

System.String Class

```
[ILASM]
.class public sealed serializable String extends
System.Object implements System.ICloneable,
System.IComparable, System.Collections.IEnumerable

[C#]
public sealed class String: ICloneable, IComparable,
IEnumerable
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- *Version:* 1.0.x.x
- *Attributes:*
 - CLSCompliantAttribute(true)

Type Attributes:

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

Implements:

- **System.IComparable**
- **System.ICloneable**
- **System.Collections.IEnumerable**

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

1 **System.String** do not change. Combining operations, such as
2 **System.String.Replace**, cannot alter existing strings. Instead, such
3 operations return a new string that contains the results of the
4 operation, an unchanged string, or the null value. To perform
5 modifications to a **System.String** use the
6 **System.Text.StringBuilder**.

7
8 Implementations of **System.String** are required to contain a variable-
9 length character buffer positioned a fixed number of bytes after the
10 beginning of the String object. [Note: The
11 **System.Runtime.CompilerServices.RuntimeHelpers.OffsetToStringData**
12 method returns the number of bytes between the start of the
13 String object and the character buffer. This information is intended
14 primarily for use by compilers, not application programmers. For
15 additional information, see
16 **System.Runtime.CompilerServices.RuntimeHelpers.OffsetToStringData**.]

17
18
19 [Note: Comparisons and searches are case-sensitive by default and
20 unless otherwise specified, use the culture defined for the current
21 thread to determine the order of the alphabet used by the strings. This
22 information is then used to compare the two strings on a character-by-
23 character basis. Upper case letters evaluate greater than their lower
24 case equivalents.

25
26 The following characters are considered white space when present in a
27 System.String instance: 0x9, 0xA, 0xB, 0xC, 0xD, 0x20, 0xA0,
28 0x2000, 0x2001, 0x2002, 0x2003, 0x2004, 0x2005, 0x2006, 0x2007,
29 0x2008, 0x2009, 0x200A, 0x200B, 0x3000, and 0xFEFF.

30
31 The null character is defined as hexadecimal 0x00.

32
33 The **System.String(System.String)** constructor is omitted for
34 performance reasons. If you need a copy of a **System.String**,
35 consider using **System.String.Copy** or the
36 **System.Text.StringBuilder** class.

37
38 To insert a formatted string representation of an object into a string
39 use the **System.String.Format** methods. These methods take one or
40 more arguments to be formatted and a format string. The format
41 string contains literals and zero or more format specifications in the
42 form { *N* [, *M*] [: *formatSpecifier*] }, where:

- 43 • *N* is a zero-based integer indicating the argument to be
44 formatted. If the actual argument is a null reference, then an
45 empty string is used in its place.
- 46 • *M* is an optional integer indicating the minimum width of the
47 region to contain the formatted value of argument *N*. If the
48 length of the string representation of the value is less than *M*,
49 then the region is padded with spaces. If *M* is negative, the
50 formatted value is left justified in the region; if *M* is positive,
51 then the value is right justified. If *M* is not specified, it is

1 assumed to be zero indicating that neither padding nor
2 alignment is customized. Note that if the length of the
3 formatted value is greater than *M*, then *M* is ignored.

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

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

23
24 To include a curly bracket in a formatted string, specify the bracket
25 twice; for example, specify "{{" to include "{" in the formatted string.
26 See Example 1.

27
28 The **System.Console** class exposes the same functionality as the
29 **System.String.Format** methods via **System.Console.Write** and
30 **System.Console.WriteLine**. The primary difference is that the
31 **System.String.Format** methods return the formatted string, while
32 the **System.Console** methods write the formatted string to a stream.]

33 Example

34 35 Example 1

36
37 The following example demonstrates formatting numeric data types
38 and inserting literal curly brackets into strings.

39 [C#]
40

```
41 using System;  
42 class StringFormatTest {  
43     public static void Main() {  
44         decimal dec = 1.99999m;  
45         double doub = 1.0000000001;  
46  
47         string somenums = String.Format("Some formatted  
48 numbers: dec={0,15:E} doub={1,20}", dec, doub);
```

```

1         Console.WriteLine(somenums);
2
3         string curlies = "Literal curly brackets: {{ and }}"
4 and {{0}}";
5         Console.WriteLine(curlies);
6
7         object nullObject = null;
8         string embeddedNull = String.Format("A null
9 argument looks like: {0}", nullObject);
10        Console.WriteLine(embeddedNull);
11    }
12 }
13

```

14 The output is

```

15
16 Some formatted numbers: dec= 1.999990E+000 doub=
17 1.0000000001
18 Literal curly brackets: {{ and }} and {{0}}
19 A null argument looks like:
20

```

21 Example 2

22
23 The following example demonstrates how formatting works if
24 **System.IFormattable** is or is not implemented by an argument to
25 the **System.String.Format** method. Note that the format specifier is
26 ignored if the argument does not implement **System.IFormattable**.

27
28 [C#]

```

29 using System;
30 class StringFormatTest {
31     public class DefaultFormatEleven {
32         public override string ToString() {
33             return "11 string";
34         }
35     }
36     public class FormattableEleven:IFormattable {
37         // The IFormattable ToString implementation.
38         public string ToString(string format,
39 IFormatProvider formatProvider) {
40             Console.Write("[IFormattable called] ");
41             return 11.ToString(format, formatProvider);
42         }

```

```

1          // Override Object.ToString to show that it is not
2 called.
3          public override string ToString() {
4              return "Formatted 11 string";
5          }
6      }
7
8      public static void Main() {
9          DefaultFormatEleven def11 = new DefaultFormatEleven
10 ();
11          FormattableEleven for11 = new
12 FormattableEleven();
13          string def11string = String.Format("{0}",def11);
14          Console.WriteLine(def11string);
15          // The format specifier x is ignored.
16          def11string = String.Format("{0,15:x}", def11);
17          Console.WriteLine(def11string);
18
19          string form11string = String.Format("{0}",for11);
20          Console.WriteLine(form11string);
21          form11string = String.Format("{0,15:x}",for11);
22          Console.WriteLine(form11string);
23      }
24 }

```

25 The output is

```

26
27 11 string
28     11 string
29 [IFormattable called] 11
30 [IFormattable called]          b
31
32

```

1 String(System.Char*) Constructor

```
2 [ILASM]
3 public rtspecialname specialname instance void .ctor(class
4 System.Char* value)
5
6 [C#]
7 unsafe public String(char* value)
```

7 Summary

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

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13 Parameter	14 Description
<i>value</i>	A pointer to a null-terminated array of Unicode characters. If <i>value</i> is a null pointer, System.String.Empty is created.

15 Description

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

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

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

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

25
26
27
28
29

1 String(System.Char*, System.Int32, 2 System.Int32) Constructor

```
3 [ILASM]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.Char* value, int32 startIndex, int32 length)  
  
6 [C#]  
7 unsafe public String(char* value, int startIndex, int  
8 length)
```

9 Summary

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

14 Type Attributes:

- 15 • CLSCompliantAttribute(false)

16 Parameters

17
18

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

19

20 Description

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

24

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

27

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

32

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

1 **Exceptions**
2
3

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>length</i> is less than zero. -or- <i>value</i> is a null pointer and <i>length</i> is not zero.

4
5
6

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

3 **String(System.SByte*, System.Int32,** 4 **System.Int32, System.Text.Encoding)** 5 **Constructor**

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

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

13 **Summary**

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

18 **Type Attributes:**

- 19 • CLSCompliantAttribute(false)

20 **Parameters**

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

24 **Description**

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

27 **Exceptions**

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>length</i> is less than zero. -or- <i>value</i> is a null pointer and <i>length</i> is not zero.

1
2
3

1 String(System.Char[], System.Int32, 2 System.Int32) Constructor

```
3 [ILASM]  
4 public rtspecialname specialname instance void .ctor(class  
5 System.Char[] value, int32 startIndex, int32 length)  
  
6 [C#]  
7 public String(char[] value, int startIndex, int length)
```

8 Summary

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

12 Parameters

13
14

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

15

16 Description

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

19 Exceptions

20
21

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

22
23
24

1 String(System.Char[]) Constructor

```
2 [ILASM]  
3 public rtspecialname specialname instance void .ctor(class  
4 System.Char[] value)  
  
5 [C#]  
6 public String(char[] value)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>value</i>	An array of Unicode characters.

13
14

14 Description

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

17

1 String(System.Char, System.Int32)

2 Constructor

```
3 [ILASM]  
4 public rtspecialname specialname instance void  
5 .ctor(valuetype System.Char c, int32 count)  
6  
7 [C#]  
8 public String(char c, int count)
```

8 Summary

9 Constructs and initializes a new instance of **System.String**.

10 Parameters

11
12

Parameter	Description
<i>c</i>	A System.Char .
<i>count</i>	A System.Int32 containing the number of occurrences of <i>c</i> .

13

14 Description

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

16 Exceptions

17
18

Exception	Condition
System.ArgumentOutOfRangeException	<i>count</i> is less than zero.

19

20 Example

21

22 The following example demonstrates using this constructor.

23
24

```
25 [C#]  
26 using System;  
27  
28 public class StringExample {  
29     public static void Main() {  
30  
31         string s = new String('a', 10);  
32  
33         Console.WriteLine(s);  
34     }  
35 }
```

1

2

3

4

The output is

aaaaaaaaaa

5

1 String.Empty Field

```
2 [ILASM]  
3 .field public static initOnly string Empty  
4  
5 [C#]  
6 public static readonly string Empty
```

6 Summary

7 A constant string representing the empty string.

8 Description

9 This field is read-only.

10

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

12

1 String.Clone() Method

```
2 [ILASM]  
3 .method public final hidebysig virtual object Clone()  
4 [C#]  
5 public object Clone()
```

6 Summary

7 Returns a reference to the current instance of **System.String**.

8 Return Value

9

10 A reference to the current instance of **System.String**.

11 Description

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

16
17 This method is implemented to support the **System.ICloneable**
18 interface.]

19

1 String.Compare(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static int32 Compare(string strA,  
5 string strB)  
  
6 [C#]  
7 public static int Compare(string strA, string strB)
```

8 Summary

9 Compares two **System.String** objects in a case sensitive manner.

10 Parameters

11
12

Parameter	Description
<i>strA</i>	The first System.String to compare. Can be a null reference.
<i>strB</i>	The second System.String to compare. Can be a null reference.

13
14
15

Return Value

16 A **System.Int32** containing a value that reflects the sort order of the
17 two specified strings. The following table defines conditions under
18 which the returned value is a negative number, zero, or a positive
19 number.

Value	Meaning
Any negative number	<i>strA</i> is lexicographically < <i>strB</i> , or <i>strA</i> is a null reference.
Zero	<i>strA</i> is lexicographically == <i>strB</i> , or both <i>strA</i> and <i>strB</i> are null references.
Any positive number	<i>strA</i> is lexicographically > <i>strB</i> , or <i>strB</i> is a null reference.

20

21 Description

22 This method performs a case-sensitive operation.

23
24
25
26
27

[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. Upper case letters evaluate greater than their lower case equivalents.

1
2
3
4
5

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.]

1 String.Compare(System.String, 2 System.String, System.Boolean) Method

```
3 [ILASM]  
4 .method public hidebysig static int32 Compare(string strA,  
5 string strB, bool ignoreCase)  
  
6 [C#]  
7 public static int Compare(string strA, string strB, bool  
8 ignoreCase)
```

9 Summary

10 Returns sort order of two **System.String** objects.

11 Parameters

12
13

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

14
15
16

15 Return Value

17 A **System.Int32** containing a value that reflects the sort order of the
18 two specified strings. The following table defines the conditions under
19 which the returned value is a negative number, zero, or a positive
20 number.

Value	Meaning
Any negative number	<i>strA</i> is < <i>strB</i> , or <i>strA</i> is a null reference.
Zero	<i>strA</i> == <i>strB</i> , or both <i>strA</i> and <i>strB</i> are null references.
Any positive number	<i>strA</i> is > <i>strB</i> , or <i>strB</i> is a null reference.

21
22

22 Description

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

1
2 The method uses the culture of the current thread to determine the
3 ordering of individual characters. The two strings are compared on a
4 character-by-character basis.

5
6 **String.Compare**(*strA*, *strB*, **false**) is equivalent to
7 **String.Compare**(*strA*, *strB*.)]

8 Example

9

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

```
12 [C#]  
13  
14 using System;  
15 public class StringComparisonExample {  
16     public static void Main() {  
17         string strA = "A STRING";  
18         string strB = "a string";  
19         int first = String.Compare(strA, strB, true);  
20         int second = String.Compare(strA, strB, false);  
21         Console.WriteLine("When 'A STRING' is compared to 'a  
22 string' in a case-insensitive manner, the return value is  
23 {0}.", first);  
24         Console.WriteLine("When 'A STRING' is compared to 'a  
25 string' in a case-sensitive manner, the return value is  
26 {0}.", second);  
27     }  
28 }  
29
```

30 The output is

31
32 When 'A STRING' is compared to 'a string' in a case-
33 insensitive manner, the return value is 0.

34
35
36 When 'A STRING' is compared to 'a string' in a case-
37 sensitive manner, the return value is 1.

38

39

1 String.Compare(System.String, 2 System.Int32, System.String, 3 System.Int32, System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig static int32 Compare(string strA,  
6 int32 indexA, string strB, int32 indexB, int32 length)  
  
7 [C#]  
8 public static int Compare(string strA, int indexA, string  
9 strB, int indexB, int length)
```

10 Summary

11 Compares substrings of two strings.

12 Parameters

13
14

Parameter	Description
<i>strA</i>	The first System.String to compare.
<i>indexA</i>	A System.Int32 containing the starting index of the substring within <i>strA</i> .
<i>strB</i>	The second System.String to compare.
<i>indexB</i>	A System.Int32 containing the starting index of the substring within <i>strB</i> .
<i>length</i>	A System.Int32 containing the number of characters in the substrings to compare. If <i>length</i> is zero, then zero is returned.

15
16
17

16 Return Value

18 A **System.Int32** containing a value that reflects the sort order of
19 substrings of the two specified strings. The following table defines the
20 conditions under which the returned value is a negative number, zero,
21 or a positive number.

Value	Meaning
Any negative number	The substring in <i>strA</i> is < the substring in <i>strB</i> , or <i>strA</i> is a null reference.
Zero	The substring in <i>strA</i> == the substring in <i>strB</i> , or both <i>strA</i> and <i>strB</i> are null references.
Any positive number	The substring in <i>strA</i> is > the substring in <i>strB</i> , or <i>strB</i> is a null reference.

1

2 **Description**

3 [Note: The result of comparing any **System.String** (including the
4 empty string) to a null reference is greater than zero. The result of
5 comparing two null references is zero. Upper case letters evaluate
6 greater than their lower case equivalents.
7

8 The method uses the culture of the current thread to determine the
9 ordering of individual characters. The two strings are compared on a
10 character-by-character basis.]

11 **Exceptions**

12

13

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

14

15 **Example**

16

17 The following example demonstrates comparing substrings.

18

19

```

20 using System;
21 public class StringComparisonExample {
22     public static void Main() {
23         string strA = "A string";
24         string strB = "B ring";
25         int first = String.Compare(strA, 4, strB, 2, 3);
26         int second = String.Compare(strA, 3, strB, 3, 3);
27         Console.WriteLine("When the substring 'rin' of 'A string'
28 is compared to the substring 'rin' of 'B ring', the return
29 value is {0}.", first);
30         Console.WriteLine("When the substring 'tri' of 'A string'
31 is compared to the substring 'ing' of 'B ring', the return
32 value is {0}.", second);
33     }
34 }

```

1

2

The output is

3

4

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

5

6

7

8

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

9

10

11

1 **String.Compare(System.String,**
2 **System.Int32, System.String,**
3 **System.Int32, System.Int32,**
4 **System.Boolean) Method**

```
5 [ILASM]  
6 .method public hidebysig static int32 Compare(string strA,  
7 int32 indexA, string strB, int32 indexB, int32 length, bool  
8 ignoreCase)
```

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

12 **Summary**

13 Compares substrings of two strings.

14 **Parameters**

Parameter	Description
<i>strA</i>	The first System.String containing a substring to compare.
<i>indexA</i>	A System.Int32 containing the starting index of the substring within <i>strA</i> .
<i>strB</i>	The second System.String containing a substring to compare.
<i>indexB</i>	A System.Int32 containing the starting index of the substring within <i>strB</i> .
<i>length</i>	A System.Int32 containing the number of characters in the substrings to compare. If <i>length</i> is zero, then zero is returned.
<i>ignoreCase</i>	A System.Boolean indicating if the comparison is case-insensitive. If <i>ignoreCase</i> is true , the comparison is case-insensitive. If <i>ignoreCase</i> is false , the comparison is case-sensitive, and upper case letters evaluate greater than their lower case equivalents.

17
18 **Return Value**

19
20 A **System.Int32** containing a value that reflects the sort order of
21 substrings of the two specified strings. The following table defines the
22 conditions under which the returned value is a negative number, zero,
23 or a positive number.

Value Type	Condition
Any negative	The substring in <i>strA</i> is < the substring in <i>strB</i> . or <i>strA</i> is a null

number	reference.
Zero	The substring in <i>strA</i> == the substring in <i>strB</i> , or both <i>strA</i> and <i>strB</i> are null references.
Any positive number	The substring in <i>strA</i> is > the substring in <i>strB</i> , or <i>strB</i> is a null reference.

1

2 Description

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

7

8 The method uses the culture of the current thread to determine the
9 ordering of individual characters. The two strings are compared on a
10 character-by-character basis.]

11 Exceptions

12

13

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

14

15 Example

16

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

19

20

[C#]

21

22

23

24

25

26

```
using System;
public class StringComparisonExample {
    public static void Main() {
        string strA = "STRING A";
        string strB = "string b";
        int first = String.Compare(strA, strB, true);
```

```
1     int second = String.Compare(strA, 0, strB, 0, 4, true);
2     int third = String.Compare(strA, 0, strB, 0, 4, false);
3     Console.WriteLine("When the string 'STRING A' is compared
4 to the string 'string b' in a case-insensitive manner, the
5 return value is {0}.", first);
6     Console.WriteLine("When the substring 'STRI' of 'STRING A'
7 is compared to the substring 'stri' of 'string b' in a
8 case-insensitive manner, the return value is {0}.",
9 second);
10    Console.WriteLine("When the substring 'STRI' of 'STRING A'
11 is compared to the substring 'stri' of 'string b' in a
12 case-sensitive manner, the return value is {0}.", third);
13    }
14    }
15
```

16 The output is

```
17
18 When the string 'STRING A' is compared to the string
19 'string b' in a case-insensitive manner, the return value
20 is -1.
```

```
21
22
23 When the substring 'STRI' of 'STRING A' is compared to the
24 substring 'stri' of 'string b' in a case-insensitive
25 manner, the return value is 0.
```

```
26
27
28 When the substring 'STRI' of 'STRING A' is compared to the
29 substring 'stri' of 'string b' in a case-sensitive manner,
30 the return value is 1.
```

31

32

1 String.CompareOrdinal(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static int32 CompareOrdinal(string  
5 strA, string strB)  
  
6 [C#]  
7 public static int CompareOrdinal(string strA, string strB)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>strA</i>	The first System.String to compare.
<i>strB</i>	The second System.String to compare.

14
15
16

15 Return Value

17 A **System.Int32** containing a value that reflects the sort order of the
18 two specified strings. The following table defines the conditions under
19 which the returned value is a negative number, zero, or a positive
20 number.

Permission	Description
Any negative number	<i>strA</i> is < <i>strB</i> , or <i>strA</i> is a null reference.
Zero	<i>strA</i> == <i>strB</i> , or both <i>strA</i> and <i>strB</i> are null references.
Any positive number	<i>strA</i> is > <i>strB</i> , or <i>strB</i> is a null reference.

21

22 Description

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

27
28
29
30

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.]

1 String.CompareOrdinal(System.String, 2 System.Int32, System.String, 3 System.Int32, System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig static int32 CompareOrdinal(string  
6 strA, int32 indexA, string strB, int32 indexB, int32  
7 length)
```

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

11 Summary

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

14 Parameters

15
16

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

17
18
19

Return Value

20 A **System.Int32** containing a value that reflects the sort order of the
21 two specified strings. The following table defines the conditions under
22 which the returned value is a negative number, zero, or a positive
23 number.

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

1

2 **Description**

3 [Note: The result of comparing any **System.String** (including the
4 empty string) to a null reference is greater than zero. The result of
5 comparing two null references is zero. Upper case letters evaluate
6 greater than their lower case equivalents.
7

8 The method uses the culture of the current thread to determine the
9 ordering of individual characters. The two strings are compared on a
10 character-by-character basis.]

11 **Exceptions**

12

13

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

14

15

16

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
<i>value</i>	The System.Object to compare to the current instance.

Return Value

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

Value	Condition
Any negative number	The current instance is lexicographically < <i>value</i> .
Zero	The current instance is lexicographically == <i>value</i> .
Any positive number	The current instance is lexicographically > <i>value</i> , or <i>value</i> 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. The result of comparing two null references is zero. Upper case letters evaluate greater than their lower case equivalents.

1 The method uses the culture of the current thread to determine the
2 ordering of individual characters. The two strings are compared on a
3 character-by-character basis.
4
5 This method is implemented to support the **System.IComparable**
6 interface.]

7 **Exceptions**
8
9

Exception	Condition
System.ArgumentException	<i>value</i> is not a System.String .

10
11
12

1 String.Concat(System.Object, 2 System.Object) Method

```
3 [ILASM]  
4 .method public hidebysig static string Concat(object arg0,  
5 object arg1)  
  
6 [C#]  
7 public static string Concat(object arg0, object arg1)
```

8 Summary

9 Concatenates the **System.String** representations of two specified
10 objects.

11 Parameters

12
13

Parameter	Description
<i>arg0</i>	The first System.Object to concatenate.
<i>arg1</i>	The second System.Object to concatenate.

14
15
16

15 Return Value

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

19 Description

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

21

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

24

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

28 Example

29

30 The following example demonstrates concatenating two objects.

31

32 [C#]

```
33 using System;  
34 public class StringConcatExample {
```

```
1     public static void Main() {  
2         string str = String.Concat('c', 32);  
3         Console.WriteLine("The concatenated Objects are: {0}",  
4         str);  
5     }  
6 }  
7
```

8 The output is

9
10 The concatenated Objects are: c32

11

1 String.Concat(System.Object, 2 System.Object, System.Object) Method

```
3 [ILASM]  
4 .method public hidebysig static string Concat(object arg0,  
5 object arg1, object arg2)  
  
6 [C#]  
7 public static string Concat(object arg0, object arg1,  
8 object arg2)
```

9 Summary

10 Concatenates the **System.String** representations of three specified
11 objects, in order provided.

12 Parameters

13
14

Parameter	Description
<i>arg0</i>	The first System.Object to concatenate.
<i>arg1</i>	The second System.Object to concatenate.
<i>arg2</i>	The third System.Object to concatenate.

15
16
17

16 Return Value

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

20 Description

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

24
25 This version of **System.String.Concat** is equivalent to
26 **String.Concat**(*arg0.ToString()*, *arg1.ToString()*, *arg2.ToString()*).

27 Example

28

29 The following example demonstrates concatenating three objects.

30
31

```
[C#]  
  
32 using System;  
33 public class StringConcatExample {  
34     public static void Main() {
```

```
1     string str = String.Concat('c', 32, "String");
2     Console.WriteLine("The concatenated Objects are: {0}",
3     str);
4     }
5     }
6
```

7 The output is

8
9 The concatenated Objects are: c32String

10

String.Concat(System.Object[]) Method

```
[ILASM]
.method public hidebysig static string Concat(class
System.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
<i>args</i>	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	<i>args</i> is a null reference.

Example

The following example demonstrates concatenating an array of objects.

```
[C#]
using System;
public class StringConcatExample {
```

```
1     public static void Main() {
2         string str = String.Concat('c', 32, "String");
3         Console.WriteLine("The concatenated Object array is: {0}",
4 str);
5     }
6 }
7
```

8 The output is

9
10 The concatenated Object array is: c32String

11

1 String.Concat(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static string Concat(string str0,  
5 string str1)  
  
6 [C#]  
7 public static string Concat(string str0, string str1)
```

8 Summary

9 Concatenates two specified instances of **System.String**.

10 Parameters

11
12

Parameter	Description
<i>str0</i>	The first System.String to concatenate.
<i>str1</i>	The second System.String to concatenate.

13
14
15

14 Return Value

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

17 Description

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

19 Example

20

21 The following example demonstrates concatenating two strings.

22
23

```
[C#]  
  
24 using System;  
25 public class StringConcatExample {  
26     public static void Main() {  
27         string str = String.Concat("one", "two");  
28         Console.WriteLine("The concatenated strings are: {0}",  
29 str);  
30     }  
31 }  
32
```

1 The output is
2
3 The concatenated strings are: onetwo

4

1 String.Concat(System.String, 2 System.String, System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static string Concat(string str0,  
5 string str1, string str2)  
  
6 [C#]  
7 public static string Concat(string str0, string str1,  
8 string str2)
```

9 Summary

10 Concatenates three specified instances of **System.String**.

11 Parameters

12
13

Parameter	Description
<i>str0</i>	The first System.String to concatenate.
<i>str1</i>	The second System.String to concatenate.
<i>str2</i>	The third System.String to concatenate.

14
15
16

15 Return Value

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

18 Description

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

20 Example

21

22 The following example demonstrates concatenating three strings.

23
24

```
[C#]  
  
25 using System;  
26 public class StringConcatExample {  
27     public static void Main() {  
28         string str = String.Concat("one", "two", "three");  
29         Console.WriteLine("The concatenated strings are: {0}",  
30 str);  
31     }  
32 }  
33
```

1
2
3

The output is

The concatenated strings are: onetwothree

4

String.Concat(System.String[]) Method

```
[ILASM]
.method public hidebysig static string Concat(class
System.String[] values)

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

Summary

Concatenates the elements of a specified array.

Parameters

Parameter	Description
<i>values</i>	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	<i>values</i> 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);
    }
}
```

```
1     }  
2     }  
3
```

```
4     The output is
```

```
5
```

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

```
7
```

1 String.Copy(System.String) Method

```
2 [ILASM]  
3 .method public hidebysig static string Copy(string str)  
4 [C#]  
5 public static string Copy(string str)
```

6 Summary

7 Creates a new instance of **System.String** with the same value as a
8 specified instance of **System.String**.

9 Parameters

10
11

Parameter	Description
<i>str</i>	The System.String to be copied.

12
13
14

13 Return Value

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

16 Exceptions

17
18

Exception	Condition
System.ArgumentNullException	<i>str</i> is a null reference.

19
20
21

20 Example

22 The following example demonstrates copying strings.

23
24

```
24 [C#]  
25 using System;  
26 public class StringCopyExample {  
27     public static void Main() {  
28         string strA = "string";  
29         Console.WriteLine("The initial string, strA, is '{0}'.",  
30 strA);  
31         string strB = String.Copy(strA);  
32         strA = strA.ToUpper();  
33         Console.WriteLine("The copied string, strB, before  
34 strA.ToUpper, is '{0}'.", strB);  
35         Console.WriteLine("The initial string after StringCopy and  
36 ToUpper, is '{0}'.", strA);
```

```
1     Console.WriteLine("The copied string, strB, after
2     strA.ToUpper, is '{0}'.", strB);
3     }
4     }
5
```

6 The output is

7
8 The initial string, strA, is 'string'.

9
10
11 The copied string, strB, before strA.ToUpper, is 'string'.

12
13
14 The initial string after StringCopy and ToUpper, is
15 'STRING'.

16
17
18 The copied string, strB, after strA.ToUpper, is 'string'.
19

20

1 String.CopyTo(System.Int32, 2 System.Char[], System.Int32, 3 System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig instance void CopyTo(int32  
6 sourceIndex, class System.Char[] destination, int32  
7 destinationIndex, int32 count)
```

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

11 Summary

12 Copies a specified number of characters from a specified position in
13 the current **System.String** instance to a specified position in a
14 specified array of Unicode characters.

15 Parameters

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

18 Exceptions

Exception	Condition
System.ArgumentNullException	<i>destination</i> is a null reference.
System.ArgumentOutOfRangeException	<i>sourceIndex</i> , <i>destinationIndex</i> , or <i>count</i> is negative -or-
	<i>count</i> is greater than the length of the substring from <i>startIndex</i> 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
<i>value</i>	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	<i>value</i> is a null reference.

Example

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

```
[C#]
using System;
```

```
1 public class StringEndsWithExample {
2     public static void Main() {
3         string str = "One string to compare";
4         Console.WriteLine("The given string is '{0}'", str);
5         Console.Write("The given string ends with 'compare'? ");
6         Console.WriteLine(str.EndsWith("compare"));
7         Console.Write("The given string ends with 'Compare'? ");
8         Console.WriteLine(str.EndsWith("Compare"));
9     }
10 }
11
```

12 The output is

13
14 The given string is 'One string to compare'

15
16
17 The given string ends with 'compare'? True

18
19
20 The given string ends with 'Compare'? False

21

22

1 String.Equals(System.Object) Method

```
2 [ILASM]  
3 .method public hidebysig virtual bool Equals(object obj)  
4 [C#]  
5 public override bool Equals(object obj)
```

6 Summary

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

9 Parameters

10
11

Parameter	Description
<i>obj</i>	A System.Object .

12
13
14

13 Return Value

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

17 Description

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

20
21

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

22 Exceptions

23
24

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

25
26
27

26 Example

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

30
31

```
31 [C#]  
32 using System;  
33 public class StringEqualsExample {  
34     public static void Main() {
```

```
1     string str = "A string";
2     Console.WriteLine("The given string is '{0}'", str);
3     Console.Write("The given string is equal to 'A string'?
4 ");
5     Console.WriteLine(str.Equals("A string"));
6     Console.Write("The given string is equal to 'A String'?
7 ");
8     Console.WriteLine(str.Equals("A String"));
9     }
10    }
11
```

12 The output is

```
13
14     The given string is 'A string'
15
16
17     The given string is equal to 'A string'? True
18
19
20     The given string is equal to 'A String'? False
21
```

22

1 String.Equals(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static bool Equals(string a,  
5 string b)  
  
6 [C#]  
7 public static bool Equals(string a, string b)
```

8 Summary

9 Determines whether two specified **System.String** objects have the
10 same value.

11 Parameters

12
13

Parameter	Description
<i>a</i>	A System.String or a null reference.
<i>b</i>	A System.String or a null reference.

14
15
16

15 Return Value

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

18 Description

19 The comparison is case-sensitive.

20 Example

21

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

24
25

```
[C#]  
  
26 using System;  
27 public class StringEqualsExample {  
28     public static void Main() {  
29         string strA = "A string";  
30         string strB = "a string";  
31         string strC = "a string";  
32         Console.Write("The string '{0}' is equal to the string  
33 '{1}'? ", strA, strB);  
34         Console.WriteLine(String.Equals(strA, strB));  
35         Console.Write("The string '{0}' is equal to the string  
36 '{1}'? ", strC, strB);
```

```
1     Console.WriteLine(String.Equals(strC, strB));
2     }
3     }
4
```

5 The output is

```
6
7     The string 'A string' is equal to the string 'a string'?
8     False
9
```

```
10
11    The string 'a string' is equal to the string 'a string'?
12    True
13
```

14

1 String.Format(System.String, 2 System.Object) Method

```
3 [ILASM]  
4 .method public hidebysig static string Format(string  
5 format, object arg0)  
  
6 [C#]  
7 public static string Format(string format, object arg0)
```

8 Summary

9 Replaces the format specification in a provided **System.String** with a
10 specified textual equivalent of the value of a specified **System.Object**
11 instance.

12 Parameters

13
14

Parameter	Description
<i>format</i>	A System.String containing zero or more format specifications.
<i>arg0</i>	A System.Object to be formatted.

15
16
17

16 Return Value

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

20 Description

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

25 Exceptions

26
27

Exception	Condition
System.ArgumentNullException	<i>format</i> or <i>arg0</i> is a null reference.
System.FormatException	The format specification in <i>format</i> 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).

1
2
3

Example

4 The following example demonstrates the **System.String.Format**
5 method.

6
7

[C#]

8
9
10
11
12
13
14
15
16
17

```
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);
    }
}
```

18

The output is

19
20
21
22

```
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
<i>format</i>	A System.String containing zero or more format specifications.
<i>arg0</i>	A System.Object to be formatted. Can be a null reference.
<i>arg1</i>	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	<i>format</i> is a null reference.
System.FormatException	<i>format</i> 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).

1
2
3

Example

4
5
6
7

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

[C#]

8
9
10
11
12
13
14
15
16
17
18
19
20

```
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);
    }
}
```

21

The output is

22
23
24
25
26
27
28
29

```
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.
```

1 String.Format(System.String, 2 System.Object, System.Object, 3 System.Object) Method

```
4 [ILASM]  
5 .method public hidebysig static string Format(string  
6 format, object arg0, object arg1, object arg2)  
  
7 [C#]  
8 public static string Format(string format, object arg0,  
9 object arg1, object arg2)
```

10 Summary

11 Replaces the format specification in a specified **System.String** with
12 the textual equivalent of the value of three specified **System.Object**
13 instances.

14 Parameters

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

17 18 Return Value

19
20 A **System.String** containing a copy of *format* in which the first,
21 second, and third format specifications have been replaced by the
22 **System.String** equivalent of *arg0*, *arg1*, and *arg2*.

23 Description

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

26
27 [Note: This version of **System.String.Format** is equivalent to
28 **String.Format**(**null**, *format*, **new Object[]** {*arg0*, *arg1*, *arg2*}). For
29 more information on the format specification see the **System.String**
30 class overview.]

1 **Exceptions**

2
3

Exception	Condition
System.ArgumentNullException	<i>format</i> is a null reference.
System.FormatException	<i>format</i> 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).

4
5 **Example**
6

7 The following example demonstrates the **System.String.Format**
8 method.

9
10

[C#]

11
12
13
14
15
16
17
18
19
20
21
22

```
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);  
    }  
}
```

23

The output is

24
25
26
27
28
29

```
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.
```

1 String.Format(System.String, 2 System.Object[]) Method

```
3 [ILASM]  
4 .method public hidebysig static string Format(string  
5 format, class System.Object[] args)  
  
6 [C#]  
7 public static string Format(string format, params object[]  
8 args)
```

9 Summary

10 Replaces the format specification in a specified **System.String** with
11 the textual equivalent of the value of a corresponding **System.Object**
12 instance in a specified array.

13 Parameters

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

16 Return Value

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

22 Description

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

25
26 [Note: This version of **System.String.Format** is equivalent to
27 **System.String.Format**(null, *format*, *args*). For more information on
28 the format specification see the **System.String** class overview.]

29 Exceptions

Exception	Condition
System.ArgumentNullException	<i>format</i> or <i>args</i> is a null reference.
System.FormatException	<i>format</i> 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.

1
2
3

Example

4
5
6
7

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

[C#]

8
9
10
11
12
13
14
15
16
17
18

```
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);
    }
}
```

19

The output is

20
21
22
23
24

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

1 String.Format(System.IFormatProvider, 2 System.String, System.Object[]) Method

```
3 [ILASM]  
4 .method public hidebysig static string Format(class  
5 System.IFormatProvider provider, string format, class  
6 System.Object[] args)  
  
7 [C#]  
8 public static string Format(IFormatProvider provider,  
9 string format, params object[] args)
```

10 Summary

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

14 Parameters

15
16

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

17

18 Return Value

19

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

23 Description

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

26

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

1 **Exceptions**
2
3

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

4
5
6

1 String.GetEnumerator() Method

```
2 [ILASM]  
3 .method public hidebysig instance class  
4 System.CharEnumerator GetEnumerator()  
  
5 [C#]  
6 public CharEnumerator GetEnumerator()
```

7 Summary

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

10 Return Value

11

12 A **System.CharEnumerator** object.

13 Description

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

17

1 String.GetHashCode() Method

```
2 [ILASM]  
3 .method public hidebysig virtual int32 GetHashCode()  
4 [C#]  
5 public override int GetHashCode()
```

6 Summary

7 Generates a hash code for the current instance.

8 Return Value

9

10 A **System.Int32** containing the hash code for this instance.

11 Description

12 The algorithm used to generate the hash code is unspecified.

13

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

15

1 String.IndexOf(System.Char) Method

```
2 [ILASM]  
3 .method public hidebysig instance int32 IndexOf(valuetype  
4 System.Char value)  
5 [C#]  
6 public int IndexOf(char value)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>value</i>	A Unicode character.

13

14 Return Value

15

16 A **System.Int32** containing a positive value equal to the index of the
17 first occurrence of *value* character in the current instance; otherwise, -
18 1 if *value* was not found.

19 Description

20 This method is case-sensitive.

21

1 String.IndexOf(System.Char, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOf(valuetype  
5 System.Char value, int32 startIndex)  
  
6 [C#]  
7 public int IndexOf(char value, int startIndex)
```

8 Summary

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

12 Parameters

13
14

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

15
16
17

16 Return Value

18 A **System.Int32** containing a positive value equal to the index of the
19 first occurrence of *value* in the current instance; otherwise, -1 if *value*
20 was not found.

21 Description

22 This method is case-sensitive.

23 Exceptions

24
25

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> is less than zero or greater than the length of the current instance.

26
27
28

27 Example

29 The following example demonstrates the **System.String.IndexOf**
30 method.

```
1
2     [C#]

3     using System;
4     public class StringIndexOf {
5         public static void Main() {
6             String str = "This is the string";
7             Console.WriteLine("Searching for the index of 'h' starting
8 from index 0 yields {0}.", str.IndexOf('h', 0));
9             Console.WriteLine("Searching for the index of 'h' starting
10 from index 10 yields {0}.", str.IndexOf('h', 10));
11         }
12     }
```

13 The output is

```
14
15     Searching for the index of 'h' starting from index 0 yields
16     1.
17
18
19     Searching for the index of 'h' starting from index 10
20     yields -1.
21
```

22

1 String.IndexOf(System.Char, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOf(valuetype  
5 System.Char value, int32 startIndex, int32 count)  
  
6 [C#]  
7 public int IndexOf(char value, int startIndex, int count)
```

8 Summary

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

12 Parameters

13
14

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

15
16
17

16 Return Value

18 A **System.Int32** containing a positive value equal to the index of the
19 first occurrence of *value* in the current instance; otherwise, -1 if *value*
20 was not found.

21 Description

22 The search begins at *startIndex* and continues until *startIndex* + *count*
23 - 1 is reached. The character at *startIndex* + *count* is not included in
24 the search.

25
26 This method is case-sensitive.

27 Exceptions

28
29

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>count</i> is negative

1
2
3

-or-

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

1 String.IndexOf(System.String) Method

```
2 [ILASM]
3 .method public hidebysig instance int32 IndexOf(string
4 value)
5
6 [C#]
7 public int IndexOf(string value)
```

7 Summary

8 Returns the index of the first occurrence of a specified **System.String**
9 in the current instance.

10 Parameters

Parameter	Description
<i>value</i>	The System.String to seek.

14 Return Value

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

Return Value	Description
A positive number equal to the index of the start of the first substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
0	<i>value</i> is equal to System.String.Empty .
-1	<i>value</i> was not found.

19 Description

20 The search begins at the first character of the current instance. The
21 search is case-sensitive, culture-sensitive, and the culture of the
22 current thread is used.

23 Exceptions

Exception	Condition
-----------	-----------

1
2
3

Example

4 The following example demonstrates the **System.String.IndexOf**
5 method.

6
7

[C#]

```
8       using System;
9       public class StringIndexOf {
10        public static void Main() {
11         String str = "This is the string";
12         Console.WriteLine("Searching for the index of \"is\"
13         yields {0,2}.", str.IndexOf("is"));
14         Console.WriteLine("Searching for the index of \"Is\"
15         yields {0,2}.", str.IndexOf("Is"));
16         Console.WriteLine("Searching for the index of \"\" yields
17         {0,2}.", str.IndexOf(""));
18         }
19       }
```

20 The output is

21
22
23
24
25
26
27
28
29

Searching for the index of "is" yields 2.

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

Searching for the index of "" yields 0.

30

1 String.IndexOf(System.String, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOf(string  
5 value, int32 startIndex)  
  
6 [C#]  
7 public int IndexOf(string value, int startIndex)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>value</i>	The System.String to seek.
<i>startIndex</i>	A System.Int32 containing the index of the current instance from which to start searching.

14
15
16

15 Return Value

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

Return Value	Description
A positive number equal to the index of the start of the first substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
<i>startIndex</i>	<i>value</i> is Empty .
-1	<i>value</i> was not found.

19

20 Description

21 This method is case-sensitive.

22 Exceptions

23
24

Exception	Condition
-----------	-----------

- 1
- 2
- 3

System.ArgumentNullException	<i>value</i> is a null reference.
System.ArgumentOutOfRangeException	<i>startIndex</i> is greater than the length of the current instance.

1 String.IndexOf(System.String, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOf(string  
5 value, int32 startIndex, int32 count)  
  
6 [C#]  
7 public int IndexOf(string value, int startIndex, int count)
```

8 Summary

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

12 Parameters

13
14

Parameter	Description
<i>value</i>	The System.String to seek.
<i>startIndex</i>	A System.Int32 containing the index of the current instance from which to start searching.
<i>count</i>	A System.Int32 containing the range of the current instance at which to end searching.

15
16
17

16 Return Value

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

Return Value	Description
A positive number equal to the index of the start of the first substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
<i>startIndex</i>	<i>value</i> is Empty .
-1	<i>value</i> was not found.

20

21 Description

22 The search begins at *startIndex* and continues until *startIndex* + *count*
23 - 1 is reached. The character at *startIndex* + *count* is not included in

1 the search.
2
3 This method is case-sensitive.

4 **Exceptions**
5
6

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

7
8
9

1 String.IndexOfAny(System.Char[])

2 Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOfAny(class  
5 System.Char[] anyOf)  
  
6 [C#]  
7 public int IndexOfAny(char[] anyOf)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>anyOf</i>	An array of Unicode characters.

14
15
16

15 Return Value

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

19 Description

20 This method is case-sensitive.

21 Exceptions

22
23

Exception	Condition
System.ArgumentNullException	<i>anyOf</i> is a null reference.

24
25
26

1 String.IndexOfAny(System.Char[], 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOfAny(class  
5 System.Char[] anyOf, int32 startIndex)  
  
6 [C#]  
7 public int IndexOfAny(char[] anyOf, int startIndex)
```

8 Summary

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

12 Parameters

13
14

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

15
16
17

16 Return Value

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

21 Description

22 This method is case-sensitive.

23 Exceptions

24
25

Exception	Condition
System.ArgumentNullException	<i>anyOf</i> is a null reference.
System.ArgumentOutOfRangeException	<i>startIndex</i> is greater than the length of the current instance

26
27
28

1 String.IndexOfAny(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 IndexOfAny(class  
5 System.Char[] anyOf, int32 startIndex, int32 count)  
  
6 [C#]  
7 public int IndexOfAny(char[] anyOf, int startIndex, int  
8 count)
```

9 Summary

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

13 Parameters

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

16 Return Value

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

22 Description

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

25
26 This method is case-sensitive.

27 Exceptions

Exception	Condition
System.ArgumentNullException	<i>anyOf</i> is a null reference.

1
2
3

System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>count</i> is negative. -or- <i>startIndex</i> + <i>count</i> is greater than the length of the current instance.
---	--

1 String.Insert(System.Int32, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig instance string Insert(int32  
5 startIndex, string value)  
  
6 [C#]  
7 public string Insert(int startIndex, string value)
```

8 Summary

9 Returns a **System.String** equivalent to the current instance with a
10 specified **System.String** inserted at the specified position.

11 Parameters

12
13

Parameter	Description
<i>startIndex</i>	A System.Int32 containing the index of the insertion.
<i>value</i>	The System.String to insert.

14
15
16

15 Return Value

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

19 Description

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

24 Exceptions

25
26

Exception	Condition
System.ArgumentNullException	<i>value</i> is a null reference.
System.ArgumentOutOfRangeException	<i>startIndex</i> is greater than the length of the current instance.

27
28
29

1 String.Intern(System.String) Method

```
2 [ILASM]  
3 .method public hidebysig static string Intern(string str)  
4 [C#]  
5 public static string Intern(string str)
```

6 Summary

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

8 Parameters

9
10

Parameter	Description
<i>str</i>	A System.String .

11
12
13

Return Value

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

15 Description

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

19
20 The intern pool conserves string storage. If a literal string constant is
21 assigned to several variables, each variable is set to reference the
22 same constant in the intern pool instead of referencing several
23 different instances of **System.String** that have identical values.

24
25 This method looks up a specified string in the intern pool. If the string
26 exists, a reference to it is returned. If it does not exist, an instance
27 equal to the specified string is added to the intern pool and a reference
28 that instance is returned.

29 Exceptions

30
31

Exception	Condition
System.ArgumentNullException	<i>str</i> is a null reference.

32
33
34

Example

1 The following example demonstrates the **System.String.Intern**
2 method.

```
3  
4 [C#]  
  
5 using System;  
6 using System.Text;  
7 public class StringExample {  
8     public static void Main() {  
9  
10         String s1 = "MyTest";  
11         String s2 = new  
12         StringBuilder().Append("My").Append("Test").ToString();  
13         String s3 = String.Intern(s2);  
14  
15         Console.WriteLine(Object.ReferenceEquals(s1, s2));  
16         //different  
17         Console.WriteLine(Object.ReferenceEquals(s1, s3));  
18         //the same  
19     }  
20 }
```

21 The output is

22
23 False

24
25
26 True

27

28

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
<i>str</i>	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	<i>str</i> is a null reference.

Example

1 The following example demonstrates the **System.String.IsInterned**
2 method.

```
3  
4 [C#]  
  
5 using System;  
6 using System.Text;  
7  
8 public class StringExample {  
9     public static void Main() {  
10  
11         String s1 = new  
12         StringBuilder().Append("My").Append("Test").ToString();  
13  
14         Console.WriteLine(String.IsInterned(s1) != null);  
15     }  
16 }
```

17 The output is

```
18  
19 True
```

20

1 String.Join(System.String, 2 System.String[]) Method

```
3 [ILASM]  
4 .method public hidebysig static string Join(string  
5 separator, class System.String[] value)  
  
6 [C#]  
7 public static string Join(string separator, string[] value)
```

8 Summary

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

12 Parameters

Parameter	Description
<i>separator</i>	A System.String .
<i>value</i>	A System.String array.

15 Return Value

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

20 Exceptions

Exception	Condition
System.ArgumentNullException	<i>value</i> is a null reference.

23 Example

26 The following example demonstrates the **System.String.Join** method.

```
27 [C#]  
28  
29 using System;  
30 public class StringJoin {  
31     public static void Main() {  
32         String[] strAry = { "Red", "Green", "Blue" };  
33         Console.WriteLine(String.Join(":: ", strAry));  
34     }  
}
```

1 }

2 The output is

3

4 Red:: Green:: Blue

5

1 String.Join(System.String, 2 System.String[], System.Int32, 3 System.Int32) Method

```
4 [ILASM]  
5 .method public hidebysig static string Join(string  
6 separator, class System.String[] value, int32 startIndex,  
7 int32 count)
```

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

11 Summary

12 Concatenates a specified separator **System.String** between the
13 elements of a specified **System.String** array, yielding a single
14 concatenated string.

15 Parameters

Parameter	Description
<i>separator</i>	A System.String .
<i>value</i>	A System.String array.
<i>startIndex</i>	A System.Int32 containing the first array element in <i>value</i> to join.
<i>count</i>	A System.Int32 containing the number of elements in <i>value</i> to join.

18 Return Value

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

22 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> plus <i>count</i> is greater than the number of elements in <i>value</i> .

27 Example

1 The following example demonstrates the **System.String.Join** method.

2
3

[C#]

```
4 using System;
5 public class StringJoin {
6     public static void Main() {
7         String[] strAry = { "Red", "Green", "Blue" };
8         Console.WriteLine(String.Join(":: ", strAry, 1, 2));
9     }
10 }
```

11 The output is

12
13

Green:: Blue

14

1 String.LastIndexOf(System.Char) Method

```
2 [ILASM]  
3 .method public hidebysig instance int32  
4 LastIndexOf(valuetype System.Char value)  
  
5 [C#]  
6 public int LastIndexOf(char value)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>value</i>	The Unicode character to locate.

13

14 Return Value

15

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

18 Description

19 This method is case-sensitive.
20

1 String.LastIndexOf(System.Char, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32  
5 LastIndexOf(valuetype System.Char value, int32 startIndex)  
6  
7 [C#]  
8 public int LastIndexOf(char value, int startIndex)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

15 Return Value

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

19 Description

20 This method searches for the last occurrence of the specified character
21 between the start of the string and the indicated index.
22

23 This method is case-sensitive.

24 Exceptions

25
26

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

27
28
29

28 Example

1 The following example demonstrates the
2 **System.String.LastIndexOf** method.

```
3  
4 [C#]  
  
5 using System;  
6 public class StringLastIndexOfTest {  
7     public static void Main() {  
8         String str = "aa bb cc dd";  
9  
10        Console.WriteLine(str.LastIndexOf('d', 8));  
11        Console.WriteLine(str.LastIndexOf('b', 8));  
12    }  
13 }
```

14 The output is

```
15  
16 -1  
17  
18  
19 4  
20
```

21

1 String.LastIndexOf(System.Char, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32  
5 LastIndexOf(valuetype System.Char value, int32 startIndex,  
6 int32 count)  
  
7 [C#]  
8 public int LastIndexOf(char value, int startIndex, int  
9 count)
```

10 Summary

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

13 Parameters

14
15

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

16
17
18

Return Value

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

22 Description

23 This method is case-sensitive.

24 Exceptions

25
26

Exception	Condition
System.ArgumentNullException	<i>value</i> is a null reference.
System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>count</i> is less than zero.
	-or-

1
2
3

	<i>startIndex - count</i> is less than -1.
--	--

1 String.LastIndexOf(System.String)

2 Method

```
3 [ILASM]
4 .method public hidebysig instance int32 LastIndexOf(string
5 value)
6
7 [C#]
8 public int LastIndexOf(string value)
```

8 Summary

9 Returns the index of the last occurrence of a specified **System.String**
10 within the current instance.

11 Parameters

12
13

Parameter	Description
<i>value</i>	A System.String .

14
15
16

15 Return Value

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

Return Value	Description
A positive number equal to the index of the start of the last substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
0	<i>value</i> is Empty .
-1	<i>value</i> was not found.

19

20 Description

21 The search is case-sensitive.

22 Exceptions

23
24

Exception	Condition
System.ArgumentNullException	<i>value</i> is a null reference.

1
2
3

1 String.LastIndexOf(System.String, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 LastIndexOf(string  
5 value, int32 startIndex)  
  
6 [C#]  
7 public int LastIndexOf(string value, int startIndex)
```

8 Summary

9 Returns the index of the last occurrence of a specified **System.String**
10 within the current instance.

11 Parameters

12
13

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

14
15
16

15 Return Value

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

Return Value	Description
A positive number equal to the index of the start of the last substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
<i>startIndex</i>	<i>value</i> is Empty.
-1	<i>value</i> was not found.

19

20 Description

21 This method searches for the last occurrence of the specified
22 **System.String** between the start of the string and the indicated
23 index.

24
25

This method is case-sensitive.

1 **Exceptions**
2
3

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

4
5
6

1 String.LastIndexOf(System.String, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32 LastIndexOf(string  
5 value, int32 startIndex, int32 count)  
  
6 [C#]  
7 public int LastIndexOf(string value, int startIndex, int  
8 count)
```

9 Summary

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

12 Parameters

13
14

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

15
16
17

Return Value

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

Return Value	Description
A positive number equal to the index of the start of the last substring in the current instance that is equal to <i>value</i> .	<i>value</i> was found.
<i>startIndex</i>	<i>value</i> is Empty .
-1	<i>value</i> was not found.

20

21 Description

22 The search begins at *startIndex* and continues until *startIndex* - *count*
23 + 1.

1
2
3
4
5

This method is case-sensitive.

Exceptions

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

6
7
8

1 String.LastIndexOfAny(System.Char[])

2 Method

```
3 [ILASM]  
4 .method public hidebysig instance int32  
5 LastIndexOfAny(class System.Char[] anyOf)  
6  
7 [C#]  
8 public int LastIndexOfAny(char[] anyOf)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>anyOf</i>	An array of Unicode characters.

14
15
16

15 Return Value

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

19 Description

20 This method is case-sensitive.

21 Exceptions

22
23

Exception	Condition
System.ArgumentNullException	<i>anyOf</i> is a null reference.

24
25
26

1 String.LastIndexOfAny(System.Char[], 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32  
5 LastIndexOfAny(class System.Char[] anyOf, int32 startIndex)  
  
6 [C#]  
7 public int LastIndexOfAny(char[] anyOf, int startIndex)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

15 Return Value

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

19 Description

20 This method searches for the last occurrence of the specified
21 characters between the start of the string and the indicated index.
22

23 This method is case-sensitive.

24 Exceptions

25
26

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

1
2
3

1 String.LastIndexOfAny(System.Char[], 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance int32  
5 LastIndexOfAny(class System.Char[] anyOf, int32 startIndex,  
6 int32 count)  
  
7 [C#]  
8 public int LastIndexOfAny(char[] anyOf, int startIndex, int  
9 count)
```

10 Summary

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

13 Parameters

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

16 Return Value

17 A **System.Int32** containing the index of the last occurrence of any
18 element of *anyOf* if found between *startIndex* and (*startIndex* - *count*
19 + 1); otherwise, -1.
20
21

22 Description

23 The search begins at *startIndex* and continues until *startIndex* - *count*
24 + 1. The character at *startIndex* - *count* is not included in the search.
25

26 This method is case-sensitive.

27 Exceptions

Exception	Condition
System.ArgumentNullException	<i>anyOf</i> is a null reference.

1
2
3

System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>count</i> is less than zero. -or- <i>startIndex - count</i> is smaller than -1.
---	---

1 String.op_Equality(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static specialname bool  
5 op_Equality(string a, string b)  
  
6 [C#]  
7 public static bool operator ==(String a, String b)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

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

18

1 String.op_Inequality(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig static specialname bool  
5 op_Inequality(string a, string b)  
  
6 [C#]  
7 public static bool operator !=(String a, String b)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

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

19

1 String.PadLeft(System.Int32) Method

```
2 [ILASM]  
3 .method public hidebysig instance string PadLeft(int32  
4 totalWidth)  
  
5 [C#]  
6 public string PadLeft(int totalWidth)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>totalWidth</i>	A System.Int32 containing the number of characters in the resulting string.

13
14
15

14 Return Value

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

21 Description

22 [Note: A space in Unicode format is defined as the hexadecimal value
23 0x20.]

24 Exceptions

25
26

Exception	Condition
System.ArgumentException	<i>totalWidth</i> is less than zero.

27
28
29

1 String.PadLeft(System.Int32, 2 System.Char) Method

```
3 [ILASM]  
4 .method public hidebysig instance string PadLeft(int32  
5 totalWidth, valuetype System.Char paddingChar)  
  
6 [C#]  
7 public string PadLeft(int totalWidth, char paddingChar)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

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

22 Exceptions

23
24

Exception	Condition
System.ArgumentException	<i>totalWidth</i> is less than zero.

25
26
27

1 String.PadRight(System.Int32) Method

```
2 [ILASM]  
3 .method public hidebysig instance string PadRight(int32  
4 totalWidth)  
5  
6 [C#]  
7 public string PadRight(int totalWidth)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>totalWidth</i>	A System.Int32 containing the number of characters in the resulting string.

14 Return Value

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

21 Exceptions

Exception	Condition
System.ArgumentException	<i>totalWidth</i> is less than zero.

1 String.PadRight(System.Int32, 2 System.Char) Method

```
3 [ILASM]  
4 .method public hidebysig instance string PadRight(int32  
5 totalWidth, valuetype System.Char paddingChar)  
  
6 [C#]  
7 public string PadRight(int totalWidth, char paddingChar)
```

8 Summary

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

12 Parameters

13
14

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

15
16
17

16 Return Value

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

23 Exceptions

24
25

Exception	Condition
System.ArgumentException	<i>totalWidth</i> is less than zero.

26
27
28

1 String.Remove(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance string Remove(int32  
5 startIndex, int32 count)  
  
6 [C#]  
7 public string Remove(int startIndex, int count)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

Return Value

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

19 Exceptions

20
21

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

22
23
24

1 String.Replace(System.Char, 2 System.Char) Method

```
3 [ILASM]  
4 .method public hidebysig instance string Replace(valuetype  
5 System.Char oldChar, valuetype System.Char newChar)  
  
6 [C#]  
7 public string Replace(char oldChar, char newChar)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>oldChar</i>	The Unicode character to be replaced.
<i>newChar</i>	The Unicode character to replace all occurrences of <i>oldChar</i> .

14
15
16

Return Value

17 A **System.String** equivalent to the current instance with all
18 occurrences of *oldChar* replaced with *NewChar*.

19

1 String.Replace(System.String, 2 System.String) Method

```
3 [ILASM]  
4 .method public hidebysig instance string Replace(string  
5 oldValue, string newValue)  
  
6 [C#]  
7 public string Replace(string oldValue, string newValue)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>oldValue</i>	A System.String containing the string value to be replaced.
<i>newValue</i>	A System.String containing the string value to replace all occurrences of <i>oldValue</i> . Can be a null reference.

14
15
16

Return Value

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

20

1 String.Split(System.Char[]) Method

```
2 [ILASM]  
3 .method public hidebysig instance class System.String[]  
4 Split(class System.Char[] separator)  
  
5 [C#]  
6 public string[] Split(params char[] separator)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>separator</i>	A System.Char array of delimiters. Can be a null reference.

13
14
15

14 Return Value

16 A **System.String** array containing the results of the split operation as
17 follows:

Return Value	Description
A single-element array containing the current instance.	None of the elements of <i>separator</i> 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 <i>separator</i> .	At least one element of <i>separator</i> 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 <i>separator</i> is a null reference or an empty array.

18

19 Description

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

23
24

Delimiter characters are not included in the substrings.

1 String.Split(System.Char[], System.Int32) 2 Method

```
3 [ILASM]  
4 .method public hidebysig instance class System.String[]  
5 Split(class System.Char[] separator, int32 count)  
6  
7 [C#]  
8 public string[] Split(char[] separator, int count)
```

8 Summary

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

11 Parameters

12
13

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

14
15
16

15 Return Value

17 A **System.String** array containing the results of the split operation as
18 follows:

Return Value	Description
A single-element array containing the current instance.	None of the elements of <i>separator</i> 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 <i>separator</i>	At least one element of <i>separator</i> 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 <i>separator</i> is a null reference or an empty array.

19

20 Description

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

4
5 Delimiter characters are not included in the substrings.

6
7 If there are more substrings in the current instance than the maximum
8 specified number, the first *count* - 1 elements of the array contain the
9 first *count* - 1 substrings. The remaining characters in the current
10 instance are returned in the last element of the array.

11 **Exceptions**

12
13

Exception	Condition
System.ArgumentOutOfRangeException	<i>count</i> is negative.

14
15
16

1 String.StartsWith(System.String) Method

```
2 [ILASM]  
3 .method public hidebysig instance bool StartsWith(string  
4 value)  
5  
6 [C#]  
7 public bool StartsWith(string value)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>value</i>	A System.String .

14 Return Value

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

19 Description

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

25
26 The comparison is case-sensitive.

27 Exceptions

Exception	Condition
System.ArgumentNullException	<i>value</i> is a null reference.

1 String.Substring(System.Int32) Method

```
2 [ILASM]
3 .method public hidebysig instance string Substring(int32
4 startIndex)
5
6 [C#]
7 public string Substring(int startIndex)
```

7 Summary

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

10 Parameters

Parameter	Description
<i>startIndex</i>	A System.Int32 containing the index of the start of the substring in the current instance.

14 Return Value

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

19 Exceptions

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> is less than zero or greater than or equal to the length of the current instance.

1 String.Substring(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance string Substring(int32  
5 startIndex, int32 length)  
  
6 [C#]  
7 public string Substring(int startIndex, int length)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>startIndex</i>	A System.Int32 containing the index of the start of the substring in the current instance.
<i>length</i>	A System.Int32 containing the number of characters in the substring.

14
15
16

Return Value

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

21 Exceptions

22
23

Exception	Condition
System.ArgumentOutOfRangeException	<i>length</i> is greater than the length of the current instance. -or- <i>startIndex</i> or <i>length</i> is less than zero.

24
25
26

1 String.System.Collections.IEnumerable.Ge 2 tEnumerator() Method

```
3 [ILASM]  
4 .method private final hidebysig virtual class  
5 System.Collections.IEnumerator  
6 System.Collections.IEnumerable.GetEnumerator()  
  
7 [C#]  
8 IEnumerator IEnumerable.GetEnumerator()
```

9 Summary

10 Implemented to support the **System.Collections.IEnumerable**
11 interface. [Note: For more information, see
12 **System.Collections.IEnumerable.GetEnumerator.**]

13

1 String.ToCharArray() Method

```
2 [ILASM]
3 .method public hidebysig instance class System.Char[]
4 ToCharArray()
5
6 [C#]
7 public char[] ToCharArray()
```

7 Summary

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

10 Return Value

11

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

15

1 String.ToArray(System.Int32, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig instance class System.Char[]  
5 ToCharArray(int32 startIndex, int32 length)  
  
6 [C#]  
7 public char[] ToCharArray(int startIndex, int length)
```

8 Summary

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

11 Parameters

12
13

Parameter	Description
<i>startIndex</i>	A System.Int32 containing the index of the start of a substring in the current instance.
<i>length</i>	A System.Int32 containing the length of the substring in the current instance.

14
15
16

15 Return Value

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

23 Exceptions

24
25

Exception	Condition
System.ArgumentOutOfRangeException	<i>startIndex</i> or <i>length</i> is less than zero. -or- <i>startIndex</i> plus <i>length</i> is greater than the length of the current instance.

26
27
28

1 String.ToLower() Method

```
2 [ILASM]  
3 .method public hidebysig instance string ToLower()  
4 [C#]  
5 public string ToLower()
```

6 Summary

7 Returns a copy of this **System.String** in lower case.

8 Return Value

9

10 A **System.String** in lower case..

11 Description

12 This method takes into account the culture of the current thread.

13

1 String.ToString(System.IFormatProvider)

2 Method

```
3 [ILASM]  
4 .method public final hidebysig virtual string  
5 ToString(class System.IFormatProvider provider)  
  
6 [C#]  
7 public string ToString(IFormatProvider provider)
```

8 Summary

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

10 Parameters

11
12

Parameter	Description
<i>provider</i>	(Reserved) A System.IFormatProvider interface implementation which supplies culture-specific formatting information.

13
14
15

14 Return Value

16 This **String**.

17 Description

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

20

1 String.ToString() Method

```
2 [ILASM]  
3 .method public hidebysig virtual string ToString()  
4 [C#]  
5 public override string ToString()
```

6 Summary

7 Returns a **System.String** representation of the value of the current
8 instance.

9 Return Value 10

11 The current **System.String**.

12 Description

13 [*Note:* This method overrides **System.Object.ToString.**]
14

1 String.ToUpper() Method

```
2 [ILASM]  
3 .method public hidebysig instance string ToUpper()  
4 [C#]  
5 public string ToUpper()
```

6 Summary

7 Returns a copy of the current instance with all elements converted to
8 upper case, using default properties.

9 Return Value

10

11 A new **System.String** in upper case.

12

1 String.Trim(System.Char[]) Method

```
2 [ILASM]  
3 .method public hidebysig instance string Trim(class  
4 System.Char[] trimChars)  
  
5 [C#]  
6 public string Trim(params char[] trimChars)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>trimChars</i>	An array of Unicode characters. Can be a null reference.

13
14
15

Return Value

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

20

1 String.Trim() Method

```
2 [ILASM]  
3 .method public hidebysig instance string Trim()  
4 [C#]  
5 public string Trim()
```

6 Summary

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

9 Return Value

10

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

13

1 String.TrimEnd(System.Char[]) Method

```
2 [ILASM]  
3 .method public hidebysig instance string TrimEnd(class  
4 System.Char[] trimChars)  
  
5 [C#]  
6 public string TrimEnd(params char[] trimChars)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>trimChars</i>	An array of Unicode characters. Can be a null reference.

13
14
15

Return Value

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

19

1 String.TrimStart(System.Char[]) Method

```
2 [ILASM]  
3 .method public hidebysig instance string TrimStart(class  
4 System.Char[] trimChars)  
  
5 [C#]  
6 public string TrimStart(params char[] trimChars)
```

7 Summary

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

10 Parameters

11
12

Parameter	Description
<i>trimChars</i>	An array of Unicode characters or a null reference.

13
14
15

Return Value

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

19

1 String.Chars Property

```
2 [ILASM]
3 .property valuetype System.Char Chars[int32 index] { public
4 hidebysig specialname instance valuetype System.Char
5 get_Chars(int32 index) }
6
7 [C#]
8 public char this[int index] { get; }
```

8 Summary

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

10 Property Value

11

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

13 Description

14 This property is read-only.

15

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

20 Exceptions

21

22

Exception	Condition
System.IndexOutOfRangeException	<i>index</i> is greater than or equal to the length of the current instance or less than zero.

23

24

25

1 String.Length Property

```
2 [ILASM]
3 .property int32 Length { public hidebysig specialname
4 instance int32 get_Length() }
5
6 [C#]
7 public int Length { get; }
```

7 Summary

8 Gets the number of characters in the current instance.

9 Property Value

10

11 A **System.Int32** containing the number of characters in the current
12 instance.

13 Description

14 This property is read-only.

15 Example

16

17 The following example demonstrates the **System.String.Length**
18 property.

19

20

```
21 using System;
22 public class StringLengthExample {
23     public static void Main() {
24         string str = "STRING";
25         Console.WriteLine("The length of string {0} is {1}", str,
26 str.Length);
27     }
28 }
```

29 The output is

30

31 The length of string STRING is 6

32