System.Array` from the beginning and end of the current instance.

Parameters

Parameter	Description
trimChars	An array of Unicode characters. Can be a null reference.

Return Value

A new `System.String` equivalent to the current instance with the characters in *trimChars* removed from its beginning and end. If *trimChars* is a null reference, all of the white space characters are removed from the beginning and end of the current instance.

String.Trim() Method

```
[ILAsm]
.method public hidebysig instance string Trim()

[C#]
public string Trim()
```

Summary

Removes all occurrences of white space characters from the beginning and end of the current instance.

Return Value

A new `System.String` equivalent to the current instance after white space characters are removed from its beginning and end.

String.TrimEnd(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance string TrimEnd(char[] trimChars)

[C#]
public string TrimEnd(params char[] trimChars)
```

Summary

Removes all occurrences of a set of characters specified in a Unicode character `System.Array` from the end of the current instance.

Parameters

Parameter	Description
trimChars	An array of Unicode characters. Can be a null reference.

Return Value

A new `System.String` equivalent to the current instance with characters in *trimChars* removed from its end. If *trimChars* is a null reference, white space characters are removed.

String.TrimStart(System.Char[]) Method

```
[ILAsm]
.method public hidebysig instance string TrimStart(char[] trimChars)

[C#]
public string TrimStart(params char[] trimChars)
```

Summary

Removes all occurrences of a set of characters specified in a Unicode character array from the beginning of the current instance.

Parameters

Parameter	Description
trimChars	An array of Unicode characters or a null reference.

Return Value

A new `System.String` equivalent to the current instance with the characters in *trimChars* removed from its beginning. If *trimChars* is a null reference, white space characters are removed.

String.Chars Property

```
[ILAsm]
.property valuetype System.Char Chars[int32 index] { public
hidebysig specialname instance valuetype System.Char get_Chars(int32
index) }

[C#]
public char this[int index] { get; }
```

Summary

Gets the character at a specified position in the current instance.

Property Value

A Unicode character at the location index in the current instance.

Description

This property is read-only.

index is the position of a character within a string. The first character in the string is at index 0. The length of a string is the number of characters it is made up of. The last accessible *index* of a string instance is its length - 1.

Exceptions

Exception	Condition
System.IndexOutOfRangeException	*index* is greater than or equal to the length of the current instance or less than zero.

String.Length Property

[ILAsm]
```
.property int32 Length { public hidebysig specialname instance int32
get_Length() }
```

[C#]
```
public int Length { get; }
```

Summary

Gets the number of characters in the current instance.

Property Value

A `System.Int32` containing the number of characters in the current instance.

Description

This property is read-only.

Example

The following example demonstrates the `System.String.Length` property.

[C#]

```
using System;
public class StringLengthExample {
 public static void Main() {
 string str = "STRING";
 Console.WriteLine( "The length of string {0} is {1}", str, str.Length
);
 }
}
```

The output is

```
The length of string STRING is 6
```