

1 System.Math Class

```
2 [ILAsm]  
3 .class public sealed Math extends System.Object  
4 [C#]  
5 public sealed class Math
```

6 Assembly Info:

- 7 • *Name:* mscorlib
- 8 • *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00]
- 9 • *Version:* 2.0.x.x
- 10 • *Attributes:*
 - 11 ○ CLSCompliantAttribute(true)

12 Summary

13 Provides constants and static methods for trigonometric, logarithmic, and other common
14 mathematical functions.

15
16 Those methods that operate on binary floating-point numbers shall follow the IEEE
17 Standard 754-2008 floating-point arithmetic recommended operations, as described in
18 section 9 of that standard, when applicable; just as specified for binary floating-point
19 calculations (this standard, Partition I, section 12.1.3).

20
21 In particular, unless otherwise specified, when a method is given a NaN argument it
22 should produce a NaN result, and also preserve the NaN payload (diagnostic error bits,
23 see IEEE 754-2008 section 6.2.1) of the argument in the result. When given multiple
24 NaN arguments, a method must produce a NaN result that preserves the payload of one
25 of those arguments.

26 Inherits From: System.Object

27
28 **Library:** ExtendedNumerics

29
30 **Thread Safety:** All public static members of this type are safe for multithreaded operations.
31 No instance members are guaranteed to be thread safe.

32

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 double precision.

8

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 to its diameter
8 rounded to double precision.

9

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

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

9 Return Value

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

12 Example

13 The following example demonstrates the System.Math.Abs(System.Decimal) method.

```
14 [C#]  
15  
16 using System;  
17  
18 public class MathAbsExample  
19 {  
20     public static void Main()  
21     {  
22         Decimal d1 = Math.Abs( (Decimal)0.00 );  
23         Decimal d2 = Math.Abs( (Decimal)(-1.23) );  
24         Console.WriteLine("Math.Abs( (Decimal)0.00 ) returns {0}",d1);  
25         Console.WriteLine("Math.Abs( (Decimal)(-1.23) ) returns {0}",d2);  
26     }  
27 }  
28 }
```

29 The output is

30 Math.Abs((Decimal)0.00) returns 0

31
32
33 Math.Abs((Decimal)(-1.23)) returns 1.23

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

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

9

10 Return Value

11 A `System.Double` containing the absolute value of *value*. If *value* is equal to
12 `System.Double.NegativeInfinity` or `System.Double.PositiveInfinity`, returns
13 `System.Double.PositiveInfinity`. If *value* is a NaN, returns that NaN.

14

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

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

9

10 Return Value

11 A `System.Single` containing the absolute value of *value*. If *value* is equal to
12 `System.Single.NegativeInfinity` or `System.Single.PositiveInfinity`, returns
13 `System.Single.PositiveInfinity`. If *value* is a NaN, returns that NaN.

14

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

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

9

10 Return Value

11 A `System.Int64` containing the absolute value of *value*.

12 Exceptions

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

13

14

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

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

9

10 Return Value

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

12 Exceptions

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

13

14

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

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

9

10 Return Value

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

12 Exceptions

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

13

14

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 <code>System.SByte</code> .

11

12 Return Value

13 A `System.SByte` containing the absolute value of *value*.

14 Description

15 This method is not CLS-compliant. For a CLS-compliant alternative, use
16 `System.Math.Abs(System.Int16)`.

17 Exceptions

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

18

19

1 Math.Acos(System.Double) Method

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

6 Summary

7 Returns the angle whose cosine is the specified `System.Double`.

8 Parameters

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

9

10 Return Value

11 A `System.Double` containing the value of an angle, θ , measured in radians, for which *d*
12 is the cosine, such that $0 \leq \theta \leq \pi$. If $d < -1$ or $d > 1$ returns `System.Double.NaN`; if *d* is a
13 NaN, returns that NaN.

14 Description

15 [*Note:* Multiply the return value by $180/\pi$ to convert from radians to degrees.]
16
17

18

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

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

9

10 Return Value

11 A `System.Double` containing the value of an angle, θ , measured in radians, for which *d*
12 is the sine, such that $-\pi/2 \leq \theta \leq \pi/2$. If *d* < -1 or *d* > 1 returns `System.Double.NaN`; if *d*
13 is a NaN, returns that NaN.

14 Description

15 [*Note:* A positive return value represents a counterclockwise angle from the positive x-
16 axis; a negative return value represents a clockwise angle.

17

18 Multiply the return value by $180/\pi$ to convert from radians to degrees.

19

20]

21

1 Math.Atan(System.Double) Method

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

6 Summary

7 Returns the angle whose tangent is the specified System.Double.

8 Parameters

Parameter	Description
<i>d</i>	A System.Double that represents a tangent.

9 Return Value

11 A System.Double containing the value of the angle, θ , measured in radians, for which *d*
12 is the tangent, such that $-\pi/2 \leq \theta \leq \pi/2$.

13
14 The following table specifies the return value if *d* is a NaN, or equal to,
15 System.Double.NegativeInfinity or System.Double.PositiveInfinity.

Return Value	Condition
<i>d</i>	<i>d</i> is a NaN.
$-\pi/2$ rounded to double precision (-1.5707963267949)	<i>d</i> is equal to System.Double.NegativeInfinity.
$\pi/2$ rounded to double precision (1.5707963267949)	<i>d</i> is equal to System.Double.PositiveInfinity.

16 Description

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

20
21 Multiply the return value by $180/\pi$ to convert from radians to degrees.

1
2]
3

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static float64 Atan2(float64 y, float64 x)  
5 [C#]  
6 public static double Atan2(double y, double x)
```

7 Summary

8 Returns the angle whose tangent is the quotient of two specified System.Double values.

9 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.

10

11 Return Value

12 A System.Double containing the value of an angle, θ , measured in radians, such that $-\pi \leq \theta \leq \pi$ and $\tan\theta = y/x$, where (x, y) is a point in the Cartesian plane.

14

15 If either one of x or y is a NaN, then that NaN is returned.

16

17 If both x and y are NaNs, then one of them is returned.

18

19 The following table specifies the return value if x or y or both are equal to
20 System.Double.NegativeInfinity Or System.Double.PositiveInfinity.

Condition	Return Value
y is equal to System.Double.PositiveInfinity, and x is equal to System.Double.PositiveInfinity.	System.Math.PI/4.
y is equal to System.Double.NegativeInfinity, and x is equal to System.Double.PositiveInfinity.	-System.Math.PI/4.
y is equal to System.Double.PositiveInfinity, and	3*

<code>x</code> is equal to <code>System.Double.NegativeInfinity</code> .	<code>System.Math.PI/4</code> .
<code>x</code> is equal to <code>System.Double.NegativeInfinity</code> , and <code>y</code> is equal to <code>System.Double.NegativeInfinity</code> .	<code>-3*</code> <code>System.Math.PI/4</code> .
<code>y</code> is equal to <code>System.Double.NegativeInfinity</code> , and <code>x</code> is not equal to <code>System.Double.PositiveInfinity</code> or <code>System.Double.NegativeInfinity</code> .	<code>-System.Math.PI/2</code> .
<code>y</code> is equal to <code>System.Double.PositiveInfinity</code> , and <code>x</code> is not equal to <code>System.Double.PositiveInfinity</code> or <code>System.Double.NegativeInfinity</code> .	<code>System.Math.PI/2</code> .
<code>x</code> is equal to <code>System.Double.PositiveInfinity</code> , and <code>y</code> is not equal to <code>System.Double.PositiveInfinity</code> or <code>System.Double.NegativeInfinity</code> .	<code>0</code> .
<code>x</code> is equal to <code>System.Double.NegativeInfinity</code> , and <code>y</code> ≥ 0 and not equal to <code>System.Double.PositiveInfinity</code> .	<code>System.Math.PI</code> .
<code>x</code> is equal to <code>System.Double.NegativeInfinity</code> , and <code>y</code> < 0 and not equal to <code>System.Double.NegativeInfinity</code> .	<code>-System.Math.PI</code> .

1

2 **Description**

3 The return value is the angle in the Cartesian plane formed by the x-axis, and a vector
4 starting from the origin, (0,0), and terminating at the point, (x,y).

5

6 [*Note:*

7

- For (x, y) in quadrant 1, $0 < \theta < \pi/2$.

8

- For (x, y) in quadrant 2, $\pi/2 < \theta < \pi$.

9

- For (x, y) in quadrant 3, $-\pi < \theta < -\pi/2$.

1 • For (x, y) in quadrant 4, $-\pi/2 < \theta < 0$.

2]

3 **Example**

4 The following example demonstrates using the `System.Math.Atan2` method.

5
6 [C#]

```
7 using System;
8
9 public class MathAtan2Example
10 {
11
12     public static void Main()
13     {
14
15         Double d1 = Math.Atan2(2,0);
16         Double d2 = Math.Atan2(0,0);
17         Console.WriteLine("Math.Atan2(2,0) returns {0}", d1);
18         Console.WriteLine("Math.Atan2(0,0) returns {0}", d2);
19
20     }
21
22 }
```

23 The output is

24
25 `Math.Atan2(2,0)` returns 1.5707963267949

26
27
28 `Math.Atan2(0,0)` returns 0

29

30

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

2 Method

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

7 Summary

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

9 Parameters

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

10

11 Return Value

12 A System.Int64 containing the product of the specified numbers.

13

1 Math.Ceiling(System.Double) Method

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

6 Summary

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

8 Parameters

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

9 Return Value

11 A System.Double containing the value of the smallest representable integer greater
12 than or equal to *a*. If *a* is equal to a NaN, System.Double.NegativeInfinity, or
13 System.Double.PositiveInfinity, that value is returned.

14 Example

15 The following example demonstrates using the System.Math.Ceiling method.

```
16 [C#]  
17  
18 using System;  
19  
20 public class MathCeilingExample  
21 {  
22  
23     public static void Main()  
24     {  
25  
26         Double d1 = Math.Ceiling(3.4);  
27         Double d2 = Math.Ceiling(-3.4);  
28         Console.WriteLine("Math.Ceiling(3.4) returns {0}", d1);  
29         Console.WriteLine("Math.Ceiling(-3.4) returns {0}", d2);  
30  
31     }  
32 }  
33 }
```

34 The output is

```
35  
36 Math.Ceiling(3.4) returns 4  
37
```

1
2 `Math.Ceiling(-3.4)` returns -3
3
4

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 angle.

8 Parameters

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

9 10 Return Value

11 A `System.Double` containing the value of the cosine of *d*. If *d* is a NaN, returns that
12 NaN; if *d* equals `System.Double.NegativeInfinity`, or
13 `System.Double.PositiveInfinity`, returns `System.Double.NaN`.

14 Description

15 [Note: Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18

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 represents an angle.

8 Parameters

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

9 Return Value

11 The hyperbolic cosine of *value*. If *value* is equal to `System.Double.NegativeInfinity`
12 or `System.Double.PositiveInfinity`, returns `System.Double.PositiveInfinity`. If
13 *value* is a NaN, returns that NaN.

14 Description

15 [*Note:* Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18

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

```
3 [ILAsm]  
4 .method public hidebysig static int32 DivRem(int32 a,int32 b, [out] int32  
5 &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 an output
10 parameter.

11 Parameters

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.

12 13 Return Value

14 A System.Int32 containing the quotient of the specified numbers.

15

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

```
3 [ILAsm]  
4 .method public hidebysig static int64 DivRem(int64 a,int64 b,[out] int64  
5 &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 an output
10 parameter.

11 Parameters

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.

12 13 Return Value

14 A System.Int64 containing the quotient of the specified numbers.

15

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 exponent.

8 Parameters

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

9 Return Value

11 A `System.Double` equal to the number e raised to the power of *d*. If *d* is a NaN or equals
12 `System.Double.PositiveInfinity`, returns that value. If *d* equals
13 `System.Double.NegativeInfinity`, returns 0.

14 Description

15 [*Note:* Use the `System.Math.Pow` method to calculate powers of other bases.
16
17 `System.Math.Exp` is the inverse of `System.Math.Log`.
18
19]

20

1 Math.Floor(System.Double) Method

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

6 Summary

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

8 Parameters

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

9 Return Value

11 A `System.Double` containing the value of the largest representable integer less than or
12 equal to *d*. If *d* is a NaN, or equals `System.Double.NegativeInfinity` or
13 `System.Double.PositiveInfinity`, that value is returned.

14 Description

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

16 Example

17 The following example demonstrates using the `System.Math.Floor` method.

```
18 [C#]  
19  
20 using System;  
21  
22 public class MathFloorExample  
23 {  
24  
25     public static void Main()  
26     {  
27  
28         Double d1 = Math.Floor(3.4);  
29         Double d2 = Math.Floor(-3.4);  
30         Console.WriteLine("Math.Floor(3.4) returns {0}", d1);  
31         Console.WriteLine("Math.Floor(-3.4) returns {0}", d2);  
32  
33     }  
34  
35 }
```

1 The output is
2
3 `Math.Floor(3.4)` returns 3
4
5
6 `Math.Floor(-3.4)` returns -4
7
8

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

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

8 Summary

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

11 Parameters

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

12

13 Return Value

14 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.
x	y is System.Double.PositiveInfinity or System.Double.NegativeInfinity and x is neither a NaN, nor System.Double.PositiveInfinity, nor System.Double.NegativeInfinity.

System.Double.NaN	$y = 0$ or x is System.Double.PositiveInfinity or System.Double.NegativeInfinity and neither is a NaN.
x	x is NaN, and y is not.
y	y is NaN, and x is not.
x or y	Both x and y are NaNs.

1

2 Description

3 This operation complies with the remainder operation defined in Section 5.3.1 of IEEE
 4 Std 754-2008; IEEE Standard for Floating-Point Arithmetic; Institute of Electrical and
 5 Electronics Engineers, Inc; 2008.

6
 7 *[Note: For more information regarding the use of +0 and -0, see Section 3.3 of IEEE Std*
 8 *754-2008; IEEE Standard for Floating-Point Arithmetic; Institute of Electrical and*
 9 *Electronics Engineers, Inc; 2008.]*

10

11

12 Example

13 The following example demonstrates using the System.Math.IEEEERemainder method.

14

15 [C#]

16 using System;

17

18 public class MathIEEEERemainderExample

19 {

20

21 public static void Main()

22 {

23

24 Double d1 = Math.IEEEERemainder(3.54,0);

25 Double d2 = Math.IEEEERemainder(9.99,-3.33);

26 Double d3 = Math.IEEEERemainder(-9.99,3.33);

27 Double d4 = Math.IEEEERemainder(9.5,1.5);

28 Console.WriteLine("Math.IEEEERemainder(3.54,0) returns {0}", d1);

29 Console.WriteLine("Math.IEEEERemainder(9.99,-3.33) returns {0}", d2);

30 Console.WriteLine("Math.IEEEERemainder(-9.99,3.33) returns {0}", d3);

31 Console.WriteLine("Math.IEEEERemainder(9.5,1.5) returns {0}", d4);

32

33 }

34

35 }

1 The output is
2
3 `Math.IEEEERemainder(3.54,0)` returns NaN
4
5
6 `Math.IEEEERemainder(9.99,-3.33)` returns 0
7
8
9 `Math.IEEEERemainder(-9.99,3.33)` returns 0
10
11
12 `Math.IEEEERemainder(9.5,1.5)` returns 0.5
13
14

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

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

9

10 Return Value

11 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	System.Double.NaN.
<i>d</i> is a NaN	<i>d</i> .
<i>d</i> is equal to System.Double.PositiveInfinity.	System.Double.PositiveInfinity.

12

13 Description

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

15

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

2 Method

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

7 Summary

8 Returns the logarithm of the specified `System.Double` in the specified base.

9 Parameters

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

10

11 Return Value

12 Returns a `System.Double` whose value is as follows:

Condition	Returns
$a > 0, newBase > 0, \text{ but } newBase \neq 1$	$\log_{newBase} a$
$a < 0$	<code>System.Double.NaN</code>
$newBase < 0$	<code>System.Double.NaN</code>
$newBase == 0, a \neq 1$	<code>System.Double.NaN</code>
$newBase == 0, a == 1$	Zero
$0 < newBase < 1, a == 0$	<code>System.Double.PositiveInfinity</code>
$0 < newBase < 1, a == +infinity$	<code>System.Double.NegativeInfinity</code>
$newBase == 1$	<code>System.Double.NaN</code>

$newBase > 1, a == 0$	System.Double.NegativeInfinity
$newBase > 1, a == +infinity$	System.Double.PositiveInfinity
$newBase == +infinity, a != 1$	System.Double.NaN
$newBase == +infinity, a == 1$	Zero
a is a NaN and $newBase$ is not a NaN	a
$newBase$ is a NaN and a is not a NaN	$newBase$
Both a and $newBase$ are NaNs	a or $newBase$

1

2

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

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

9

10 Return Value

11 Returns a `System.Double` as indicated by the following table.

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

12

13

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

2 Method

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

7 Summary

8 Returns the greater of two specified `System.SByte` values.

9 Type Attributes:

- 10 • `CLSCompliantAttribute(false)`

11 Parameters

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

12

13 Return Value

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

16 Description

17 This method is not CLS-compliant. For a CLS-compliant alternative, use
18 `System.Math.Max(System.Int16, System.Int16)`.

19

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

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

7 Summary

8 Returns the greater of two specified System.Byte values.

9 Parameters

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.

10

11 Return Value

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

14

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

2 Method

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

7 Summary

8 Returns the greater of two specified System.Int16 values.

9 Parameters

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.

10

11 Return Value

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

14

1 `Math.Max(System.UInt16, System.UInt16)`

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int16 Max(unsigned int16 val1,  
5 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

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

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 `System.Math.Max(System.Int32, System.Int32)`.

20

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

2 Method

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

7 Summary

8 Returns the greater of two specified `System.Int32` values.

9 Parameters

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

10

11 Return Value

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

14

1 `Math.Max(System.UInt32, System.UInt32)`

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int32 Max(unsigned int32 val1,  
5 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

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

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 `System.Math.Max(System.Int64, System.Int64)`.

20

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

2 **Method**

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

7 **Summary**

8 Returns the greater of two specified `System.Int64` values.

9 **Parameters**

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

10

11 **Return Value**

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

14

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int64 Max(unsigned int64 val1,  
5 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

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.

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 System.Math.Max(System.Decimal, System.Decimal).

20

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

2 Method

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

7 Summary

8 Returns the greater of two specified `System.Single` values.

9 Parameters

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

10

11 Return Value

12 A `System.Single` equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the
13 return value is equal to *val2*. If one of *val1* or *val2* is a NaN, then that NaN is returned;
14 if both are NaN, then one of them is returned.

15

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

2 Method

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

7 Summary

8 Returns the greater of two specified System.Double values.

9 Parameters

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.

10

11 Return Value

12 A System.Double equal to *val1* if *val1* is greater than or equal to *val2*; otherwise, the
13 return value is equal to *val2*. If one of *val1* or *val2* is a NaN, then that NaN is returned;
14 if both are NaN, then one of them is returned.

15

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

2 Method

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

7 Summary

8 Returns the greater of two specified System.Decimal values.

9 Parameters

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.

10

11 Return Value

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

14

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

2 Method

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

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.

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 System.Math.Min(System.Int32, System.Int32).

20

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.Int16` values.

9 Parameters

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

10

11 Return Value

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

14

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

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

7 Summary

8 Returns the lesser of two specified `System.Byte` values.

9 Parameters

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

10

11 Return Value

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

14

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.SByte` values.

9 Type Attributes:

- 10 • `CLSCompliantAttribute(false)`

11 Parameters

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

12

13 Return Value

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

16 Description

17 This method is not CLS-compliant. For a CLS-compliant alternative, use
18 `System.Math.Min(System.Int16, System.Int16)`.

19

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int64 Min(unsigned int64 val1,  
5 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

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.

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 System.Math.Min(System.Decimal, System.Decimal).

20

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.Single` values.

9 Parameters

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

10

11 Return Value

12 A `System.Single` equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return
13 value is equal to *val2*. If one of *val1* or *val2* is a NaN, then that NaN is returned; if both
14 are NaN, then one of them is returned.

15

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.Int64` values.

9 Parameters

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

10

11 Return Value

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

14

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

2 Method

```
3 [ILAsm]  
4 .method public hidebysig static unsigned int32 Min(unsigned int32 val1,  
5 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

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

13

14 Return Value

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

17 Description

18 This method is not CLS-compliant. For a CLS-compliant alternative, use
19 `System.Math.Min(System.Int64, System.Int64)`.

20

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.Int32` values.

9 Parameters

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

10

11 Return Value

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

14

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

2 Method

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

7 Summary

8 Returns the lesser of two specified `System.Double` values.

9 Parameters

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

10

11 Return Value

12 A `System.Double` equal to *val1* if *val1* is less than or equal to *val2*; otherwise, the return
13 value is equal to *val2*. If one of *val1* or *val2* is a NaN, then that NaN is returned; if both
14 are NaN, then one of them is returned.

15

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

2 Method

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

7 Summary

8 Returns the lesser of two specified System.Decimal values.

9 Parameters

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.

10

11 Return Value

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

14

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

2 Method

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

7 Summary

8 Returns the specified System.Double raised to the specified power.

9 Parameters

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

10

11 Return Value

12 A System.Double equal to x raised to the power y. The following table specifies the
13 results if x or y is a NaN or equal to System.Double.NegativeInfinity or
14 System.Double.PositiveInfinity.

Parameter Values	Returns
y == 0	1
x == +0 and y an odd integer < 0	System.Double.PositiveInfinity
x == -0 and y an odd integer < 0	System.Double.NegativeInfinity
x == ±0 and y is -infinity	System.Double.PositiveInfinity
x == ±0 and y is +infinity	+0
x == ±0 and y < 0 not an odd integer	System.Double.PositiveInfinity
x == ±0 and y > 0 an odd integer	±0
x == ±0 and y > 0 not an odd integer	+0

$x == -1, y == -\text{infinity or } +\text{infinity}$	1
$x == 1$	1
$x == -\text{infinity}, y < 0$	0
$x == -\text{infinity}, y$ is a positive odd integer	System.Double.NegativeInfinity
$x == -\text{infinity}, y$ is neither 0 nor a positive odd integer	System.Double.PositiveInfinity
$x < 0, (-1 < y < 0)$ or $(0 < y < 1)$	System.Double.NaN
$x < -1, y == -\text{infinity}$	0
$x < -1, y == +\text{infinity}$	System.Double.PositiveInfinity
$(-1 < x \leq 0), y == -\text{infinity}$	System.Double.PositiveInfinity
$(-1 < x \leq 0), y == +\text{infinity}$	0
$(0 < x < 1), y == -\text{infinity}$	System.Double.PositiveInfinity
$(0 < x < 1), y == +\text{infinity}$	0
$x > 1, y == -\text{infinity}$	0
$x > 1, y == +\text{infinity}$	System.Double.PositiveInfinity
$x == +\text{infinity}, y < 0$	0
$x == +\text{infinity}, y > 0$	System.Double.PositiveInfinity
x is a NaN and y is not 0	x
y is a NaN and x is not 1	y
Both x and y are NaNs	One of x or y

1

2

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 <code>System.Decimal</code> to be rounded.

9

10 Return Value

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

14 Description

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

16 Example

17 The following example demonstrates using the `System.Math.Round(System.Decimal)`
18 method.

```
19 [C#]  
20  
21 using System;  
22  
23 public class MathRoundExample  
24 {  
25  
26     public static void Main()  
27     {  
28  
29         Double d1 = Math.Round(4.4);  
30         Double d2 = Math.Round(4.5);  
31         Double d3 = Math.Round(4.6);  
32         Console.WriteLine("Math.Round(4.4) returns {0}", d1);  
33         Console.WriteLine("Math.Round(4.5) returns {0}", d2);  
34         Console.WriteLine("Math.Round(4.6) returns {0}", d3);  
35
```

```
1     }
2
3   }
4   The output is
5
6   Math.Round(4.4) returns 4
7
8
9   Math.Round(4.5) returns 4
10
11
12  Math.Round(4.6) returns 5
13
14
```

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

2 Method

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

7 Summary

8 Returns the number nearest the specified `System.Double` within the specified precision.

9 Parameters

Parameter	Description
<i>value</i>	A <code>System.Double</code> to be rounded.
<i>digits</i>	A <code>System.Int32</code> 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.

10

11 Return Value

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

19 Description

20 The behavior of this method follows IEEE Standard 754-2008, section 4.3.

21 Exceptions

Exception	Condition
<code>System.ArgumentOutOfRangeException</code>	<i>digits</i> < 0 -or-

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35

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.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

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

9 Return Value

11 A `System.Double` containing the value of the representable integer nearest *a*. If *a* is
12 exactly halfway between two representable integers, one of which is even and the other
13 odd, then the even integer is returned. If *a* is a NaN, then the result is that NaN.

14 Description

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

16 Example

17 The following example demonstrates using the `System.Math.Round(System.Double)`
18 method.

```
19 [C#]  
20  
21 using System;  
22  
23 public class MathRoundExample  
24 {  
25  
26     public static void Main()  
27     {  
28  
29         Double d1 = Math.Round(4.4);  
30         Double d2 = Math.Round(4.5);  
31         Double d3 = Math.Round(4.6);  
32         Console.WriteLine("Math.Round(4.4) returns {0}", d1);  
33         Console.WriteLine("Math.Round(4.5) returns {0}", d2);  
34         Console.WriteLine("Math.Round(4.6) returns {0}", d3);  
35
```

```
1     }
2
3   }
4   The output is
5
6   Math.Round(4.4) returns 4
7
8
9   Math.Round(4.5) returns 4
10
11
12  Math.Round(4.6) returns 5
13
14
```

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

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

9 10 Return Value

11 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.

12
13

1 Math.Sign(System.Double) Method

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

6 Summary

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

8 Parameters

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

9

10 Return Value

11 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.

12

13 Exceptions

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

14

15 Example

16 The following example demonstrates using the System.Math.Sign(System.Double)
17 method.

18

19 [C#]

```
1 using System;
2
3 public class MathSignExample
4 {
5
6     public static void Main()
7     {
8
9         Double d1 = Math.Sign(4.4);
10        Double d2 = Math.Sign(0.0);
11        Double d3 = Math.Sign(-4.5);
12        Console.WriteLine("Math.Sign(4.4) returns {0}", d1);
13        Console.WriteLine("Math.Sign(0.0) returns {0}", d2);
14        Console.WriteLine("Math.Sign(-4.5) returns {0}", d3);
15
16    }
17
18 }
19 The output is
20
21 Math.Sign(4.4) returns 1
22
23
24 Math.Sign(0.0) returns 0
25
26
27 Math.Sign(-4.5) returns -1
28
29
```

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

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

9 Return Value

11 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.

12 Exceptions

Exception	Condition
<code>System.ArithmeticException</code>	<i>value</i> is a <code>System.Single.NaN</code> .

14
15

1 Math.Sign(System.Int64) Method

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

6 Summary

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

8 Parameters

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

9

10 Return Value

11 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.

12

13

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

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

9

10 Return Value

11 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.

12

13

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

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

9

10 Return Value

11 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.

12

13

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

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

11

12 Return Value

13 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.

14

15 Description

16 This method is not CLS-compliant. For a CLS-compliant alternative, use
17 `System.Math.Sign(System.Int16)`.

18

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 angle.

8 Parameters

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

9 Return Value

11 A System.Double containing the value of the sine of *a*. If *a* is a NaN, returns that NaN; if
12 *a* is equal to System.Double.NegativeInfinity or System.Double.PositiveInfinity,
13 returns System.Double.NaN.

14 Description

15 [Note: Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18 Example

19 The following example demonstrates using the System.Math.Sin method.

```
20 [C#]  
21  
22 using System;  
23  
24 public class MathSinExample  
25 {  
26  
27     public static void Main()  
28     {  
29  
30         Double d1 = Math.Sin(0);  
31         Double d2 = Math.Sin(Math.PI/2.0);  
32         Console.WriteLine("Math.Sin(0) returns {0}", d1);  
33         Console.WriteLine("Math.Sin(Math.PI/2.0) returns {0}", d2);  
34  
35     }  
}
```

```
1
2 }
3 The output is
4
5 Math.Sin(0) returns 0
6
7
8 Math.Sin(Math.PI/2.0) returns 1
9
10
```

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 represents an angle.

8 Parameters

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

9 Return Value

11 A `System.Double` containing the value of the hyperbolic sine of *value*. If *value* is equal
12 to `System.Double.NegativeInfinity` or `System.Double.PositiveInfinity`, or is a
13 NaN, returns *value*.

14 Description

15 [Note: Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18 Example

19 The following example demonstrates using the `System.Math.Sinh` method.

```
20 [C#]  
21  
22 using System;  
23  
24 public class MathSinhExample  
25 {  
26  
27     public static void Main()  
28     {  
29  
30         Double d1 = Math.Sinh(0);  
31         Double d2 = Math.Sinh(Math.PI);  
32         Console.WriteLine("Math.Sinh(0) returns {0}", d1);  
33         Console.WriteLine("Math.Sinh(Math.PI) returns {0}", d2);  
34  
35     }  
}
```

```
1
2 }
3 The output is
4
5 Math.Sinh(0) returns 0
6
7
8 Math.Sinh(Math.PI) returns 11.5487393572577
9
10
```

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

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

9 Return Value

11 A System.Double whose value is indicated as follows:

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

12 Example

13 Example

1 The following example demonstrates using the System.Math.Sqrt method.

2

3 [C#]

4 using System;

5

6 public class MathSqrtExample

7 {

8

9 public static void Main()

10 {

11

12 Double d1 = Math.Sqrt(16.0);

13 Double d2 = Math.Sqrt(0.0);

14 Double d3 = Math.Sqrt(-10.0);

15 Console.WriteLine("Math.Sqrt(16.0) returns {0}", d1);

16 Console.WriteLine("Math.Sqrt(0.0) returns {0}", d2);

17 Console.WriteLine("Math.Sqrt(-10.0) returns {0}", d3);

18

19 }

20

21 }

22 The output is

23

24 Math.Sqrt(16.0) returns 4

25

26

27 Math.Sqrt(0.0) returns 0

28

29

30 Math.Sqrt(-10.0) returns NaN

31

32

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 an angle.

8 Parameters

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

9 Return Value

11 A System.Double containing the value of the tangent of *a*. If *a* is a NaN, returns that
12 NaN; if *a* is equal to System.Double.NegativeInfinity or
13 System.Double.PositiveInfinity, returns System.Double.NaN.

14 Description

15 [*Note:* Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18 Example

19 The following example demonstrates using the System.Math.Tan method.

```
20 [C#]  
21  
22 using System;  
23  
24 public class MathTanExample  
25 {  
26  
27     public static void Main()  
28     {  
29  
30         Double d1 = Math.Tan(0);  
31         Double d2 = Math.Tan(Math.PI/2.0);  
32         Console.WriteLine("Math.Tan(0) returns {0}", d1);  
33         Console.WriteLine("Math.Tan(Math.PI/2.0) returns {0}", d2);  
34  
35     }  
}
```

```
1
2 }
3 The output is
4
5 Math.Tan(0) returns 0
6
7
8 Math.Tan(Math.PI/2.0) returns 1.63317787283838E+16
9
10
```

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 represents an angle.

8 Parameters

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

9 Return Value

11 A `System.Double` containing the value of the hyperbolic tangent of *value*. If *value* is
12 equal to `System.Double.NegativeInfinity`, returns -1. If *value* is equal to
13 `System.Double.PositiveInfinity`, returns 1. If *value* is a NaN, returns that NaN.

14 Description

15 [*Note:* Multiply by $\pi/180$ to convert degrees to radians.]
16
17

18 Example

19 The following example demonstrates using the `System.Math.Tanh` method.

```
20 [C#]  
21  
22 using System;  
23  
24 public class MathTanhExample  
25 {  
26  
27     public static void Main()  
28     {  
29  
30         Double d1 = Math.Tanh(0);  
31         Double d2 = Math.Tanh(Math.PI);  
32         Console.WriteLine("Math.Tanh(0) returns {0}", d1);  
33         Console.WriteLine("Math.Tanh(Math.PI) returns {0}", d2);  
34  
35     }  
}
```

```
1
2 }
3 The output is
4
5 Math.Tanh(0) returns 0
6
7
8 Math.Tanh(Math.PI) returns 0.99627207622075
9
10
```