

1 System.Math Class

2

3

4

[ILASM]

5

```
.class public sealed Math extends System.Object
```

6

[C#]

7

```
public sealed class Math
```

8

Assembly Info:

9

- Name: mscorlib

10

- Public Key: [00 00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]

11

- Version: 1.0.x.x

12

- Attributes:

13

- CLSCompliantAttribute(true)

14

Summary

15

16

Provides constants and static methods for trigonometric, logarithmic,

17

and other common mathematical functions.

18

Inherits From: System.Object

19

20

Library: ExtendedNumerics

21

22

Thread Safety: All public static members of this type are safe for multithreaded

23

operations. No instance members are guaranteed to be thread safe.

24

25

26

1 Math.E Field

```
2 [ILASM]  
3 .field public static literal float64 E = 2.71828182845905  
4 [C#]  
5 public const double E = 2.71828182845905
```

6 Summary

7 A constant, **e**, which specifies the natural logarithmic base rounded to
8 double precision.

9 Description

10 The value of this constant is 2.7182818284590452354 converted to
11 **System.Double**.

12

1 Math.PI Field

```
2 [ILASM]  
3 .field public static literal float64 PI = 3.14159265358979  
4 [C#]  
5 public const double PI = 3.14159265358979
```

6 Summary

7 A constant, π , which specifies the ratio of the circumference of a circle
8 to its diameter rounded to double precision.

9 Description

10 The value of this constant is 3.14159265358979323846 converted to
11 **System.Double**.

12

1 Math.Abs(System.SByte) Method

```
2 [ILASM]  
3 .method public hidebysig static int8 Abs(int8 value)  
4 [C#]  
5 public static sbyte Abs(sbyte value)
```

6 Summary

7 Returns the absolute value of the specified **System.SByte**.

8 Type Attributes:

- 9 • CLSCompliantAttribute(false)

10 Parameters

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

13 Return Value

14 A **System.SByte** containing the absolute value of *value*.

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative,
19 use **System.Math.Abs(System.Int16)**.

20 Exceptions

Exception	Condition
System.OverflowException	<i>value</i> equals System.SByte.MinValue .

23
24
25

1 Math.Abs(System.Int16) Method

```
2 [ILASM]  
3 .method public hidebysig static int16 Abs(int16 value)  
4 [C#]  
5 public static short Abs(short value)
```

6 Summary

7 Returns the absolute value of the specified **System.Int16**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int16 .

11
12
13

Return Value

14 A **System.Int16** containing the absolute value of *value*.

15 Exceptions

16
17

Exception	Condition
System.OverflowException	<i>value</i> equals System.Int16.MinValue .

18
19
20

1 Math.Abs(System.Int32) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Abs(int32 value)  
4 [C#]  
5 public static int Abs(int value)
```

6 Summary

7 Returns the absolute value of the specified **System.Int32**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int32 .

11
12
13

Return Value

14 A **System.Int32** containing the absolute value of *value*.

15 Exceptions

16
17

Exception	Condition
System.OverflowException	<i>value</i> equals System.Int32.MinValue .

18
19
20

1 Math.Abs(System.Int64) Method

```
2 [ILASM]  
3 .method public hidebysig static int64 Abs(int64 value)  
4 [C#]  
5 public static long Abs(long value)
```

6 Summary

7 Returns the absolute value of the specified **System.Int64**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int64 .

11
12
13

Return Value

14 A **System.Int64** containing the absolute value of *value*.

15 Exceptions

16
17

Exception	Condition
System.OverflowException	<i>value</i> equals System.Int64.MinValue .

18
19
20

1 Math.Abs(System.Single) Method

```
2 [ILASM]  
3 .method public hidebysig static float32 Abs(float32 value)  
4 [C#]  
5 public static float Abs(float value)
```

6 Summary

7 Returns the absolute value of the specified **System.Single**.

8 Parameters

9
10

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

11
12
13

Return Value

14 A **System.Single** containing the absolute value of *value*. If *value* is
15 equal to **System.Single.NegativeInfinity** or
16 **System.Single.PositiveInfinity**, returns
17 **System.Single.PositiveInfinity**. If *value* is equal to
18 **System.Single.NaN**, returns **System.Single.NaN**.

19

1 Math.Abs(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Abs(float64 value)  
4 [C#]  
5 public static double Abs(double value)
```

6 Summary

7 Returns the absolute value of the specified **System.Double**.

8 Parameters

9
10

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

11
12
13

Return Value

14 A **System.Double** containing the absolute value of *value*. If *value* is
15 equal to **System.Double.NegativeInfinity** or
16 **System.Double.PositiveInfinity**, returns
17 **System.Double.PositiveInfinity**. If *value* is equal to
18 **System.Double.NaN**, returns **System.Double.NaN**.

19

1 Math.Abs(System.Decimal) Method

```
2 [ILASM]  
3 .method public hidebysig static decimal Abs(decimal value)  
4 [C#]  
5 public static decimal Abs(decimal value)
```

6 Summary

7 Returns the absolute value of the specified **System.Decimal**.

8 Parameters

9
10

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

11

12 Return Value

13

14 A **System.Decimal** containing the absolute value of *value*.

15 Example

16

17 The following example demonstrates the
18 **System.Math.Abs(System.Decimal)** method.

19
20

```
[C#]  
21  
22 using System;  
23  
24 public class MathAbsExample  
25 {  
26     public static void Main()  
27     {  
28         Decimal d1 = Math.Abs((Decimal)0.00);  
29         Decimal d2 = Math.Abs((Decimal)(-1.23));  
30         Console.WriteLine("Math.Abs((Decimal)0.00) returns  
31 {0}",d1);  
32         Console.WriteLine("Math.Abs((Decimal)(-1.23)) returns  
33 {0}",d2);  
34     }  
35 }
```

36 The output is

37

```
1      Math.Abs((Decimal)0.00) returns 0
2
3
4      Math.Abs((Decimal)(-1.23)) returns 1.23
5
6
```

18

19

degrees.]

20

1 Math.Asin(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Asin(float64 d)  
4 [C#]  
5 public static double Asin(double d)
```

6 Summary

7 Returns the angle whose sine is the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>d</i>	A System.Double representing a sine, where $-1 \leq d \leq 1$.

11
12
13

Return Value

14 A **System.Double**
15 radians, for which *d* is the sine, such that - $d < -$
16 1, $d > 1$, or $d = \mathbf{System.Double.NaN}$, returns **System.Double.NaN**.

17 Description

18 [Note: A positive return value represents a counterclockwise angle
19 from the positive x-axis; a negative return value represents a
20 clockwise angle.
21
22
23 degrees.]

24

15
16
17
18
19

The following table specifies the return value if *d* is equal to **System.Double.NaN**, **System.Double.NegativeInfinity**, or **System.Double.PositiveInfinity**.

Return Value	Condition
System.Double.NaN	<i>d</i> is equal to System.Double.NaN .
- double precision (-1.5707963267949)	<i>d</i> is equal to System.Double.NegativeInfinity .
(1.5707963267949)	<i>d</i> is equal to System.Double.PositiveInfinity .

20

21 Description

22 [Note: A positive return value represents a counterclockwise angle
23 from the positive x-axis; a negative return value represents a
24 clockwise angle.

25
26
27

degrees.]

28

1 Math.Atan2(System.Double, 2 System.Double) Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Atan2(float64 y,  
5 float64 x)
```

```
6 [C#]  
7 public static double Atan2(double y, double x)
```

8 Summary

9 Returns the angle whose tangent is the quotient of two specified
10 **System.Double** values.

11 Parameters

Parameter	Description
y	A System.Double representing the y coordinate of a point.
x	A System.Double representing the x coordinate of a point.

14 Return Value

17 A **System.Double**
18 radians, such that - y/x , where (x, y) is a
19 point in the Cartesian plane.

20
21 If both x and y are any combination of
22 **System.Double.NegativeInfinity** and
23 **System.Double.PositiveInfinity**, **System.Double.NaN** is returned.

24
25 If either x or y is equal to **System.Double.NaN**,
26 **System.Double.NaN** is returned.

27
28 The following table specifies the return value if x or y is equal to
29 **System.Double.NegativeInfinity** or
30 **System.Double.PositiveInfinity**.

Condition	Return Value
y is equal to System.Double.PositiveInfinity or System.Double.NegativeInfinity , and x is equal to System.Double.PositiveInfinity or System.Double.NegativeInfinity .	System.Double.NaN .
v is equal to Svstem.Double.NeaativeInfnitiv . and	-System.Math.PI/2 .

9

10 • For (x, y) in quadrant 3, - -

11 • For (x, y) in quadrant 4, -

12]

13 **Example**

14

15 The following example demonstrates using the **System.Math.Atan2**
16 method.

```
1
2     [C#]
3
4     using System;
5
6     public class MathAtan2Example
7     {
8         public static void Main()
9         {
10
11             Double d1 = Math.Atan2(2,0);
12             Double d2 = Math.Atan2(0,0);
13             Console.WriteLine("Math.Atan2(2,0) returns {0}", d1);
14             Console.WriteLine("Math.Atan2(0,0) returns {0}", d2);
15
16         }
17
18     }
```

```
19     The output is
20
21     Math.Atan2(2,0) returns 1.5707963267949
22
23
24     Math.Atan2(0,0) returns 0
25
```

26

1 Math.BigMul(System.Int32, System.Int32)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int64 BigMul(int32 a,int32  
5 b)  
6 [C#]  
7 public static long BigMul(int a, int b)
```

8 Summary

9 Produces the full product of two 32-bit numbers.

10 Parameters

11
12

Parameter	Description
<i>a</i>	The first System.Int32 to multiply.
<i>b</i>	The second System.Int32 to multiply.

13
14
15

Return Value

16 A **System.Int64** containing the product of the specified numbers.
17

1 Math.Ceiling(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Ceiling(float64 a)  
4  
5 [C#]  
6 public static double Ceiling(double a)
```

6 Summary

7 Returns the smallest integer greater than or equal to the specified
8 **System.Double**.

9 Parameters

10
11

Parameter	Description
<i>a</i>	A System.Double .

12
13
14

13 Return Value

15 A **System.Double** containing the value of the smallest integer greater
16 than or equal to *a*. If *a* is equal to **System.Double.NaN**,
17 **System.Double.NegativeInfinity**, or
18 **System.Double.PositiveInfinity**, that value is returned.

19 Example

20

21 The following example demonstrates using the **System.Math.Ceiling**
22 method.

23
24

```
25 [C#]  
26 using System;  
27 public class MathCeilingExample  
28 {  
29  
30     public static void Main()  
31     {  
32  
33         Double d1 = Math.Ceiling(3.4);  
34         Double d2 = Math.Ceiling(-3.4);  
35         Console.WriteLine("Math.Ceiling(3.4) returns {0}",  
36 d1);  
37         Console.WriteLine("Math.Ceiling(-3.4) returns {0}",  
38 d2);  
39     }  
40 }
```

```
1  
2     }
```

```
3     The output is
```

```
4  
5     Math.Ceiling(3.4) returns 4
```

```
6  
7  
8     Math.Ceiling(-3.4) returns -3  
9
```

```
10
```

1 Math.Cos(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Cos(float64 d)  
4 [C#]  
5 public static double Cos(double d)
```

6 Summary

7 Returns the cosine of the specified **System.Double** that represents an
8 angle.

9 Parameters

10
11

Parameter	Description
<i>d</i>	A System.Double that represents an angle measured in radians.

12
13
14

Return Value

15 A **System.Double** containing the value of the cosine of *d*. If *d* is equal
16 to **System.Double.NaN**, **System.Double.NegativeInfinity**, or
17 **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

18 Description

19 [Note:

20

1 Math.Cosh(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Cosh(float64 value)  
4 [C#]  
5 public static double Cosh(double value)
```

6 Summary

7 Returns the hyperbolic cosine of the specified **System.Double** that
8 represents an angle.

9 Parameters

10
11

Parameter	Description
<i>value</i>	A System.Double that represents an angle measured in radians.

12
13
14

Return Value

15 The hyperbolic cosine of *value*. If *value* is equal to
16 **System.Double.NegativeInfinity** or
17 **System.Double.PositiveInfinity**, returns
18 **System.Double.PositiveInfinity**. If *value* is equal to
19 **System.Double.NaN**, returns **System.Double.NaN**.

20 Description

21 [Note:

22

1 Math.DivRem(System.Int32, 2 System.Int32, System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig static int32 DivRem(int32 a,int32  
5 b, [out] int32 &result)  
6 [C#]  
7 public static int DivRem(int a, int b, out int result)
```

8 Summary

9 Returns the quotient of two numbers, also passing the remainder as
10 an output parameter.

11 Parameters

12
13

Parameter	Description
<i>a</i>	A System.Int32 that contains the dividend.
<i>b</i>	A System.Int32 that contains the divisor.
<i>result</i>	A System.Int32 that receives the remainder.

14
15
16

Return Value

17 A **System.Int32** containing the quotient of the specified numbers.

18

1 Math.DivRem(System.Int64, 2 System.Int64, System.Int64) Method

```
3 [ILASM]  
4 .method public hidebysig static int64 DivRem(int64 a,int64  
5 b,[out] int64 &result)  
  
6 [C#]  
7 public static long DivRem(long a, long b, out long result)
```

8 Summary

9 Returns the quotient of two numbers, also passing the remainder as
10 an output parameter.

11 Parameters

12
13

Parameter	Description
<i>a</i>	A System.Int64 that contains the dividend.
<i>b</i>	A System.Int64 that contains the divisor.
<i>result</i>	A System.Int64 that receives the remainder.

14
15
16

Return Value

17 A **System.Int64** containing the quotient of the specified numbers.

18

1 Math.Exp(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Exp(float64 d)  
4 [C#]  
5 public static double Exp(double d)
```

6 Summary

7 Returns **e** raised to the specified **System.Double** that represents an
8 exponent.

9 Parameters

10
11

Parameter	Description
<i>d</i>	A System.Double that represents an exponent.

12
13
14

Return Value

15 A **System.Double** equal to the number **e** raised to the power of *d*. If
16 *d* equals **System.Double.NaN** or **System.Double.PositiveInfinity**,
17 returns that value. If *d* equals **System.Double.NegativeInfinity**,
18 returns 0.

19 Description

20 [Note: Use the **System.Math.Pow** method to calculate powers of
21 other bases.

22
23
24

System.Math.Exp is the inverse of **System.Math.Log**.]

1 Math.Floor(System.Double) Method

```
2 [ILASM]
3 .method public hidebysig static float64 Floor(float64 d)
4
5 [C#]
6 public static double Floor(double d)
```

6 Summary

7 Returns the largest integer less than or equal to the specified
8 **System.Double**.

9 Parameters

Parameter	Description
<i>d</i>	A System.Double .

12

13 Return Value

14

15 A **System.Double** containing the value of the largest integer less than
16 or equal to *d*. If *d* is equal to **System.Double.NaN**,
17 **System.Double.NegativeInfinity**, or
18 **System.Double.PositiveInfinity**, that value is returned..

19 Description

20 The behavior of this method follows IEEE Standard 754, section 4.

21 Example

22

23 The following example demonstrates using the **System.Math.Floor**
24 method.

25

```
26 [C#]
27 using System;
28
29 public class MathFloorExample
30 {
31
32     public static void Main()
33     {
34
35         Double d1 = Math.Floor(3.4);
36         Double d2 = Math.Floor(-3.4);
37         Console.WriteLine("Math.Floor(3.4) returns {0}", d1);
```

```
1         Console.WriteLine("Math.Floor(-3.4) returns {0}",
2     d2);
3
4     }
5
6 }
```

```
7     The output is
8
9     Math.Floor(3.4) returns 3
10
11
12     Math.Floor(-3.4) returns -4
13
```

14

1 Math.IEEERemainder(System.Double, 2 System.Double) Method

```
3 [ILASM]  
4 .method public hidebysig static float64  
5 IEEEERemainder(float64 x, float64 y)  
  
6 [C#]  
7 public static double IEEEERemainder(double x, double y)
```

8 Summary

9 Returns the remainder resulting from the division of one specified
10 **System.Double** by another specified **System.Double**.

11 Parameters

12
13

Parameter	Description
x	A System.Double that represents a dividend.
y	A System.Double that represents a divisor.

14
15
16

Return Value

17 A **System.Double** whose value is as follows:

Value	Description
$x - (y Q)$,	Q is the quotient of x/y rounded to the nearest integer (if x/y is exactly halfway between two integers, the even integer is returned).
+0	Q is the quotient of x/y rounded to the nearest integer (if x/y is exactly halfway between two integers, the even integer is returned), $x - (y Q)$ is zero, and x is positive.
-0	Q is the quotient of x/y rounded to the nearest integer (if x/y is exactly halfway between two integers, the even integer is returned), $x - (y Q)$ is zero, and x is negative.
System.Double.NaN	$y = 0$.

18
19

Description

20 This operation complies with the remainder operation defined in
21 Section 5.1 of ANSI/IEEE Std 754-1985; IEEE Standard for Binary
22 Floating-Point Arithmetic; Institute of Electrical and Electronics

1 Engineers, Inc; 1985.
2
3 [Note: For more information regarding the use of +0 and -0, see
4 Section 3.1 of ANSI/IEEE Std 754-1985; IEEE Standard for Binary
5 Floating-Point Arithmetic; Institute of Electrical and Electronics
6 Engineers, Inc; 1985.]

7 **Example**
8

9 The following example demonstrates using the
10 **System.Math.IEEERemainder** method.

```
11 [C#]  
12  
13 using System;  
14  
15 public class MathIEEERemainderExample  
16 {  
17  
18     public static void Main()  
19     {  
20  
21         Double d1 = Math.IEEERemainder(3.54,0);  
22         Double d2 = Math.IEEERemainder(9.99,-3.33);  
23         Double d3 = Math.IEEERemainder(-9.99,3.33);  
24         Double d4 = Math.IEEERemainder(9.5,1.5);  
25         Console.WriteLine("Math.IEEERemainder(3.54,0) returns  
26 {0}", d1);  
27         Console.WriteLine("Math.IEEERemainder(9.99,-3.33)  
28 returns {0}", d2);  
29         Console.WriteLine("Math.IEEERemainder(-9.99,3.33)  
30 returns {0}", d3);  
31         Console.WriteLine("Math.IEEERemainder(9.5,1.5)  
32 returns {0}", d4);  
33  
34     }  
35  
36 }
```

37 The output is

38
39 Math.IEEERemainder(3.54,0) returns NaN

40
41
42 Math.IEEERemainder(9.99,-3.33) returns 0
43

1
2
3
4
5
6
7

`Math.IEEEremainder(-9.99,3.33)` returns 0

`Math.IEEEremainder(9.5,1.5)` returns 0.5

1 Math.Log(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Log(float64 d)  
4 [C#]  
5 public static double Log(double d)
```

6 Summary

7 Returns the natural logarithm of the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>d</i>	A System.Double whose natural logarithm is to be found.

11
12
13

Return Value

14 Returns a **System.Double** whose value is as follows.

Condition	Returns
$d > 0$.	The value of the natural logarithm of <i>d</i> .
$d == 0$.	System.Double.NegativeInfinity .
$d < 0$. -or- <i>d</i> is equal to System.Double.NegativeInfinity . -or- <i>d</i> is equal to System.Double.NaN .	System.Double.NaN .
<i>d</i> is equal to System.Double.PositiveInfinity .	System.Double.PositiveInfinity .

15

16 Description

17 *d* is specified as a base 10 number.

18

1 Math.Log(System.Double, System.Double)

2 Method

```
3 [ILASM]
4 .method public hidebysig static float64 Log(float64 a,
5 float64 newBase)
6
7 [C#]
8 public static double Log(double a, double newBase)
```

8 Summary

9 Returns the logarithm of the specified **System.Double** in the specified
10 base.

11 Parameters

12
13

Parameter	Description
<i>a</i>	A System.Double whose logarithm is to be found.
<i>newBase</i>	A System.Double containing the value of the base of the logarithm.

14
15
16

15 Return Value

17 Returns a **System.Double** whose value is as follows:

Condition	Returns
$a > 0$.	The value of $\text{Log}_{\text{newBase}}a$, if and only if <i>newBase</i> is greater than or equal to 0; otherwise, System.Double.NaN .
$a == 0$.	System.Double.NegativeInfinity .
$a < 0$.	System.Double.NaN .

18
19
20
21
22
23
24
25
26
27
28
29
30

If *a* is equal to **System.Double.PositiveInfinity** and *newBase* is not equal **System.Double.PositiveInfinity**, **System.Double.NegativeInfinity**, or **System.Double.NaN**, returns **System.Double.PositiveInfinity**. If *newBase* is equal to **System.Double.PositiveInfinity** and *a* is not equal to **System.Double.PositiveInfinity**, **System.Double.NegativeInfinity**, or **System.Double.NaN**, returns 0. If both *a* and *newBase* are equal to **System.Double.PositiveInfinity**, or *a* or *newBase* is equal to **System.Double.NaN** or **System.Double.NegativeInfinity**, returns **System.Double.NaN**.

1 Math.Log10(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Log10(float64 d)  
4 [C#]  
5 public static double Log10(double d)
```

6 Summary

7 Returns \log_{10} of the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>d</i>	A System.Double whose logarithm is to be found.

11
12
13

Return Value

14 Returns a **System.Double** as indicated by the following table.

Condition	Returns
$d > 0$.	A System.Double containing the value of $\log_{10}d$.
$d == 0$.	System.Double.NegativeInfinity .
$d < 0$. -or- d is equal to System.Double.NegativeInfinity . -or- d is equal to System.Double.NaN .	System.Double.NaN .
d is equal to System.Double.PositiveInfinity .	System.Double.PositiveInfinity .

15
16

1 Math.Max(System.SByte, System.SByte)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int8 Max(int8 val1, int8  
5 val2)  
  
6 [C#]  
7 public static sbyte Max(sbyte val1, sbyte val2)
```

8 Summary

9 Returns the greater of two specified **System.SByte** values.

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.Byte values to compare.
<i>val2</i>	The second of two specified System.Byte values to compare.

15
16
17

16 Return Value

18 A **System.SByte** that is equal to *val1* if *val1* is greater than or equal
19 to *val2*; otherwise, the return value is equal to *val2*.

20 Description

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Max(System.Int16, System.Int16)**.

23

1 Math.Max(System.Byte, System.Byte)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static unsigned int8 Max(unsigned  
5 int8 val1, unsigned int8 val2)  
6  
7 [C#]  
8 public static byte Max(byte val1, byte val2)
```

8 Summary

9 Returns the greater of two specified **System.Byte** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Byte values to compare.
<i>val2</i>	The second of two specified System.Byte values to compare.

13
14
15

14 Return Value

16 A **System.Byte** that is equal to *val1* if *val1* is greater than or equal to
17 *val2*; otherwise, the return value is equal to *val2*.

18

1 Math.Max(System.Int16, System.Int16)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int16 Max(int16 val1, int16  
5 val2)  
  
6 [C#]  
7 public static short Max(short val1, short val2)
```

8 Summary

9 Returns the greater of two specified **System.Int16** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int16 values to compare.
<i>val2</i>	The second of two specified System.Int16 values to compare.

13
14
15

Return Value

16 A **System.Int16** that is equal to *val1* if *val1* is greater than or equal
17 to *val2*; otherwise, the return value is equal to *val2*.

18

1 **Math.Max(System.UInt16,** 2 **System.UInt16) Method**

```
3 [ILASM]  
4 .method public hidebysig static unsigned int16 Max(unsigned  
5 int16 val1, unsigned int16 val2)  
  
6 [C#]  
7 public static ushort Max(ushort val1, ushort val2)
```

8 **Summary**

9 Returns the greater of two specified **System.UInt16** values.

10 **Type Attributes:**

- 11 • CLSCompliantAttribute(false)

12 **Parameters**

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt16 values to compare.
<i>val2</i>	The second of two specified System.UInt16 values to compare.

15
16
17

16 **Return Value**

18 A **System.UInt16** that is equal to *val1* if *val1* is greater than or equal
19 to *val2*; otherwise, the return value is equal to *val2*.

20 **Description**

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Max(System.Int32, System.Int32)**.

23

1 Math.Max(System.Int32, System.Int32)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int32 Max(int32 val1, int32  
5 val2)  
6 [C#]  
7 public static int Max(int val1, int val2)
```

8 Summary

9 Returns the greater of two specified **System.Int32** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int32 values to compare.
<i>val2</i>	The second of two specified System.Int32 values to compare.

13
14
15

Return Value

16 A **System.Int32** that is equal to *val1* if *val1* is greater than or equal
17 to *val2*; otherwise, the return value is equal to *val2*.

18

1 **Math.Max(System.UInt32,** 2 **System.UInt32) Method**

```
3 [ILASM]  
4 .method public hidebysig static unsigned int32 Max(unsigned  
5 int32 val1, unsigned int32 val2)  
  
6 [C#]  
7 public static uint Max(uint val1, uint val2)
```

8 **Summary**

9 Returns the greater of two specified **System.UInt32** values.

10 **Type Attributes:**

- 11 • CLSCompliantAttribute(false)

12 **Parameters**

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt32 values to compare.
<i>val2</i>	The second of two specified System.UInt32 values to compare.

15
16
17

16 **Return Value**

18 A **System.UInt32** that is equal to *val1* if *val1* is greater than or equal
19 to *val2*; otherwise, the return value is equal to *val2*.

20 **Description**

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Max(System.Int64, System.Int64)**.

23

1 Math.Max(System.Int64, System.Int64)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int64 Max(int64 val1, int64  
5 val2)  
6 [C#]  
7 public static long Max(long val1, long val2)
```

8 Summary

9 Returns the greater of two specified **System.Int64** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int64 values to compare.
<i>val2</i>	The second of two specified System.Int64 values to compare.

13
14
15

Return Value

16 A **System.Int64** that is equal to *val1* if *val1* is greater than or equal
17 to *val2*; otherwise, the return value is equal to *val2*.

18

1 Math.Max(System.UInt64, 2 System.UInt64) Method

```
3 [ILASM]  
4 .method public hidebysig static unsigned int64 Max(unsigned  
5 int64 val1, unsigned int64 val2)  
  
6 [C#]  
7 public static ulong Max(ulong val1, ulong val2)
```

8 Summary

9 Returns the greater of two specified **System.UInt64** values.

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt64 values to compare.
<i>val2</i>	The second of two specified System.UInt64 values to compare.

15
16
17

16 Return Value

18 A **System.UInt64** equal to *val1* if *val1* is greater than or equal to *val2*;
19 otherwise, the return value is equal to *val2*.

20 Description

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Max(System.Decimal, System.Decimal)**.

23

1 Math.Max(System.Single, System.Single)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static float32 Max(float32 val1,  
5 float32 val2)  
6 [C#]  
7 public static float Max(float val1, float val2)
```

8 Summary

9 Returns the greater of two specified **System.Single** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Single values to compare.
<i>val2</i>	The second of two specified System.Single values to compare.

13
14
15

14 Return Value

16 A **System.Single** equal to *val1* if *val1* is greater than or equal to *val2*;
17 otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are
18 equal to **System.Single.NaN**, **System.Single.NaN** is returned.

19

1 Math.Max(System.Double, 2 System.Double) Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Max(float64 val1,  
5 float64 val2)  
6 [C#]  
7 public static double Max(double val1, double val2)
```

8 Summary

9 Returns the greater of two specified **System.Double** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Double values to compare.
<i>val2</i>	The second of two specified System.Double values to compare.

13
14
15

Return Value

16 A **System.Double** equal to *val1* if *val1* is greater than or equal to
17 *val2*; otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both
18 are equal to **System.Double.NaN**, **System.Double.NaN** is returned.

19

1 Math.Max(System.Decimal, 2 System.Decimal) Method

```
3 [ILASM]  
4 .method public hidebysig static decimal Max(decimal val1,  
5 decimal val2)  
  
6 [C#]  
7 public static decimal Max(decimal val1, decimal val2)
```

8 Summary

9 Returns the greater of two specified **System.Decimal** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Decimal values to compare.
<i>val2</i>	The second of two specified System.Decimal values to compare.

13
14
15

Return Value

16 A **System.Decimal** that is equal to *val1* if *val1* is greater than or
17 equal to *val2*; otherwise, the return value is equal to *val2*.

18

1 **Math.Min(System.SByte, System.SByte)**

2 **Method**

```
3 [ILASM]  
4 .method public hidebysig static int8 Min(int8 val1, int8  
5 val2)  
6 [C#]  
7 public static sbyte Min(sbyte val1, sbyte val2)
```

8 **Summary**

9 Returns the lesser of two specified **System.SByte** values.

10 **Type Attributes:**

- 11 • CLSCompliantAttribute(false)

12 **Parameters**

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.SByte values to compare.
<i>val2</i>	The second of two specified System.SByte values to compare.

15
16
17

16 **Return Value**

18 A **System.SByte** equal to *val1* if *val1* is less than or equal to *val2*;
19 otherwise, the return value is equal to *val2*.

20 **Description**

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Min(System.Int16, System.Int16)**.

23

1 Math.Min(System.Byte, System.Byte)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static unsigned int8 Min(unsigned  
5 int8 val1, unsigned int8 val2)  
6  
7 [C#]  
8 public static byte Min(byte val1, byte val2)
```

8 Summary

9 Returns the lesser of two specified **System.Byte** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Byte values to compare.
<i>val2</i>	The second of two specified System.Byte values to compare.

13
14
15

14 Return Value

16 A **System.Byte** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*.

18

1 Math.Min(System.Int16, System.Int16)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int16 Min(int16 val1, int16  
5 val2)  
6 [C#]  
7 public static short Min(short val1, short val2)
```

8 Summary

9 Returns the lesser of two specified **System.Int16** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int16 values to compare.
<i>val2</i>	The second of two specified System.Int16 values to compare.

13
14
15

Return Value

16 A **System.Int16** that is equal to *val1* if *val1* is less than or equal to
17 *val2*; otherwise, the return value is equal to *val2*.

18

1 Math.Min(System.UInt16, System.UInt16)

2 Method

```
3 [ILASM]
4 .method public hidebysig static unsigned int16 Min(unsigned
5 int16 val1, unsigned int16 val2)
6
7 [C#]
8 public static ushort Min(ushort val1, ushort val2)
```

8 Summary

9 Returns the lesser of two specified **System.UInt16** values.

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt16 values to compare.
<i>val2</i>	The second of two specified System.UInt16 values to compare.

15
16
17

16 Return Value

18 A **System.UInt16** equal to *val1* if *val1* is less than or equal to *val2*;
19 otherwise, the return value is equal to *val2*.

20 Description

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Min(System.Int32, System.Int32)**.

23

1 Math.Min(System.Int32, System.Int32)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int32 Min(int32 val1, int32  
5 val2)  
6 [C#]  
7 public static int Min(int val1, int val2)
```

8 Summary

9 Returns the lesser of two specified **System.Int32** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int32 values to compare.
<i>val2</i>	The second of two specified System.Int32 values to compare.

13
14
15

Return Value

16 A **System.Int32** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*.

18

1 Math.Min(System.UInt32, System.UInt32)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static unsigned int32 Min(unsigned  
5 int32 val1, unsigned int32 val2)  
  
6 [C#]  
7 public static uint Min(uint val1, uint val2)
```

8 Summary

9 Returns the lesser of two specified **System.UInt32** values.

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt32 values to compare.
<i>val2</i>	The second of two specified System.UInt32 values to compare.

15
16
17

16 Return Value

18 A **System.UInt32** equal to *val1* if *val1* is less than or equal to *val2*;
19 otherwise, the return value is equal to *val2*.

20 Description

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Min(System.Int64, System.Int64)**.

23

1 Math.Min(System.Int64, System.Int64)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static int64 Min(int64 val1, int64  
5 val2)  
6 [C#]  
7 public static long Min(long val1, long val2)
```

8 Summary

9 Returns the lesser of two specified **System.Int64** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Int64 values to compare.
<i>val2</i>	The second of two specified System.Int64 values to compare.

13
14
15

14 Return Value

16 A **System.Int64** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*.

18

1 Math.Min(System.UInt64, System.UInt64)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static unsigned int64 Min(unsigned  
5 int64 val1, unsigned int64 val2)  
  
6 [C#]  
7 public static ulong Min(ulong val1, ulong val2)
```

8 Summary

9 Returns the lesser of two specified **System.UInt64** values.

10 Type Attributes:

- 11 • CLSCompliantAttribute(false)

12 Parameters

13
14

Parameter	Description
<i>val1</i>	The first of two specified System.UInt64 values to compare.
<i>val2</i>	The second of two specified System.UInt64 values to compare.

15
16
17

16 Return Value

18 A **System.UInt64** equal to *val1* if *val1* is less than or equal to *val2*;
19 otherwise, the return value is equal to *val2*.

20 Description

21 This method is not CLS-compliant. For a CLS-compliant alternative,
22 use **System.Math.Min(System.Decimal, System.Decimal)**.

23

1 Math.Min(System.Single, System.Single)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static float32 Min(float32 val1,  
5 float32 val2)  
6 [C#]  
7 public static float Min(float val1, float val2)
```

8 Summary

9 Returns the lesser of two specified **System.Single** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Single values to compare.
<i>val2</i>	The second of two specified System.Single values to compare.

13
14
15

14 Return Value

16 A **System.Single** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are
18 equal to **System.Single.NaN**, **System.Single.NaN** is returned.

19

1 Math.Min(System.Double, System.Double)

2 Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Min(float64 val1,  
5 float64 val2)  
6 [C#]  
7 public static double Min(double val1, double val2)
```

8 Summary

9 Returns the lesser of two specified **System.Double** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Double values to compare.
<i>val2</i>	The second of two specified System.Double values to compare.

13
14
15

14 Return Value

16 A **System.Double** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*. If *val1*, *val2*, or both are
18 equal to **System.Double.NaN**, **System.Double.NaN** is returned.

19

1 Math.Min(System.Decimal, 2 System.Decimal) Method

```
3 [ILASM]  
4 .method public hidebysig static decimal Min(decimal val1,  
5 decimal val2)  
6  
7 [C#]  
8 public static decimal Min(decimal val1, decimal val2)
```

8 Summary

9 Returns the lesser of two specified **System.Decimal** values.

10 Parameters

11
12

Parameter	Description
<i>val1</i>	The first of two specified System.Decimal values to compare.
<i>val2</i>	The second of two specified System.Decimal values to compare.

13
14
15

14 Return Value

16 A **System.Decimal** equal to *val1* if *val1* is less than or equal to *val2*;
17 otherwise, the return value is equal to *val2*.

18

1 Math.Pow(System.Double, 2 System.Double) Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Pow(float64 x,  
5 float64 y)  
  
6 [C#]  
7 public static double Pow(double x, double y)
```

8 Summary

9 Returns the specified **System.Double** raised to the specified power.

10 Parameters

11
12

Parameter	Description
x	A System.Double to be raised to a power.
y	A System.Double that specifies that power.

13
14
15

Return Value

16 A **System.Double** equal to x raised to the power y. The following
17 table specifies the results if x or y is equal to **System.Double.NaN**,
18 **System.Double.NegativeInfinity**, or
19 **System.Double.PositiveInfinity**.

Parameter Values	Returns
x or y is equal to System.Double.NaN	System.Double.NaN .
x is equal to System.Double.NegativeInfinity	System.Double.NegativeInfinity if y is an odd integer; otherwise, System.Double.PositiveInfinity .
y is equal to System.Double.NegativeInfinity	0.
x is equal to System.Double.PositiveInfinity	0 if y is equal to System.Double.NegativeInfinity ; otherwise, System.Double.PositiveInfinity .
y is equal to System.Double.PositiveInfinity	System.Double.PositiveInfinity .

20
21

1 Math.Round(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Round(float64 a)  
4 [C#]  
5 public static double Round(double a)
```

6 Summary

7 Returns the integer nearest the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>a</i>	A System.Double to be rounded.

11
12
13

12 Return Value

14 A **System.Double** containing the value of the integer nearest *a*. If *a* is
15 exactly halfway between two integers, one of which is even and the
16 other odd, then the even integer is returned.

17 Description

18 The behavior of this method follows IEEE Standard 754, section 4.1.

19 Example

20

21 The following example demonstrates using the
22 **System.Math.Round(System.Double)** method.

23
24

```
24 [C#]  
25 using System;  
26  
27 public class MathRoundExample  
28 {  
29  
30     public static void Main()  
31     {  
32  
33         Double d1 = Math.Round(4.4);  
34         Double d2 = Math.Round(4.5);  
35         Double d3 = Math.Round(4.6);  
36         Console.WriteLine("Math.Round(4.4) returns {0}", d1);  
37         Console.WriteLine("Math.Round(4.5) returns {0}", d2);  
38         Console.WriteLine("Math.Round(4.6) returns {0}", d3);
```

```
1
2     }
3
4     }

5     The output is
6
7     Math.Round(4.4) returns 4
8
9
10    Math.Round(4.5) returns 4
11
12
13    Math.Round(4.6) returns 5
14
```

15

1 Math.Round(System.Double, 2 System.Int32) Method

```
3 [ILASM]  
4 .method public hidebysig static float64 Round(float64  
5 value, int32 digits)  
  
6 [C#]  
7 public static double Round(double value, int digits)
```

8 Summary

9 Returns the number nearest the specified **System.Double** within the
10 specified precision.

11 Parameters

12
13

Parameter	Description
<i>value</i>	A System.Double to be rounded.
<i>digits</i>	A System.Int32 containing the value of the number of significant fractional digits (precision) in the return value. This number is required to be greater than or equal to 0 and less than or equal to 15.

14
15
16

Return Value

17 A **System.Double** containing the value of the number nearest *value*
18 with a precision equal to *digits*. If the digit in *value* that is in the $10^{-(digits + 1)}$
19 $10^{-(digits + 1)}$ place is equal to 5 and there are no non-zero numbers in any
20 less significant place, then the digit in the $10^{-digits}$ place will be
21 unchanged if it is even, else it will be set to the closest even integer
22 value in the direction of the digit in the $10^{-(digits + 1)}$ place. If the
23 precision of *value* is less than *digits*, then *value* is returned unchanged.
24 If *digits* is zero, this method behaves in the same manner as
25 **System.Math.Round** (*value*).

26 Description

27 The behavior of this method follows IEEE Standard 754, section 4.1.

28 Exceptions

29
30

Exception	Condition
System.ArgumentOutOfRangeException	<i>digits</i> < 0.

Example

The following example demonstrates using the **System.Math.Round(System.Double, System.Int32)** method.

[C#]

```
using System;

public class MathRoundExample
{
    public static void Main()
    {
        Double d1 = Math.Round(3.44,1);
        Double d2 = Math.Round(3.45,1);
        Double d3 = Math.Round(3.55,1);
        Console.WriteLine("Math.Round(3.44, 1) returns {0}",
d1);
        Console.WriteLine("Math.Round(3.45, 1) returns {0}",
d2);
        Console.WriteLine("Math.Round(3.55, 1) returns {0}",
d3);
    }
}
```

The output is

Math.Round(3.44, 1) returns 3.4

Math.Round(3.45, 1) returns 3.4

Math.Round(3.55, 1) returns 3.6

1 Math.Round(System.Decimal) Method

```
2 [ILASM]  
3 .method public hidebysig static decimal Round(decimal d)  
4 [C#]  
5 public static decimal Round(decimal d)
```

6 Summary

7 Returns the integer nearest the specified **System.Decimal**.

8 Parameters

Parameter	Description
<i>d</i>	A System.Decimal to be rounded.

11 Return Value

12 A **System.Decimal** containing the value of the integer nearest *d*. If *d*
13 is exactly halfway between two integers, one of which is even and the
14 other odd, then the even integer is returned.
15
16

17 Description

18 The behavior of this method follows IEEE Standard 754, section 4.1.

19 Example

20
21 The following example demonstrates using the
22 **System.Math.Round(System.Decimal)** method.
23

```
24 [C#]  
25 using System;  
26  
27 public class MathRoundExample  
28 {  
29  
30     public static void Main()  
31     {  
32  
33         Double d1 = Math.Round(4.4);  
34         Double d2 = Math.Round(4.5);  
35         Double d3 = Math.Round(4.6);  
36         Console.WriteLine("Math.Round(4.4) returns {0}", d1);  
37         Console.WriteLine("Math.Round(4.5) returns {0}", d2);  
38         Console.WriteLine("Math.Round(4.6) returns {0}", d3);
```

```
1
2     }
3
4     }

5     The output is
6
7     Math.Round(4.4) returns 4
8
9
10    Math.Round(4.5) returns 4
11
12
13    Math.Round(4.6) returns 5
14
```

15

1 Math.Sign(System.SByte) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(int8 value)  
4 [C#]  
5 public static int Sign(sbyte value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.SByte**.

8 Type Attributes:

- 9 • CLSCompliantAttribute(false)

10 Parameters

11
12

Parameter	Description
<i>value</i>	A System.SByte whose sign is to be determined.

13

14 Return Value

15

16 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

17

18 Description

19 This method is not CLS-compliant. For a CLS-compliant alternative,
20 use **System.Math.Sign(System.Int16)**.

21

1 Math.Sign(System.Int16) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(int16 value)  
4 [C#]  
5 public static int Sign(short value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Int16**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int16 whose sign is to be determined.

11
12
13

Return Value

14 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15
16

1 Math.Sign(System.Int32) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(int32 value)  
4  
5 [C#]  
6 public static int Sign(int value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Int32**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int32 whose sign is to be determined.

11
12
13

12 Return Value

14 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15
16

1 Math.Sign(System.Int64) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(int64 value)  
4  
5 [C#]  
6 public static int Sign(long value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Int64**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Int64 whose sign is to be determined.

11
12
13

12 Return Value

14 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15
16

1 Math.Sign(System.Single) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(float32 value)  
4 [C#]  
5 public static int Sign(float value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Single**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Single whose sign is to be determined.

11
12
13

12 Return Value

14 A **System.Int32** indicating the sign of value.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15

16 Exceptions

17
18

Exception	Condition
System.ArithmeticException	<i>value</i> is equal to System.Single.NaN .

19
20
21

1 Math.Sign(System.Double) Method

```
2 [ILASM]
3 .method public hidebysig static int32 Sign(float64 value)
4
5 [C#]
6 public static int Sign(double value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Double whose sign is to be determined.

11
12
13

12 Return Value

14 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15

16 Exceptions

17
18

Exception	Condition
System.ArithmeticException	<i>value</i> is equal to System.Double.NaN .

19
20
21

20 Example

22 The following example demonstrates using the
23 **System.Math.Sign(System.Double)** method.

24
25

[C#]

```
26 using System;
```

```
1 public class MathSignExample
2 {
3
4     public static void Main()
5     {
6
7         Double d1 = Math.Sign(4.4);
8         Double d2 = Math.Sign(0.0);
9         Double d3 = Math.Sign(-4.5);
10        Console.WriteLine("Math.Sign(4.4) returns {0}", d1);
11        Console.WriteLine("Math.Sign(0.0) returns {0}", d2);
12        Console.WriteLine("Math.Sign(-4.5) returns {0}", d3);
13
14    }
15
16 }
```

```
17 The output is
18
19 Math.Sign(4.4) returns 1
20
21
22 Math.Sign(0.0) returns 0
23
24
25 Math.Sign(-4.5) returns -1
26
```

27

1 Math.Sign(System.Decimal) Method

```
2 [ILASM]  
3 .method public hidebysig static int32 Sign(decimal value)  
4 [C#]  
5 public static int Sign(decimal value)
```

6 Summary

7 Returns a value indicating the sign of the specified **System.Decimal**.

8 Parameters

9
10

Parameter	Description
<i>value</i>	A System.Decimal number whose sign is to be determined.

11
12
13

Return Value

14 A **System.Int32** indicating the sign of *value*.

Number	Description
-1	<i>value</i> < 0.
0	<i>value</i> == 0.
1	<i>value</i> > 0.

15
16

1 Math.Sin(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Sin(float64 a)  
4 [C#]  
5 public static double Sin(double a)
```

6 Summary

7 Returns the sine of the specified **System.Double** that represents an
8 angle.

9 Parameters

Parameter	Description
<i>a</i>	A System.Double containing the value of an angle measured in radians.

12

13 Return Value

14

15 A **System.Double** containing the value of the sine of *a*. If *a* is equal to
16 **System.Double.NaN**, **System.Double.NegativeInfinity**, or
17 **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

18 Description

19 *[Note:*

20 Example

21

22 The following example demonstrates using the **System.Math.Sin**
23 method.

24

25

```
26 using System;  
27  
28 public class MathSinExample  
29 {  
30  
31     public static void Main()  
32     {  
33  
34         Double d1 = Math.Sin(0);  
35         Double d2 = Math.Sin(Math.PI/2.0);  
36         Console.WriteLine("Math.Sin(0) returns {0}", d1);
```

```
1         Console.WriteLine("Math.Sin(Math.PI/2.0) returns
2     {0}", d2);
3
4     }
5
6 }
```

```
7     The output is
8
9     Math.Sin(0) returns 0
10
11
12     Math.Sin(Math.PI/2.0) returns 1
13
```

14

1 Math.Sinh(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Sinh(float64 value)  
4 [C#]  
5 public static double Sinh(double value)
```

6 Summary

7 Returns the hyperbolic sine of the specified **System.Double** that
8 represents an angle.

9 Parameters

10
11

Parameter	Description
<i>value</i>	A System.Double containing the value of an angle measured in radians.

12
13
14

Return Value

15 A **System.Double** containing the value of the hyperbolic sine of *value*.
16 If *value* is equal to **System.Double.NegativeInfinity**,
17 **System.Double.PositiveInfinity**, or **System.Double.NaN**, returns a
18 **System.Double** equal to *value*.

19 Description

20 [Note:

21 Example

22

23 The following example demonstrates using the **System.Math.Sinh**
24 method.

25
26

```
[C#]  
27 using System;  
28  
29 public class MathSinhExample  
30 {  
31  
32     public static void Main()  
33     {  
34  
35         Double d1 = Math.Sinh(0);  
36         Double d2 = Math.Sinh(Math.PI);  
37         Console.WriteLine("Math.Sinh(0) returns {0}", d1);
```

```
1         Console.WriteLine("Math.Sinh(Math.PI) returns {0}",
2     d2);
3
4     }
5
6 }
```

7 The output is

8
9 Math.Sinh(0) returns 0

10
11
12 Math.Sinh(Math.PI) returns 11.5487393572577

13

14

1 Math.Sqrt(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Sqrt(float64 d)  
4 [C#]  
5 public static double Sqrt(double d)
```

6 Summary

7 Returns the square root of the specified **System.Double**.

8 Parameters

9
10

Parameter	Description
<i>d</i>	A System.Double .

11
12
13

Return Value

14 A **System.Double** whose value is indicated as follows:

Condition	Returns
<i>d</i> == 0	A System.Double containing the positive square root of <i>d</i> .
<i>d</i> < 0 <i>d</i> is equal to System.Double.NegativeInfinity . <i>d</i> is equal to System.Double.NaN .	System.Double.NaN .
<i>d</i> is equal to System.Double.PositiveInfinity	System.Double.PositiveInfinity .

15

16 Example

17

18 The following example demonstrates using the **System.Math.Sqrt**
19 method.

20
21

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

```
1
2 public class MathSqrtExample
3 {
4
5     public static void Main()
6     {
7
8         Double d1 = Math.Sqrt(16.0);
9         Double d2 = Math.Sqrt(0.0);
10        Double d3 = Math.Sqrt(-10.0);
11        Console.WriteLine("Math.Sqrt(16.0) returns {0}", d1);
12        Console.WriteLine("Math.Sqrt(0.0) returns {0}", d2);
13        Console.WriteLine("Math.Sqrt(-10.0) returns {0}",
14        d3);
15
16    }
17
18 }
```

```
19 The output is
20
21 Math.Sqrt(16.0) returns 4
22
23
24 Math.Sqrt(0.0) returns 0
25
26
27 Math.Sqrt(-10.0) returns NaN
28
```

29

1 Math.Tan(System.Double) Method

```
2 [ILASM]  
3 .method public hidebysig static float64 Tan(float64 a)  
4 [C#]  
5 public static double Tan(double a)
```

6 Summary

7 Returns the tangent of the specified **System.Double** that represents
8 an angle.

9 Parameters

10
11

Parameter	Description
<i>a</i>	A System.Double that represents an angle measured in radians.

12
13
14

Return Value

15 A **System.Double** containing the value of the tangent of *a*. If *a* is
16 equal to **System.Double.NaN**, **System.Double.NegativeInfinity**,
17 or **System.Double.PositiveInfinity**, returns **System.Double.NaN**.

18 Description

19 [Note:

20 Example

21

22 The following example demonstrates using the **System.Math.Tan**
23 method.

24
25

```
[C#]  
26 using System;  
27  
28 public class MathTanExample  
29 {  
30  
31     public static void Main()  
32     {  
33  
34         Double d1 = Math.Tan(0);  
35         Double d2 = Math.Tan(Math.PI/2.0);  
36         Console.WriteLine("Math.Tan(0) returns {0}", d1);  
37         Console.WriteLine("Math.Tan(Math.PI/2.0) returns  
38         {0}", d2);
```

1
2
3
4

5
6
7
8
9
10
11

```
    }  
}
```

The output is

Math.Tan(0) returns 0

Math.Tan(Math.PI/2.0) returns 1.63317787283838E+16

12

1 Math.Tanh(System.Double) Method

```
2 [ILASM]
3 .method public hidebysig static float64 Tanh(float64 value)
4 [C#]
5 public static double Tanh(double value)
```

6 Summary

7 Returns the hyperbolic tangent of the specified **System.Double** that
8 represents an angle.

9 Parameters

10
11

Parameter	Description
<i>value</i>	A System.Double that represents an angle measured in radians.

12
13
14

13 Return Value

15 A **System.Double** containing the value of the hyperbolic tangent of
16 *value*. If *value* is equal to **System.Double.NegativeInfinity**, returns
17 -1. If *value* is equal to **System.Double.PositiveInfinity**, returns 1. If
18 *value* is equal to **System.Double.NaN**, returns **System.Double.NaN**.

19 Description

20 [Note:

21 Example

22

23 The following example demonstrates using the **System.Math.Tanh**
24 method.

25
26

```
26 [C#]
27 using System;
28
29 public class MathTanhExample
30 {
31
32     public static void Main()
33     {
34
35         Double d1 = Math.Tanh(0);
36         Double d2 = Math.Tanh(Math.PI);
37         Console.WriteLine("Math.Tanh(0) returns {0}", d1);
```

```
1         Console.WriteLine("Math.Tanh(Math.PI) returns {0}",
2     d2);
3
4     }
5
6 }
```

7 The output is

8
9 Math.Tanh(0) returns 0

10
11
12 Math.Tanh(Math.PI) returns 0.99627207622075

13

14