Rational Struct

名前空間: WS.Theia.ExtremelyPrecise
アセンブリ: ExtremelyPrecise.dll
任意の大きさを持つ有理数を、誤差を発生させずに表現します。

```
[System.Serializable]
public partial struct Rational:IComparable,IComparable<Rational>,
IEquatable<Rational>,Iformattable
```

継承：Object → ValueType → Rational
属性：SerializableAttribute
実装：IComparable, IComparable<Rational>, IEquatable<Rational>, IFormattable

注釈
Rational 型は任意の有理数を事実上（厳密な上限、下限は Rational の表現可能な値の範囲とメモリ消費を参照してください)、上限や下限なく表現できる構造体です。また、事実上、打切り誤差、情報落ち、桁落ちが起こらない内部表現をしており、循環小数の様な通常の浮動小数点型では正しく表現できない値を正しく表現できます。.NET Framework の浮動小数点型（Float、Double、Decimal)、と異なり MinValue、MaxValue、及び、Epsilon プロパティがありません。

> ⚠注意
> Rational 型は変更できません（Rational 型の可変性を参照してください）。上限および下限が無く、又、誤差を引き起こさない内部表現を使用している都合で、単純な足し算であっても大きな内部情報をコピーする可能性があります。コピーの時点でメモリ確保量が急増して OutOfMemoryException が スローされる可能性に注意してください。

Rational オブジェクトをインタンス化するには

Rational 型はいくつかの方法でインスタンス化することができます。

New キーワードを使用して Rational 型のコンストラクターに任意の整数型、または、浮動小数点型を使用することで初期化できます。

```
Rational rationalFromDouble = new Rational (179032.6541);
Console.WriteLine(rationalFromDouble);
Rational rationalFromInt64 = new Rational(934157136952);
Console.WriteLine(rationalFromInt64);
// The example displays the following output:
//     179032.6541
//     934157136952
```

Rational 型は変数割り当てと同様に宣言することができます。Rational 変数と割り当ての場合、割り当てたい値が格納された整数型、又は、浮動小数点型を代入します。 次の例では、割り当てを使用して、Int64 の値から Rational をインスタンス化します。

```
long longValue = 6315489358112;
Rational assignedFromLong = longValue;
Console.WriteLine(assignedFromLong);
// The example displays the following output:
//     6315489358112
```

10 進数または浮動小数点値を割り当てる場合も、整数型同様に割り当てる時に明示的なキャストは不要です。

```
Rational assignedFromDouble = 179032.6541;
Console.WriteLine(assignedFromDouble);
Rational assignedFromDecimal = 64312.65m;
Console.WriteLine(assignedFromDecimal);
// The example displays the following output:
//     179032.6541
//     64312.65
```

Byte 配列から初期化するメソッドを使用すると既存の整数型、及び、浮動小数点型の上限を超えた値で初期化することができます。new キーワードを使用して Rational 型のコンストラクターに、符号を示す bool 値、分母を示す Byte 配列、分子を示す Byte 配列を提供します。

```
bool sign=false;
byte[] numerator = { 10, 9, 8, 7, 6, 5, 4, 3, 2, 1, 0};
byte[] denominator = {1};
Rational rational = new Rational (sign,numerator,denominator);
Console.WriteLine("The value of Rational is {0}.", rational);
// The example displays the following output:
//      The value of Rational is 4759477275222530853130.
```

文字列形式を Rational に変換する場合には Parse、又は、TryParse メソッドを使用します。

```
string positiveString = "913896812479936712554321120000000";
string negativeString = "-90315837410896312071002088037140000";
Rational posRational = 0;
Rational negRational = 0;
bool status=false;

try {
    posRational = Rational.Parse(positiveString);
    Console.WriteLine(posRational);
}
catch (FormatException)
{
    Console.WriteLine("Unable to convert the string '{0}' to a Rational value.",
positiveString);
}

(status, negRational) = Rational.TryParse(negativeString);
if (status)
   Console.WriteLine(negRational);
else
    Console.WriteLine("Unable to convert the string '{0}' to a Rational value.",
negativeString);

// The example displays the following output:
//     913896812479936712554321120000000
//     -90315837410896312071002088037140000
```

任意の整数、又は、浮動小数点に対し Rational 型の演算子やメソッドを使用する事で、Rational 型を初期化できます。次の例では UInt64.MaxValue を 3 乗しています。

```
Rational number = Math.Pow(UInt64.MaxValue, 3);
Console.WriteLine(number);
// The example displays the following output:
//       6277101735386680762814942322444851025767571854389858533375
```

初期化されてしていない Rational 型は Zero と等価になります。

Rational の値を操作するには

Rational インスタンスには他の整数型、または、浮動小数点型と同様に演算子を使用することができます。Rational 型は加算、減算、除算、乗算、減算、否定、単項マイナス演算子などの基本的な算術演算をオーバー ロードしており、整数型、または、浮動小数点型と同様に演算子を使用できます。また、比較演算子も同様に使用することができます。Rational 型は Add、Or、Xor、Left Shift、Right Shift、Ones Complement といったビット演算をサポートしています。Rational 型は、カスタム演算子をサポートしない言語に対して、算術演算を実行するための同等のメソッドも用意しています。 以下の Add、 Divide、 Multiply、 Negate、 Subtract、およびその他のいくつかのメソッドが該当します。
Rational 型で定義している多くのメンバーは浮動小数点型のメンバーに直接対応します。Rational 型では浮動小数点型にはないメンバーを定義しています。

- Sign：Rational の符号を返す。 （Math クラスに定義しています）
- Abs：Rational の絶対値を返す。（Math クラスに定義しています）
- DivRem：商と除算の剰余の両方が返す。 （Math クラスに定義しています）
- GreatestCommonDivisor： 2 つの Rational 値の最大公約数を返す。

ただし、Rational 型と同様の任意精度型である BigInteger 型と異なり、浮動小数点型では System.Math クラスで定義しているメンバーは WS.Theia.ExtremelyPrecise.Math クラスに定義しています。

Rational 型の可変性

次の例ではインスタンス化し Rational オブジェクトの値をインクリメントしています。

```
Rational number = Rational.Multiply(Int64.MaxValue, 3);
number++;
Console.WriteLine(number);
```

この例では、既存のオブジェクトの値を変更しているように見えますが、これは正しくありません。 Rational オブジェクトは値を変更ができません。 つまり内部的には、新しい Rational オブジェクトを生成し、インクリメント前の値より 1 つ大きな値を割り当てています。そして新しいオブジェクトを、呼び出し元に返しています。

> ⚠注意
> .NET Framework の他の数値型も変更できません。ただし、Rational 型は上限および下限が無く、又、誤差を引き起こさない内部表現を使用しているため、内部値が大きくなっている可能性があり、パフォーマンスに多大な影響が出る場合があります。例えばインクリメントをする場合、必要メモリの最大値は約 128GiB になります。

このプロセスは呼び出し元には透過的に行われ、パフォーマンスの低下を発生されます。特に、ループ内など操作を繰り返して行う状況で、Rational オブジェクトに大きな値が設定しているとと大幅なパフォーマンス低下を引き起こします。次の例では、SomeOperationSucceeds メソッドが成功する度に Rational オブジェクトの生成、内部データのコピー、Rational オブジェクトの破棄を、最大で 100 万回行う事となり大きなパフォーマンスの低下を引き起こします。

```
Rational number = Int64.MaxValue ^ 5;
int repetitions = 1000000;
// Perform some repetitive operation 1 million times.
for (int ctr = 0; ctr <= repetitions; ctr++)
{
    // Perform some operation. If it fails, exit the loop.
    if (! SomeOperationSucceeds()) break;
    // The following code executes if the operation succeeds.
    number++;
}
```

このような場合、整数型、又は、浮動小数点型の中間変数を使い、ループ終了後に Rational 変数に割り当てる事によりパフォーマンスを向上できます。

```
Rational number = Int64.MaxValue ^ 5;
int repetitions = 1000000;
int actualRepetitions = 0;
// Perform some repetitive operation 1 million times.
for (int ctr = 0; ctr <= repetitions; ctr++)
{
    // Perform some operation. If it fails, exit the loop.
    if (! SomeOperationSucceeds()) break;
    // The following code executes if the operation succeeds.
    actualRepetitions++;
}
number += actualRepetitions;
```

Rational の表現可能な値の範囲とメモリ消費

Rational 型では下図の内部表現を行います。

<table>
<tr><td rowspan="2">符号フラグ部</td><td>分子部（最大で 512Gbit）</td><td rowspan="2">無限大フラグ部</td></tr>
<tr><td>分母部（最大で 512Gbit）</td></tr>
</table>

Rational 型では下表の範囲と精度で数値を表現できます。

表現可能な値の範囲	±∞、±1/D～N/1（n/d で表現可能な範囲である事）、±0、NaN ※1
表現可能な精度	最大精度：1/D、最小精度：1/1
使用メモリ	インスタンスあたり 64Bit～512Gbit×2+8bit×2※2

※1　整数の分子を n、整数の分母を d、分子部の 512Gbit で表現可能な整数値の上限を N、分母部の 512Gbit で表現可能な整数値の上限を D、として表記。
※2　値を保持する為に必要なメモリであり、演算時はワーキングメモリを含めると瞬間的に必要領域が 1TB 程度に達する可能性がある。

上記の特性から小さな値であるほど精度が高く、大きな値であるほど精度が低くなります。ただし、地球の直径（mm）×π（2019 年 3 月 14 日に達成された約 31 兆桁とする）の演算に十二分な演算精度と提供できます。通常の演算では精度による問題は起こりません。

バイト配列と 16 進数の文字列の操作

Byte 配列を Rational 値に、又は、Rational 値を Byte 配列に変換するには順序を考慮する必要があります。Rational 構造体は、リトルエディアンでデータを格納しており、相互に変換する Byte 配列もリトルエディアンでなければなりません。次に示す例では、Rational の TobyteArray メソッドの結果を、Rational(Boolean,Byte[],Byte[])コンストラクターに渡しています。

```
Rational number = Math.Pow(Int64.MaxValue, 2);
Console.WriteLine(number);

// Write the BigInteger value to a byte array.
var bytes = number.ToByteArray();

// Display the byte array.
foreach (byte byteValue in bytes.Numerator)
    Console.Write("0x{0:X2} ", byteValue);
Console.WriteLine();

// Restore the Rational value from a Byte array.
Rational newNumber = new Rational (bytes.Sign ,bytes.Numerator,
bytes.Denominator);
Console.WriteLine(newNumber);
// The example displays the following output:
//       85070591730234615847396907784232501249
//       0x01 0x00 0x00 0x00 0x00 0x00 0x00 0x00 0xFF 0xFF 0xFF 0xFF 0xFF
0xFF 0xFF 0x3F
//
//       85070591730234615847396907784232501249
```

Rational オブジェクトをインスタンス化するに当たり整数値のバイト配列を利用することができます。BitConverter.GetBytes メソッドの結果を分子、分母を 1 として、Rational(Boolean,Byte[],Byte[])コンストラクターを呼び出す事で等価な値にすることができます。次の例では、Int16 の値を示す Byte 配列から Rational オブジェクトを初期化しています。

```
short originalValue = 30000;
Console.WriteLine(originalValue);

// Convert the Int16 value to a byte array.
byte[] bytes = BitConverter.GetBytes(originalValue);

// Display the byte array.
foreach (byte byteValue in bytes)
    Console.Write("0x{0} ", byteValue.ToString("X2"));
Console.WriteLine();

// Pass byte array to the Rational constructor.
Rational number = new Rational (false,bytes,new byte[]{ 1 });
Console.WriteLine(number);
// The example displays the following output:
//       30000
//       0x30 0x75
//       30000
```

Rational 型では BigInteger 型と異なり符号フラグとして別に持っており、2 の補数表現を使って定義することができません。

コンストラクター

Rational(Boolean)	Boolean 値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Boolean,Byte[],Byte[])	バイト配列の値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Decimal)	Decimal 値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Double)	倍精度浮動小数点値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Int32)	32 ビット符号付き整数値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Int64)	64 ビット符号付き整数値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(Single)	単精度浮動小数点値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(UInt32)	32 ビット符号なし整数値を使用して、Rational 構造体の新しいインスタンスを初期化します。
Rational(UInt64)	64 ビット符号なし整数値を使用して、Rational 構造体の新しいインスタンスを初期化します。

プロパティ

Accuracy	演算精度。無理数を演算する際の打切り桁数、丸目の既定値として使用します。
IsEven	現在の Rational オブジェクトの値が偶数かどうかを示します。
IsOne	現在の Rational オブジェクトの値が One かどうかを示します。
IsPowerOfTwo	現在の Rational オブジェクトの値が 2 の累乗かどうかを示します。
IsZero	現在の Rational オブジェクトの値が Zero かどうかを示します。
MinusOne	負の 1 (-1) を表す値を取得します。
One	正の 1 (1) を表す値を取得します。
Zero	0 (ゼロ) を表す値を取得します。
NaN	非数(NaN) の値を表します。 このフィールドは定数です。
NegativeInfinity	負の無限大を表します。 このフィールドは定数です。
PositiveInfinity	正の無限大を表します。 このフィールドは定数です。

メソッド

Add(Rational,Rational)	2 つの Rational 値を加算し、その結果を返します。
BitwiseAnd(Rational,Rational)	2 つの BigInteger 値に対し、ビットごとの And 演算を実行します。
BitwiseOr(Rational, Rational)	2 つの Rational 値に対し、ビットごとの Or 演算を実行します。
Compare(Rational,Rational)	2 つの Rational 値を比較し、1 番目の値が 2 番目の値よりも小さいか、同じか、または大きいかを示す整数を返します。
Clone()	Rational 値を複製します。
CompareTo(decimal)	このインスタンスと 10 進数を比較し、このインスタンスの値が 10 進数の値よりも小さいか、同じか、または大きいかを示す整数を返します。
CompareTo(double)	このインスタンスと 倍精度浮動小数点数を比較し、このインスタンスの値が 倍精度浮動小数点数の値よりも小さいか、同じか、または大きいかを示す整数を返します。
CompareTo(long)	このインスタンスと符号付き 64 ビット整数を比較し、このインスタンスの値が符号付き 64 ビット整数の値よりも小さいか、同じか、または大きいかを示す整数を返します。
CompareTo(object)	このインスタンスと指定したオブジェクトを比較し、このインスタンスの値が指定したオブジェクトの値よりも小さいか、同じか、または大きいかを示す整数を返します。
CompareTo(Rational)	このインスタンスと Rational を比較し、このインスタンスの値が Rational の値よりも小さいか、同じか、または大きいかを示す整数を返します。
CompareTo(ulong)	このインスタンスと符号なし 64 ビット整数を比較し、このインスタンスの値が符号なし 64 ビット整数の値よりも小さいか、同じか、または大きいかを示す整数を返します。
Decrement(Rational)	Rational 値を 1 だけデクリメントします。
Divide(Rational,Rational)	一方の Rational 値をもう一方の値で除算し、その結果を返します。
Equals(decimal)	現在のインスタンスの値と 10 進数の値が等しいかどうかを示す値を返します。

Equals(double)	現在のインスタンスの値と倍精度浮動小数点数の値が等しいかどうかを示す値を返します。
Equals(long)	現在のインスタンスの値と符号付き 64 ビット整数の値が等しいかどうかを示す値を返します。
Equals(object)	現在のインスタンスの値と指定されたオブジェクトの値が等しいかどうかを示す値を返します。
Equals(Rational)	現在のインスタンスの値と Rational の値が等しいかどうかを示す値を返します。
Equals(ulong)	現在のインスタンスの値と符号無し 64 ビット整数の値が等しいかどうかを示す値を返します。
Equals(Rational,Rational)	2 つの Rational オブジェクトの値が等しいかどうかを示す値を返します。
FromOACurrency(long cy)	OLE オートメーション通貨値を格納している指定した 64 ビット符号付き整数を、それと等価の Rational 値に変換します。
GetHashCode()	現在の Rational オブジェクトのハッシュ コードを返します。
GetTypeCode()	TypeCode 値型の Object を返します。
GreatestCommonDivisor (Rational,Rational)	2 つの Rational 値の最大公約数を求めます。
Increment(Rational)	Rational 値を 1 だけインクリメントします。
IsInfinity(Rational)	指定した数値が負または正の無限大と評価されるかどうかを示す値を返します。
IsNaN(Rational)	指定した値が非数値（NaN）かどうかを示す値を返します。
IsNegativeInfinity(Rational)	指定した数値が負の無限大と評価されるかどうかを示す値を返します。
IsPositiveInfinity(Rational)	指定した数値が正の無限大と評価されるかどうかを示す値を返します。
LeftShift(Rational,int)	指定されたビット数だけ Rational 値を左にシフトします。
Mod(Rational,Rational)	指定された 2 つの Rational 値の除算の結果生じた剰余を返します。
ModPow(Rational,Rational,Rational)	ある数値を別の数値で累乗し、それをさらに別の数値で割った結果生じた剰余を求めます。
Multiply(Rational,Rational)	2 つの Rational 値の積を返します。

Negate(Rational)	指定された Rational 値を否定（負数化）します。
OnesComplement(Rational)	Rational 値のビットごとの 1 の補数を返します。
Parse(String)	数値の文字列形式を、それと等価の Rational に変換します。
Parse(String,NumberStyles)	指定のスタイルで表現された数値の文字列形式を、それと等価な Rational に変換します。
Parse(String,IFormatProvider)	指定されたカルチャ固有の書式で表現された文字列形式の数値を、それと等価の Rational に変換します。
Parse(String,NumberStyles,IFormatProvider)	指定したスタイルおよびカルチャ固有の書式の数値の文字列形式を、それと等価の BigInteger に変換します。
Plus(Rational)	Rational オペランドの値を返します。 オペランドの符号は変更されません。
RightShift(Rational,int)	指定されたビット数だけ Rational 値を右にシフトします。
Subtract(Rational,Rational)	ある Rational 値を別の値から減算し、その結果を返します。
ToByte(Rational)	指定した Rational の値を、等価の 8 ビット符号なし整数に変換します。
ToByteArray()	Rational 値をバイト配列に変換します。
ToDouble(Rational)	指定した Rational の値を、それと等価の倍精度浮動小数点数に変換します。
ToInt16(Rational)	指定した Rational の値を、等価の 16 ビット符号付き整数に変換します。
ToInt32(Rational)	指定した Rational の値を、等価の 32 ビット符号付き整数に変換します。
ToInt64(Rational)	指定した Rational の値を、等価の 64 ビット符号付き整数に変換します。
ToOACurrency(Rational)	指定した Rational 値を、64 ビット符号付き整数に格納されるそれと等価の OLE オートメーション通貨値に変換します。
ToSByte(Rational)	指定した Rational の値を、等価の 8 ビット符号付き整数に変換します。
ToSingle(Rational)	指定した Rational の値を、それと等価の単精度浮動小数点数に変換します。

ToString()	現在の BigInteger オブジェクトの数値を等価の文字列形式に変換します。
ToString(IFormatProvider)	指定されたカルチャ固有の書式情報を使用して、現在の Rational オブジェクトの数値をそれと等価の文字列形式に変換します。
ToString(String)	指定された書式を使用して、現在の Rational オブジェクトの数値をそれと等価な文字列形式に変換します。
ToString(String,IFormatProvider)	指定された書式とカルチャ固有の書式情報を使用して、現在の Rational オブジェクトの数値をそれと等価の文字列形式に変換します。
ToUInt16(Rational)	指定した Rational の値を、等価の 16 ビット符号付き整数に変換します。
ToUInt32(Rational)	指定した Rational の値を、等価の 32 ビット符号付き整数に変換します。
ToUInt64(Rational)	指定した Rational の値を、等価の 64 ビット符号付き整数に変換します。
ToDecimal(Rational)	指定した Rational の値を、等価の 10 進数に変換します。
TryParse(String)	数値の文字列形式を対応する Rational 表現に変換できるかどうかを試行し、変換に成功したかどうかを示す値を返します。
TryParse(String,NumberStyles style,IFormatProvider)	数値の文字列形式を対応する Rational 表現に変換できるかどうかを試行し、変換に成功したかどうかを示す値を返します。
Xor(Rational,Rational)	2 つの Rational 値に対し、ビットごとの排他的 Or (XOr) 演算を実行します。

演算子

Addition (Rational,Rational)	指定された 2 つの Rational オブジェクトの値を加算します。
BitwiseAnd (Rational,Rational)	2 つの BigInteger 値に対し、ビットごとの And 演算を実行します。
BitwiseOr (Rational,Rational)	2 つの Rational 値に対し、ビットごとの Or 演算を実行します。
Decrement (Rational)	Rational 値を 1 だけデクリメントします。
Division (Rational,Rational)	整数除算を使用して、指定された Rational 値をもう 1 つの指定された Rational 値で除算します。
Equality (Rational,Rational)	2 つの Rational オブジェクトの値が等しいかどうかを示す値を返します。
ExclusiveOr (Rational,Rational)	2 つの Rational 値に対し、ビットごとの排他的 Or (XOr) 演算を実行します。
Explicit(Rational to Byte)	Rational オブジェクトから byte 値への明示的な変換を定義します。
Explicit(Rational to SByte)	Rational オブジェクトから sbyte 値への明示的な変換を定義します。
Explicit(Rational to Int32)	Rational オブジェクトから int 値への明示的な変換を定義します。
Explicit(Rational to UInt32)	Rational オブジェクトから uint 値への明示的な変換を定義します。
Explicit(Rational to Int16)	Rational オブジェクトから short 値への明示的な変換を定義します。
Explicit(Rational to UInt16)	Rational オブジェクトから ushort 値への明示的な変換を定義します。
Explicit(Rational to Int64)	Rational オブジェクトから long 値への明示的な変換を定義します。
Explicit(Rational to UInt64)	Rational オブジェクトから ulong 値への明示的な変換を定義します。
Explicit(Rational to Single)	Rational オブジェクトから float 値への明示的な変換を定義します。

Explicit(Rational to Double)	Rational オブジェクトから double 値への明示的な変換を定義します。
Explicit(Rational to Boolean)	Rational オブジェクトから bool 値への明示的な変換を定義します。
Explicit(Rational to Decimal)	Rational オブジェクトから decimal 値への明示的な変換を定義します。
GreaterThan (Rational,Rational)	Rational 値がもう 1 つの Rational 値より大きいかどうかを示す値を返します。
GreaterThanOrEqual (Rational,Rational)	Rational 値がもう 1 つの Rational 値以上かどうかを示す値を返します。
Implicit(Byte to Rational)	byte 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(SByte to Rational)	sbyte 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Int32 to Rational)	int 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(UInt32 to Rational)	uint 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Int16 to Rational)	short 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(UInt16 to Rational)	ushort 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Int64 to Rational)	long 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(UInt64 to Rational)	ulong 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Single to Rational)	float 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Double to Rational)	double 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Boolean to Rational)	bool 値から BigInteger 値への暗黙的な変換を定義します。
Implicit(Decimal to Rational)	decimal 値から BigInteger 値への暗黙的な変換を定義します。
Increment (Rational)	Rational 値を 1 だけインクリメントします。

Inequality (Rational,Rational)	2 つの Rational オブジェクトの値が異なるかどうかを示す値を返します。
LeftShift (Rational,Int32)	指定されたビット数だけ Rational 値を左にシフトします。
LessThan (Rational,Rational)	Rational 値がもう 1 つの Rational 値より小さいかどうかを示す値を返します。
LessThanOrEqual (Rational,Rational)	Rational 値がもう 1 つの Rational 値以下かどうかを示す値を返します。
Modulus (Rational,Rational)	指定された 2 つの Rational 値の除算の結果生じた剰余を返します。
Multiply (Rational,Rational)	指定された 2 つの Rational 値を乗算します。
OnesComplement (Rational)	Rational 値のビットごとの 1 の補数を返します。
RightShift (Rational,Int32)	指定されたビット数だけ Rational 値を右にシフトします。
Subtraction (Rational,Rational)	Rational 値をもう 1 つの Rational 値から減算します。
UnaryNegation (Rational)	指定された Rational 値を否定（負数化）します。
UnaryPlus (Rational)	Rational オペランドの値を返します。 オペランドの符号は変更されません。

適用対象

.NET Core

2.0

.NET Framework

4.6.1 2

.NET Standard

2.0

UWP

10.0.16299

Xamarin.Android

8.0

Xamarin.iOS

10.14

Xamarin.Mac

3.8