compact1, compact2, compact3

java.lang

Class Math

java.lang.Object
    java.lang.Math

```
public final class Math
extends Object
```

Math类包含执行基本数字运算的方法，如基本指数，对数，平方根和三角函数。

与StrictMath类的一些数字方法不同，Math类的StrictMath所有Math都没有定义为返回比特位相同的结果。 这种放松允许在不需要严格再现性的情况下执行更好的实现。

默认情况下，许多Math方法只需调用中的对应方法是StrictMath组织落实。 鼓励代码生成器使用平台特定的本机库或微处理器指令（如果可用），以提供Math方法的Math实现。 这种更高性能的实现仍然必须符合Math的Math 。

实施规范的质量有两个属性，返回结果的准确性和方法的单调性。 的浮点精度Math方法在*ULPS*，单位在最后的地方来衡量。 对于给定的浮点格式， ulp特定实数值的是两个浮点值包围该数值之间的距离。 当讨论整个方法的准确性而不是一个具体的参数时，引用的ulps的数量是任何参数的最坏情况错误。 如果方法总是出现小于0.5 ulps的错误，则该方法总是返回最接近精确结果的浮点数; 这种方法是*正确的四舍五入* 。 一个正确舍入的方法通常是一个最好的浮点近似可以; 然而，对于许多浮点方法来说，这是不切实际的。 相反，对于Math类，某些方法允许更大的1或2 Math的误差范围。 非正式地，使用1 ul的错误限制，当确切的结果是可表示的数字时，应该将精确的结果作为计算结果返回; 否则，可以返回包含确切结果的两个浮点值之一。 对于大量的精确结果，支架的端点之一可能是无限大的。 除了个人论证的准确性之外，维持不同论点的方法之间的适当关系也很重要。 因此，绝大多数0.5 ulp误差的方法都是*半单调的* ：每当数学函数不减小时，数学函数也是不减少的，同样地，当数学函数不增加时，浮点数近似也是如此点近似。 并非所有具有1 ulp精度的近似值将自动满足单调性要求。

该平台使用带有int和long基元类型的带符号二进制补码整数运算。 开发人员应该选择原始类型，以确保算术运算始终产生正确的结果，这在某些情况下意味着操作不会溢出计算值的范围。 最佳做法是选择原始类型和算法以避免溢出。 在情况下，大小为int或long ，需要检测溢出错误，方法addExact ， subtractExact ， multiplyExact和toIntExact抛出ArithmeticException结果溢出时。 对于其他算术运算，如除法，绝对值，递增，递减和否定溢出仅在特定的最小值或最大值发生时，应根据最小值或最大值进行检查。

从以下版本开始：

JDK1.0

Field Summary

Fields

Modifier and Type	Field and Description
static double	**E** double值比其他任何一个都更接近 e ， 自然对数的基数。
static double	**PI** double值比任何其他的更接近 pi ， 圆周长与其直径的比率。

方法摘要

所有方法

静态方法

具体的方法

Modifier and Type	Method and Description
static double	**abs**(double a) 返回值为 double绝对值。
static float	**abs**(float a) 返回 float值的绝对值。
static int	**abs**(int a)

	返回值为 int绝对值。
static long	**abs**(long a) 返回值为 long绝对值。
static double	**acos**(double a) 返回值的反余弦值; 返回的角度在0.0到*pi*的范围内。
static int	**addExact**(int x, int y) 返回其参数的总和，如果结果溢出int，则抛出 int 。
static long	**addExact**(long x, long y) 返回其参数的总和，如果结果溢出long，则抛出 long 。
static double	**asin**(double a) 返回值的正弦值; 返回角度在*pi* / 2到*pi* / 2的范围内。
static double	**atan**(double a) 返回值的反正切值; 返回角度在*pi* / 2到*pi* / 2的范围内。
static double	**atan2**(double y, double x) 返回从直角坐标（转换角度 *theta* x ， y ）为极坐标 （*R*，*θ*-）。
static double	**cbrt**(double a) 返回 double值的多维数据集根。
static double	**ceil**(double a) 返回大于或等于参数的最小（最接近负无穷大） double值，等于一个数学整数。
static double	**copySign**(double magnitude, double sign) 使用第二个浮点参数的符号返回第一个浮点参数。
static float	**copySign**(float magnitude, float sign) 使用第二个浮点参数的符号返回第一个浮点参数。
static double	**cos**(double a) 返回角度的三角余弦。

static double	**cosh**(double x) 返回的双曲余弦 double值。
static int	**decrementExact**(int a) 返回一个递减1的参数，如果结果溢出int，则 int 。
static long	**decrementExact**(long a) 将返回的参数递减1，如果结果溢出long，则 long 。
static double	**exp**(double a) 返回欧拉的数字 *e*提高到一个 double价值。
static double	**expm1**(double x) 返回 e^x-1。
static double	**floor**(double a) 返回小于或等于参数的最大（最接近正无穷大） double值，等于一个数学整数。
static int	**floorDiv**(int x, int y) 返回小于或等于代数商的最大（最接近正无穷大） int值。
static long	**floorDiv**(long x, long y) 返回小于或等于代数商的最大（最接近正无穷大） long值。
static int	**floorMod**(int x, int y) 返回 int参数的底部模数。
static long	**floorMod**(long x, long y) 返回 long参数的底模数。
static int	**getExponent**(double d) 返回a的表示中使用的无偏指数 double 。
static int	**getExponent**(float f) 返回a的表示中使用的无偏指数 float 。
static double	**hypot**(double x, double y) 返回sqrt（x^2+y^2），没有中间溢出或下溢。

static double	**IEEEremainder**(double f1, double f2) 根据IEEE 754标准计算两个参数的余数运算。
static int	**incrementExact**(int a) 返回自变量1，如果结果溢出int，则 int 。
static long	**incrementExact**(long a) 返回一个增加1的参数，如果结果溢出long，则 long 。
static double	**log**(double a) 返回的自然对数（以 *e*为底） double值。
static double	**log10**(double a) 返回一个 double的基数10对数值。
static double	**log1p**(double x) 返回参数和1的和的自然对数。
static double	**max**(double a, double b) 返回两个 double值中的较大值。
static float	**max**(float a, float b) 返回两个 float的较大值。
static int	**max**(int a, int b) 返回两个 int值中的较大值。
static long	**max**(long a, long b) 返回两个 long的较大值。
static double	**min**(double a, double b) 返回两个 double的较小值。
static float	**min**(float a, float b) 返回两个 float的较小值。
static int	**min**(int a, int b) 返回两个 int的较小值。

static long	**min**(long a, long b) 返回两个 long的较小值。
static int	**multiplyExact**(int x, int y) 返回参数的乘积，如果结果溢出int，则抛出 int 。
static long	**multiplyExact**(long x, long y) 返回参数的乘积，如果结果溢出long，则抛出 long 。
static int	**negateExact**(int a) 返回参数的否定，如果结果溢出int，则 int 。
static long	**negateExact**(long a) 返回参数的否定，如果结果溢出long，则 long 。
static double	**nextAfter**(double start, double direction) 返回与第二个参数方向相邻的第一个参数的浮点数。
static float	**nextAfter**(float start, double direction) 返回与第二个参数方向相邻的第一个参数的浮点数。
static double	**nextDown**(double d) 返回与负无穷大方向相邻的 d的浮点值。
static float	**nextDown**(float f) 返回与负无穷大方向相邻的 f的浮点值。
static double	**nextUp**(double d) 返回与正无穷大方向相邻的 d的浮点值。
static float	**nextUp**(float f) 返回与正无穷大方向相邻的 f的浮点值。
static double	**pow**(double a, double b) 将第一个参数的值返回到第二个参数的幂。
static double	**random**() 返回值为 double值为正号，大于等于 0.0 ，小于 1.0 。

static double	**rint**(double a) 返回与参数最接近值的 double值，并且等于数学整数。
static long	**round**(double a) 返回参数中最接近的 long ，其中 long四舍五入为正无穷大。
static int	**round**(float a) 返回参数中最接近的 int ，其中 int四舍五入为正无穷大。
static double	**scalb**(double d, int scaleFactor) 返回 d × 2 scaleFactor四舍五入，好像由单个正确四舍五入的浮点乘以双重值集合的成员执行。
static float	**scalb**(float f, int scaleFactor) 返回 f × 2 scaleFactor四舍五入，就像一个正确圆形的浮点数乘以浮点值集合的成员一样。
static double	**signum**(double d) 返回参数的signum函数; 如果参数为零，则为零，如果参数大于零则为1.0，如果参数小于零，则为-1.0。
static float	**signum**(float f) 返回参数的signum函数; 如果参数为零，则为零，如果参数大于零则为1.0f，如果参数小于零，则为 -1.0f。
static double	**sin**(double a) 返回角度的三角正弦。
static double	**sinh**(double x) 返回的双曲正弦 double值。
static double	**sqrt**(double a) 返回的正确舍入正平方根 double值。
static int	**subtractExact**(int x, int y) 返回参数的差异，如果结果溢出int，则抛出 int 。
static long	**subtractExact**(long x, long y) 返回参数的差异，如果结果溢出long，则抛出 long 。

static double	**tan**(double a) 返回角度的三角正切。
static double	**tanh**(double x) 返回的双曲正切 double值。
static double	**toDegrees**(double angrad) 将以弧度测量的角度转换为以度为单位的近似等效角度。
static int	**toIntExact**(long value) 返回long参数的值; 如果值溢出int，则int 。
static double	**toRadians**(double angdeg) 将以度为单位的角度转换为以弧度测量的大致相等的角度。
static double	**ulp**(double d) 返回参数的ulp的大小。
static float	**ulp**(float f) 返回参数的ulp的大小。

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

字段详细信息

E

```
public static final double E
```

double值比其他任何一个更接近 e，自然对数的基数。

另请参见：

Constant Field Values

PI

```
public static final double PI
```

double值比任何其他的更接近 pi，圆周长与其直径的比率。

另请参见：

Constant Field Values,

方法详细信息

sin

```
public static double sin(double a)
```

返回角度的三角正弦。 特殊情况：

- 如果参数为NaN或无穷大，则结果为NaN。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 以弧度表示的角度。

结果

论证的正义。

cos

```
public static double cos(double a)
```

返回角度的三角余弦。 特殊情况：

- 如果参数为NaN或无穷大，则结果为NaN。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 以弧度表示的角度。

结果

论证的余弦。

tan

```
public static double tan(double a)
```

返回角度的三角正切。 特殊情况：

- 如果参数为NaN或无穷大，则结果为NaN。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 以弧度表示的角度。

结果

争论的切线。

asin

```
public static double asin(double a)
```

返回值的正弦值; 返回角度在*pi* / 2到*pi* / 2的范围内。 特殊情况：

- 如果参数为NaN或其绝对值大于1，则结果为NaN。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 要返回正弦的值。

结果

争论的正弦。

acos

```
public static double acos(double a)
```

返回值的反余弦值; 返回的角度在0.0到*pi*的范围内。 特例：

- 如果参数为NaN或其绝对值大于1，则结果为NaN。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 要返回的余弦值。

结果

论证的反余弦。

atan

```
public static double atan(double a)
```

返回值的反正切值; 返回角度在*pi* / 2到*pi* / 2的范围内。 特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 要返回的反正切值。

结果

论点的反正切。

toRadians

```
public static double toRadians(double angdeg)
```

将以度为单位的角度转换为以弧度测量的大致相等的角度。 从度数到弧度的转换通常是不准确的。

参数

angdeg – 以度为单位的角度

结果

测量角度 angdeg （弧度）。

从以下版本开始：

1.2

toDegrees

```
public static double toDegrees(double angrad)
```

将以弧度测量的角度转换为以度为单位的近似等效角度。 从弧度到度的转换通常是不准确的; 用户*不*应该指望`cos(toRadians(90.0))`正好等于`0.0` 。

参数

`angrad` – 以弧度表示的角度

结果

测量角度 `angrad`的度数。

从以下版本开始:

`1.2`

exp

```
public static double exp(double a)
```

返回欧拉的数字*e*提高到`double`价值。 特殊情况:

- 如果参数是NaN，结果是NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为负无穷大，则结果为正零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

`a` – 提高 *e*的指数。

结果

值 e^{a} ，其中 *e*是自然对数的基数。

log

```
public static double log(double a)
```

返回的自然对数（以*e*为底） double值。 特殊情况：

- 如果参数为NaN或小于零，则结果为NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为正零或负零，则结果为负无穷大。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 一个值

结果

值ln a ，自然对数为 a 。

log10

```
public static double log10(double a)
```

返回一个double的基数10对数值。 特殊情况：

- 如果参数为NaN或小于零，则结果为NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为正零或负零，则结果为负无穷大。
- 如果参数等于10^{n}，用于整数n，则结果为n。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

a – 一个值

结果

基数10对数为 a 。

从以下版本开始：

1.5

sqrt

```
public static double sqrt(double a)
```

返回double值正确舍入的正平方根。 特殊情况：
- 如果参数为NaN或小于零，则结果为NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为正零或负零，则结果与参数相同。

否则，结果是double最接近参数值的真实数学平方根值。

参数

a – 一个值。

结果

正平方根a 。 如果参数为NaN或小于零，则结果为NaN。

cbrt

```
public static double cbrt(double a)
```

返回double值的多维数据集根。 对于正有限x， cbrt(-x) == -cbrt(x)；也就是说，负值的多边形根是该值的大小的立方根的负数。 特殊情况：
- 如果参数是NaN，那么结果是NaN。
- 如果参数是无限的，则结果是与参数具有相同符号的无穷大。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。

参数

a – 一个值。

结果

a 。

从以下版本开始：

1.5

IEEEremainder

```
public static double IEEEremainder(double f1,
```

double f2)

根据IEEE 754标准计算两个参数的余数运算。 余数的算术值等于`f1 - f2` *A-n*，其中*n*是最接近商的精确算术值的数学整数`f1/f2` ，并且如果两个整数都同样接近`f1/f2` ，那么*n*是其中的偶数。 如果余数为零，其符号与第一个参数的符号相同。 特殊情况：

- 如果任一参数为NaN，或第一个参数为无穷大，或第二个参数为正零或负零，则结果为NaN。
- 如果第一个参数是有限的，第二个参数是无穷大的，那么结果与第一个参数相同。

参数

`f1` – 股息。

`f2` – 除数。

结果

剩余时间为 `f1`除以 `f2` 。

ceil

```
public static double ceil(double a)
```

返回大于或等于参数的最小（最接近负无穷大） `double`值，并等于数学整数。 特殊情况：

- 如果参数值已经等于数学整数，则结果与参数相同。
- 如果参数为NaN或无穷大或正零或负零，则结果与参数相同。
- 如果参数值小于零但大于-1.0，则结果为负零。

需要注意的是价值`Math.ceil(x)`正是价值`-Math.floor(-x)` 。

参数

`a` – 一个值。

结果

最大（最接近负无穷大）的浮点值大于或等于参数，并且等于一个数学整数。

floor

```
public static double floor(double a)
```

返回小于或等于参数的最大（最接近正无穷大） `double`值，等于数学整数。 特殊情况：

- 如果参数值已经等于数学整数，则结果与参数相同。

- 如果参数为NaN或无穷大或正零或负零，则结果与参数相同。

参数

a – 一个值。

结果

小于或等于参数的最大（最接近正无穷大）浮点值，等于数学整数。

rint

```
public static double rint(double a)
```

返回与参数最接近值的double值，等于数学整数。 如果作为数学整数的两个double值同样接近，则结果为均匀的整数值。 特殊情况：

- 如果参数值已经等于数学整数，则结果与参数相同。
- 如果参数为NaN或无穷大或正零或负零，则结果与参数相同。

参数

a – 一个 double价值。

结果

最接近的浮点值为 a ，等于一个数学整数。

atan2

```
public static double atan2(double y,
                           double x)
```

返回从直角坐标（转换角度*theta* x ， y ）为极坐标（*R*，*θ*-）。 该方法通过计算反正切计算的相位*THETA* y/x *PI*到*pi* -在的范围内。 特殊情况：

- 如果任一参数为NaN，则结果为NaN。
- 如果第一个参数为正零，第二个参数为正，或第一个参数为正和有限，第二个参数为正无穷大，则结果为正零。
- 如果第一个参数为负零，第二个参数为正，或第一个参数为负，有限，第二个参数为正无穷大，则结果为负零。
- 如果第一个参数为正零，第二个参数为负，或第一个参数为正和有限，第二个参数为负无穷大，则结果为最接近*pi*的double值。
- 如果第一个参数为负零，第二个参数为负，或者第一个参数为负，有限，第二个参数为负无穷大，则结果为最接近*pi*的double值。
- 如果第一个参数为正，第二个参数为正零或负零，或第一个参数为正无穷大，第二个参数为有限，则结果为最接近*pi* / 2的double值。
- 如果第一个参数为负，第二个参数为正零或负零，或者第一个参数为负无穷大，第二个参数为有限，则结果为最接近-*pi* / 2的值为

double 。
- 如果两个参数均为无穷大，则结果为最接近*pi* / 4的double值。
- 如果第一个参数为正无穷大，第二个参数为负无穷大，则结果为最接近3 * *pi* / 4的double值。
- 如果第一个参数为负无穷大，第二个参数为正无穷大，则结果为最接近-*pi* / 4的double值。
- 如果两个参数均为负无穷大，则结果为最接近-3 * *pi* / 4的double值。

计算结果必须在精确结果的2 ulps之内。 结果必须是半单调的。

参数

y – 纵坐标坐标

x – 横坐标坐标

结果

对应于在笛卡尔坐标系的点 *(x, y)*的极坐标点 *(R, θ–)*的 *theta*部分。

pow

```
public static double pow(double a,
                         double b)
```

将第一个参数的值返回到第二个参数的幂。 特殊情况：
- 如果第二个参数为正或负零，则结果为1.0。
- 如果第二个参数为1.0，则结果与第一个参数相同。
- 如果第二个参数是NaN，那么结果是NaN。
- 如果第一个参数是NaN，第二个参数是非零，那么结果是NaN。
- 如果
  - 第一个参数的绝对值大于1，第二个参数为正无穷大，或
  - 第一个参数的绝对值小于1，第二个参数为负无穷大，

  那么结果是正无穷大。
- 如果
  - 第一个参数的绝对值大于1，第二个参数为负无穷大，或
  - 第一个参数的绝对值小于1，第二个参数为正无穷大，

  那么结果是正零。
- 如果第一个参数的绝对值等于1，第二个参数为无穷大，则结果为NaN。
- 如果
  - 第一个参数是正零，第二个参数大于零，或者

    - 第一个参数是正无穷大，第二个参数小于零，

  那么结果是正零。
- 如果
    - 第一个参数是正零，第二个参数小于零，或者
    - 第一个参数是正无穷大，第二个参数大于零，

  那么结果是正无穷大。
- 如果
    - 第一个参数是负零，第二个参数大于零，但不是有限的奇整数，或者
    - 第一个参数是负无穷大，第二个参数小于零，但不是有限奇整数，

  那么结果是正零。
- 如果
    - 第一个参数为负零，第二个参数为正有限奇数整数，或
    - 第一个参数为负无穷大，第二个参数为负有限奇数整数，

  那么结果是负零。
- 如果
    - 第一个参数为负零，第二个参数小于零，但不是有限奇整数
    - 第一个参数为负无穷大，第二个参数大于零但不是有限奇整数，

  那么结果是正无穷大。
- 如果
    - 第一个参数为负零，第二个参数为负有限奇数整数，或
    - 第一个参数是负无穷大，第二个参数是正有限奇数整数，

  那么结果是负无穷大。
- 如果第一个参数是有限的并且小于零
    - 如果第二个参数是有限的偶数整数，则结果等于将第一个参数的绝对值提高到第二个参数的幂的结果
    - 如果第二个参数是有限的奇整数，则结果等于将第一个参数的绝对值提高到第二个参数的幂的结果的负数
    - 如果第二个参数是有限的，而不是一个整数，则结果是NaN。
- 如果两个参数都是整数，那么结果完全等于将第一个参数提升到第二个参数的幂的数学结果，如果该结果事实上可以完全按照`double`值进行表示。

（在前述的说明中，一个浮点值被认为是一个整数，当且仅当它是有限的，并且该方法的一个固定点`ceil` ，或等价地，该方法的一个固定点`floor` 。的值是一个固定的点的唯一方法，当且仅当将该方法应用于该值的结果等于该值时）。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

`a` – 基地。

b – 指数。

结果

值 a b 。

round

```
public static int round(float a)
```

返回参数最接近的int，其中int四舍五入为正无穷大。

特殊情况：

- 如果参数是NaN，结果为o。
- 如果参数为负无穷大或小于或等于Integer.MIN_VALUE的值，则结果等于Integer.MIN_VALUE的值。
- 如果参数为正无穷大或大于或等于Integer.MAX_VALUE的值，则结果等于Integer.MAX_VALUE的值。

参数

a – 要舍入为整数的浮点值。

结果

参数的值四舍五入到最接近的 int值。

另请参见：

Integer.MAX_VALUE , Integer.MIN_VALUE

round

```
public static long round(double a)
```

返回参数中最接近的long，其中long四舍五入为正无穷大。

特殊情况：

- 如果参数是NaN，结果为o。
- 如果参数为负无穷大或小于或等于Long.MIN_VALUE的值，则结果等于Long.MIN_VALUE的值。
- 如果参数为正无穷大或大于或等于Long.MAX_VALUE的值，则结果等于Long.MAX_VALUE的值。

参数

a – 要舍入为 long的浮点值。

结果

参数的值四舍五入到最接近的 long值。

另请参见:

Long.MAX_VALUE , Long.MIN_VALUE

random

```
public static double random()
```

返回一个double值为正号，大于等于0.0 ，小于1.0 。 返回的值是从该范围（大约）均匀分布而伪随机选择的。

首先调用此方法时，它将创建一个新的伪随机数生成器，就像表达式一样

```
new java.util.Random()
```

此新的伪随机数生成器此后用于对该方法的所有调用，并在其他地方使用。

该方法正确同步，以允许多个线程正确使用。 然而，如果许多线程需要以很高的速率产生伪随机数，则可以减少每个线程的争用以拥有自己的伪随机数发生器。

结果

一个伪随机 double大于或等于 0.0并小于 1.0 。

另请参见:

Random.nextDouble()

addExact

```
public static int addExact(int x,
                           int y)
```

返回其参数的总和，如果结果溢出int，则抛出 int 。

参数

x - 第一个值

y - 第二个值

结果

结果

异常

ArithmeticException - 如果结果溢出一个int

从以下版本开始：

1.8

addExact

```
public static long addExact(long x,
                            long y)
```

返回其参数的总和，如果结果溢出long，则抛出 long 。

参数

x - 第一个值

y - 第二个值

结果

结果

异常

ArithmeticException - 如果结果溢出了一个长的

从以下版本开始：

1.8

subtractExact

```
public static int subtractExact(int x,
                                int y)
```

返回参数的差异，如果结果溢出int，则抛出 int 。

参数

x – 第一个值

y – 从第一个值减去的第二个值

结果

结果

异常

ArithmeticException – 如果结果溢出一个int

从以下版本开始：

1.8

subtractExact

```
public static long subtractExact(long x,
                                 long y)
```

返回参数的差异，如果结果溢出long，则抛出 long 。

参数

x – 第一个值

y – 从第一个值减去的第二个值

结果

结果

异常

ArithmeticException – 如果结果溢出了很长时间

从以下版本开始：

1.8

multiplyExact

```
public static int multiplyExact(int x,
                                int y)
```

返回参数的乘积，如果结果溢出int，则抛出 int 。

参数

x - 第一个值

y - 第二个值

结果

结果

异常

ArithmeticException - 如果结果溢出一个int

从以下版本开始：

1.8

multiplyExact

```
public static long multiplyExact(long x,
                                 long y)
```

返回参数的乘积，如果结果溢出long，则抛出 long 。

参数

x - 第一个值

y - 第二个值

结果

结果

异常

ArithmeticException - 如果结果溢出了很长时间

从以下版本开始：

1.8

incrementExact

```
public static int incrementExact(int a)
```

返回一个增加1的参数，如果结果溢出int，则 `int` 。

参数

`a` – a的值

结果

结果

异常

`ArithmeticException` – 如果结果溢出一个int

从以下版本开始：

1.8

incrementExact

```
public static long incrementExact(long a)
```

返回自变量1，如果结果溢出long，则 `long` 。

参数

`a` – a的值

结果

结果

异常

`ArithmeticException` – 如果结果溢出很长

从以下版本开始：

1.8

decrementExact

```
public static int decrementExact(int a)
```

返回一个递减1的参数，如果结果溢出int，则 int 。

参数

a – a的值

结果

结果

异常

ArithmeticException – 如果结果溢出一个int

从以下版本开始：

1.8

decrementExact

```
public static long decrementExact(long a)
```

返回一个递减1的参数，如果结果溢出long则 long 。

参数

a – a的值

结果

结果

异常

ArithmeticException – 如果结果溢出很长

从以下版本开始：

1.8

negateExact

```
public static int negateExact(int a)
```

返回参数的否定，如果结果溢出int，则 int 。

参数

a – 否定的值

结果

结果

异常

ArithmeticException – 如果结果溢出一个int

从以下版本开始：

1.8

negateExact

```
public static long negateExact(long a)
```

返回参数的否定，如果结果溢出long，则 long 。

参数

a – 否定的值

结果

结果

异常

ArithmeticException – 如果结果溢出了很长时间

从以下版本开始：

1.8

toIntExact

```
public static int toIntExact(long value)
```

返回long参数的值; 如果值溢出int，则int 。

参数

value - 长价值

结果

作为int的参数

异常

ArithmeticException - 如果 argument溢出一个int

从以下版本开始:

1.8

floorDiv

```
public static int floorDiv(int x,
                           int y)
```

返回小于或等于代数商的最大（最接近正无穷大） int值。 有一个特殊情况，如果股息为Integer.MIN_VALUE ，除数为-1 ，则发生整数溢出，结果等于Integer.MIN_VALUE 。

正常整数除法在四舍五入方式（截断）下运行。 这个操作代替在向下的无穷大（倒圆）舍入模式下。 当精确结果为负时，地板舍入模式提供与截断不同的结果。

- 如果参数的符号相同，那么floorDiv和/操作符的结果是一样的。
  例如， floorDiv(4, 3) == 1和(4 / 3) == 1 。
- 如果参数的符号不同，则商为负， floorDiv返回小于或等于商的整数， /操作符返回最接近零的整数。
  例如， floorDiv(-4, 3) == -2 ，而(-4 / 3) == -1 。

参数

x - 股息

y - 除数

结果

小于或等于代数商的最大（最接近正无穷大） int值。

异常

ArithmeticException - 如果除数 y为零

从以下版本开始:

1.8

另请参见:

floorMod(int, int) , floor(double)

floorDiv

```
public static long floorDiv(long x,
                            long y)
```

返回小于或等于代数商的最大（最接近正无穷大） long值。 有一个特殊情况，如果股息为Long.MIN_VALUE ，除数为-1 ，则发生整数溢出，结果等于Long.MIN_VALUE 。

正常整数除法在四舍五入方式（截断）下运行。 这个操作代替在向下的无穷大（倒圆）舍入模式下。 当精确结果为负时，地板舍入模式提供与截断不同的结果。

例如，参见floorDiv(int, int) 。

参数

x – 股息

y – 除数

结果

小于或等于代数商的最大（最接近正无穷大） long值。

异常

ArithmeticException – 如果除数 y为零

从以下版本开始:

1.8

另请参见:

floorMod(long, long) , floor(double)

floorMod

```
public static int floorMod(int x,
```

int y)

返回int参数的底模。

地板模量为x - (floorDiv(x, y) * y)，具有相同的符号作为除数y，并且是在范围-abs(y) < r < +abs(y)。

之间的关系floorDiv和floorMod是这样的：

- floorDiv(x, y) * y + floorMod(x, y) == x

floorMod和%操作符之间的值之间的%是由于floorDiv之间的floorDiv返回整数小于或等于商，/运算符返回最接近零的整数。

例子：

- 如果参数的符号相同，floorMod和%操作符的结果是一样的。
  - floorMod(4, 3) == 1；和(4 % 3) == 1
- 如果参数的符号不同，结果与%运算符不同。
  - floorMod(+4, -3) == -2；和(+4 % -3) == +1
  - floorMod(-4, +3) == +2；和(-4 % +3) == -1
  - floorMod(-4, -3) == -1；和(-4 % -3) == -1

如果参数的符号未知，并且需要正模数，则可以计算为(floorMod(x, y) + abs(y)) % abs(y)。

参数

x - 股息

y - 除数

结果

地板模数 x - (floorDiv(x, y) * y)

异常

ArithmeticException - 如果除数 y为零

从以下版本开始：

1.8

另请参见：

floorDiv(int, int)

floorMod

```
public static long floorMod(long x,
                            long y)
```

返回long参数的楼层模数。

地板模数为x - (floorDiv(x, y) * y)，与除数y具有相同的符号，并在-abs(y) < r < +abs(y)的范围内。

之间的关系floorDiv和floorMod是这样的：

- floorDiv(x, y) * y + floorMod(x, y) == x

例如，参见floorMod(int, int)。

参数

x - 股息

y - 除数

结果

地板模数 x - (floorDiv(x, y) * y)

异常

ArithmeticException - 如果除数 y为零

从以下版本开始：

1.8

另请参见：

floorDiv(long, long)

abs

```
public static int abs(int a)
```

返回值为int绝对值。 如果参数不为负，则返回参数。 如果参数为负，则返回参数的否定。

请注意，如果参数等于Integer.MIN_VALUE的值，则最负数为int值，结果是相同的值，即为负数。

参数

a – 绝对值要确定的参数

结果

参数的绝对值。

abs

```
public static long abs(long a)
```

返回一个long值的绝对值。 如果参数不为负，则返回参数。 如果参数为负，则返回参数的否定。

注意，如果参数等于Long.MIN_VALUE的值，则最负数可表示long值，结果是相同的值，即为负数。

参数

a – 要确定其绝对值的参数

结果

参数的绝对值。

abs

```
public static float abs(float a)
```

返回一个float值的绝对值。 如果参数不为负，则返回参数。 如果参数为负，则返回参数的否定。 特殊情况：

- 如果参数为正零或负零，结果为正零。
- 如果论证是无限的，结果是正无穷大。
- 如果参数是NaN，结果是NaN。

换句话说，结果与表达式的值相同：

```
Float.intBitsToFloat(0x7fffffff & Float.floatToIntBits(a))
```

参数

a – 绝对值要确定的参数

结果

参数的绝对值。

abs

```
public static double abs(double a)
```

返回一个double值的绝对值。 如果参数不为负，则返回参数。 如果参数为负，则返回参数的否定。 特殊情况：

- 如果参数为正零或负零，结果为正零。
- 如果论证是无限的，结果是正无穷大。
- 如果参数是NaN，结果是NaN。

换句话说，结果与表达式的值相同：

```
Double.longBitsToDouble((Double.doubleToLongBits(a)<<1)>>>1)
```

参数

a – 绝对值要确定的参数

结果

参数的绝对值。

max

```
public static int max(int a,
                      int b)
```

返回两个int的较大值。 也就是说，结果是更接近Integer.MAX_VALUE的价值。 如果参数的值相同，结果是相同的值。

参数

a – 一个论据。

b – 另一个论点。

结果

较大的 a和 b 。

max

```
public static long max(long a,
                       long b)
```

返回两个long的较大值。 也就是说，结果是更接近Long.MAX_VALUE的价值。 如果参数的值相同，结果是相同的值。

参数

a – 一个论据。

b – 另一个论点。

结果

较大的 a和 b 。

max

```
public static float max(float a,
                        float b)
```

返回两个float值中的较大值。 也就是说，结果是更接近正无穷大的论据。 如果参数的值相同，结果是相同的值。 如果任一值为NaN，则结果为NaN。 与数值比较运算符不同，该方法认为负零严格小于正零。 如果一个参数为正零，另一个为负，结果为正零。

参数

a – 一个论据。

b – 另一个论点。

结果

较大的 a和 b 。

max

```
public static double max(double a,
                         double b)
```

返回两个double值中的较大值。 也就是说，结果是更接近正无穷大的论据。 如果参数的值相同，结果是相同的值。 如果任一值为NaN，则结果为NaN。 与数值比较运算符不同，该方法认为负零严格小于正零。 如果一个参数为正零，另一个为负，结果为正零。

参数

a – 一个论据。

b – 另一个论点。

结果

较大的 a和 b 。

min

```
public static int min(int a,
                      int b)
```

返回两个int的较小值。 也就是说，结果是更接近Integer.MIN_VALUE的价值。 如果参数的值相同，结果是相同的值。

参数

a – 一个论点。

b – 另一个论点。

结果

较小的 a和 b 。

min

```
public static long min(long a,
                       long b)
```

返回两个long的较小值。 也就是说，结果是更接近于Long.MIN_VALUE的值。 如果参数的值相同，结果是相同的值。

参数

a – 一个论点。

b – 另一个参数。

结果

较小的 a和 b 。

min

```
public static float min(float a,
                        float b)
```

返回两个float值中的较小值。 也就是说，结果是更接近负无穷大的值。 如果参数的值相同，结果是相同的值。 如果任一值为NaN，则结果为NaN。 与数值比较运算符不同，该方法认为负零严格小于正零。 如果一个参数为正零，另一个为负零，结果为负零。

参数

a – 一个论点。

b – 另一个论点。

结果

较小的 a和 b 。

min

```
public static double min(double a,
                         double b)
```

返回两个double的较小值。 也就是说，结果是更接近负无穷大的值。 如果参数的值相同，结果是相同的值。 如果任一值为NaN，则结果为NaN。 与数值比较运算符不同，该方法认为负零严格小于正零。 如果一个参数为正零，另一个为负零，结果为负零。

参数

a – 一个论据。

b – 另一个论点。

结果

较小的 a和 b 。

ulp

```
public static double ulp(double d)
```

返回参数的ulp的大小。 double值的最后一个double是这个浮点值和double值之间的正距离，其值越大。 请注意，对于非NaN x ，ulp(−x) == ulp(x) 。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数为正或负无穷大，则结果为正无穷大。
- 如果参数为正或负零，则结果为Double.MIN_VALUE 。
- 如果参数为± Double.MAX_VALUE ，则结果等于2^{971} 。

参数

d – 要返回ulp的浮点值

结果

参数的ulp的大小

从以下版本开始:

1.5

ulp

```
public static float ulp(float f)
```

返回参数的ulp的大小。 float的最后一个float是这个浮点值和float值之间的正距离，其值越大。 请注意，对于非NaN x ， ulp(−x) == ulp(x) 。

特殊情况:

- 如果参数是NaN，那么结果是NaN。
- 如果参数为正或负无穷大，则结果为正无穷大。
- 如果参数为正或负零，则结果为Float.MIN_VALUE 。
- 如果参数为± Float.MAX_VALUE ，则结果等于2^{104} 。

参数

f – 要返回ulp的浮点值

结果

参数的ulp的大小

从以下版本开始:

1.5

signum

```
public static double signum(double d)
```

返回参数的signum函数; 如果参数为零，则为零，如果参数大于零则为1.0，如果参数小于零，则为-1.0。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数为正零或负零，则结果与参数相同。

参数

d – 要返回其 d的浮点值

结果

参数的signum函数

从以下版本开始：

1.5

signum

```
public static float signum(float f)
```

返回参数的signum函数; 如果参数为零，则为零，如果参数大于零则为1.0f，如果参数小于零，则为-1.0f。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数为正零或负零，则结果与参数相同。

参数

f – 要返回其 f的浮点值

结果

参数的signum函数

从以下版本开始：

1.5

sinh

```
public static double sinh(double x)
```

返回的双曲正弦double值。 x的双曲正弦定义为（ $e^x - e^{-x}$ ）/ 2，其中e为Euler's number 。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数是无限的，则结果是与参数具有相同符号的无穷大。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果2.5 ul以内。

参数

x – 要返回双曲正弦的数字。

结果

双曲正弦 x 。

从以下版本开始：

1.5

cosh

```
public static double cosh(double x)
```

返回的双曲余弦double值。 x的双曲余弦被定义为（ $e^x + e^{-x}$ ）/ 2，其中e是Euler's number 。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数是无穷大的，则结果是正无穷大。
- 如果参数为零，则结果为1.0 。

计算结果必须在精确结果2.5 ul以内。

参数

x – 要返回双曲余弦的数字。

结果

的双曲余弦 x 。

从以下版本开始：

1.5

tanh

```
public static double tanh(double x)
```

返回一个double值的双曲正切值。 x的双曲正切定义为（ e^x -e^{-x} ）/（ e^x + e^{-x} ），换句话说， sinh(*x*) / cosh(*x*) 。 请注意，确切的tanh的绝对值始终小于1。

特殊情况：

- 如果参数是NaN，那么结果是NaN。
- 如果参数为零，则结果为与参数相同符号的零。
- 如果参数为无穷大，则结果为+1.0 。
- 如果参数为负无穷大，则结果为-1.0 。

计算结果必须在精确结果2.5 ul以内。 任何有限输入的tanh的结果必须具有小于或等于1的绝对值。请注意，一旦tanh的确切结果在±1的极限值的ulp的1/2内，正确签名± 1.0应该退回

参数

x – 要返回其双曲正切的数字。

结果

双曲正切 x 。

从以下版本开始：

1.5

hypot

```
public static double hypot(double x,
                           double y)
```

返回sqrt（ $x^2 + y^2$ ），没有中间溢出或下溢。

特殊情况：

- 如果任一参数为无穷大，则结果为无穷大。
- 如果任一参数为NaN，且两个参数都不是无穷大，则结果为NaN。

计算结果必须在精确结果的1 ulp之内。 如果一个参数保持不变，则其他参数的结果必须是半单调的。

参数

`x` – 一个值

`y` – 一个值

结果

`sqrt ( x ² + y ² )`，没有中间溢出或下溢

从以下版本开始：

`1.5`

expm1

```
public static double expm1(double x)
```

返回e^x-1。 请注意，对于接近0的x的值， `expm1(x)` + 1的确切总和更接近于e^x的真实结果，而不是`exp(x)`。

特殊情况：

- 如果参数是NaN，结果是NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为负无穷大，则结果为-1.0。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。 任何有限输入的`expm1`的结果必须大于或等于`-1.0`。 请注意，一旦e^x的确切结果在限制值-1的1/2 ulp内，则应返回`-1.0`。

参数

`x` – 在 $e^x - 1$的计算中提高 e的指数。

结果

值 $e^x - 1$。

从以下版本开始：

1.5

log1p

```
public static double log1p(double x)
```

返回参数之和1注的自然对数，对于小值x ，结果log1p(x)更接近LN（1 +的真实结果x ）比的浮点评价log(1.0+x) 。

特殊情况：

- 如果参数为NaN或小于-1，则结果为NaN。
- 如果参数为无穷大，则结果为正无穷大。
- 如果参数为负，则结果为负无穷大。
- 如果参数为零，则结果为与参数相同符号的零。

计算结果必须在精确结果的1 ulp之内。 结果必须是半单调的。

参数

x – 一个值

结果

值ln (x + 1) ，自然对数为 x + 1

从以下版本开始：

1.5

copySign

```
public static double copySign(double magnitude,
                              double sign)
```

使用第二个浮点参数的符号返回第一个浮点参数。 请注意，与StrictMath.copySign方法不同，此方法不需要将NaN sign参数视为正值;允许实现将一些NaN参数视为正值，其他NaN参数为负，以允许更高的性能。

参数

magnitude – 提供结果大小的参数

sign – 提供结果符号的参数

结果

价值 magnitude ，标志 sign 。

从以下版本开始：

1.6

copySign

```
public static float copySign(float magnitude,
                             float sign)
```

使用第二个浮点参数的符号返回第一个浮点参数。 请注意，与StrictMath.copySign方法不同，此方法不需要将NaN sign参数视为正值; 允许实现将一些NaN参数视为正值，其他NaN参数为负，以允许更高的性能。

参数

magnitude – 提供结果大小的参数

sign – 提供结果符号的参数

结果

随着大小的值 magnitude和符号 sign 。

从以下版本开始：

1.6

getExponent

```
public static int getExponent(float f)
```

返回a的表示中使用的无偏指数float 。 特殊情况：

- 如果参数为NaN或无限，则结果为Float.MAX_EXPONENT + 1。
- 如果参数为零或次正常，则结果为Float.MIN_EXPONENT -1。

参数

f – 一个 float价值

结果

参数的无偏指数

从以下版本开始：

1.6

getExponent

```
public static int getExponent(double d)
```

返回a的表示中使用的无偏指数double 。 特殊情况：

- 如果参数为NaN或无限，则结果为Double.MAX_EXPONENT + 1。
- 如果参数为零或低于正常，则结果为Double.MIN_EXPONENT -1。

参数

d – 一个 double价值

结果

参数的无偏指数

从以下版本开始：

1.6

nextAfter

```
public static double nextAfter(double start,
                               double direction)
```

返回与第二个参数方向相邻的第一个参数的浮点数。 如果两个参数都相等，则返回第二个参数。

特殊情况：

- 如果任一参数是NaN，则返回NaN。
- 如果两个参数都是带符号的零， direction将被不变地返回（如果参数比较相等则返回第二个参数的要求所暗示的）。
- 如果start为± Double.MIN_VALUE ，并且direction具有使结果应具有较小幅度的值，则返回与start相同符号的零。
- 如果start是无限的， direction具有这样的结果应该有一个小幅度的值Double.MAX_VALUE具有相同的符号start返回。
- 如果start等于± Double.MAX_VALUE ，并且direction具有使得结果应该具有更大幅度的值，则返回与start相同符号的无穷大。

参数

start – 起始浮点值

direction表明其价值– start的邻居或 start应返回

结果

start相邻的浮点数方向为 direction 。

从以下版本开始：

1.6

nextAfter

```
public static float nextAfter(float start,
                              double direction)
```

返回与第二个参数方向相邻的第一个参数的浮点数。 如果两个参数都相等，则返回等于第二个参数的值。

特殊情况：

- 如果任一参数是NaN，则返回NaN。
- 如果两个参数都是带符号的零，则返回direction的值。
- 如果start为± Float.MIN_VALUE ，并且direction具有使结果应具有较小幅度的值，则返回与start相同符号的零。
- 如果start是无限的， direction具有这样的结果应该有一个小幅度的值Float.MAX_VALUE具有相同的符号start返回。
- 如果start等于± Float.MAX_VALUE ，并且direction具有使得结果应该具有较大幅度的值，则返回与start相同符号的无穷大。

参数

start – 起始浮点值

direction表明其价值– start的邻居或 start应返回

结果

start附近的 start在direction的 direction 。

从以下版本开始：

1.6

nextUp

```
public static double nextUp(double d)
```

返回与正无穷大方向相邻的d的浮点值。 这种方法语义上nextAfter(d, Double.POSITIVE_INFINITY)；然而，一个nextUp实现可能比其等效的nextAfter调用运行速度更快。

特殊情况：

- 如果参数是NaN，结果是NaN。
- 如果论证是正无穷大，结果是正无穷大。
- 如果参数为零，则结果为Double.MIN_VALUE

参数

d – 起始浮点值

结果

相邻浮点值接近正无穷大。

从以下版本开始：

1.6

nextUp

```
public static float nextUp(float f)
```

在正无穷大的方向上返回与f相邻的浮点值。 这种方法语义上nextAfter(f, Float.POSITIVE_INFINITY)；但是，一个nextUp实现可能运行速度比其等效的nextAfter调用。

特殊情况：

- 如果参数是NaN，结果是NaN。
- 如果论证是正无穷大，结果是正无穷大。
- 如果参数为零，结果为Float.MIN_VALUE

参数

f – 起始浮点值

结果

相邻浮点值接近正无穷大。

从以下版本开始：

1.6

nextDown

```
public static double nextDown(double d)
```

返回与负无穷大方向相邻的d的浮点值。 这种方法在语义上相当于nextAfter(d, Double.NEGATIVE_INFINITY)；但是，一个nextDown实现可能比其等效的nextAfter调用运行速度更快。

特殊情况：

- 如果参数是NaN，结果是NaN。
- 如果参数为负无穷大，则结果为负无穷大。
- 如果参数为零，结果为-Double.MIN_VALUE

参数

d – 起始浮点值

结果

相邻浮点值接近负无穷大。

从以下版本开始：

1.8

nextDown

```
public static float nextDown(float f)
```

返回与负无穷大方向相邻的f的浮点值。 这种方法在语义上相当于nextAfter(f, Float.NEGATIVE_INFINITY)；但是，一个nextDown实现可能运行速度比其等效的nextAfter调用。

特殊情况：

- 如果参数是NaN，结果是NaN。
- 如果参数为负无穷大，则结果为负无穷大。
- 如果参数为零，结果为-Float.MIN_VALUE

参数

`f` – 起始浮点值

结果

相邻浮点值接近负无穷大。

从以下版本开始:

`1.8`

scalb

```
public static double scalb(double d,
                           int scaleFactor)
```

返回`d` × 2^`scaleFactor`四舍五入，好像由单个正确四舍五入的浮点乘以双重值集合的成员执行。 有关浮点值集的讨论，请参阅Java语言规范。 如果结果的指数在`Double.MIN_EXPONENT`和`Double.MAX_EXPONENT`之间，那么答案是准确计算的。 如果结果的指数大于`Double.MAX_EXPONENT`，则返回无限大。 请注意，如果结果异常，则精度可能会丢失; 也就是说，当`scalb(x, n)`是次正常时，`scalb(scalb(x, n), -n)`可能不等于x。 当结果为非NaN时，结果与`d`具有相同的符号。

特殊情况:

- 如果第一个参数是NaN，则返回NaN。
- 如果第一个参数为无穷大，则返回相同符号的无穷大。
- 如果第一个参数为零，则返回相同符号的零。

参数

`d` – `d`两个幂来缩放的号码。

`scaleFactor` – 2的功率用于缩放 `d`

结果

`d` × 2^`scaleFactor`

从以下版本开始:

`1.6`

scalb

```
public static float scalb(float f,
                          int scaleFactor)
```

返回`f` × 2^`scaleFactor`四舍五入，就像一个正确圆形的浮点数乘以浮点值集合的成员一样。 有关浮点值集的讨论，请参阅Java语言规范。 如果结果的指数在`Float.MIN_EXPONENT`和`Float.MAX_EXPONENT`之间，那么答案是准确计算的。 如果结果的指数大于`Float.MAX_EXPONENT` ，则返回无限大。 请注意，如果结果异常，则精度可能会丢失; 也就是说，当`scalb(x, n)`是次正常时，`scalb(scalb(x, n), -n)`可能不等于x 。 当结果为非NaN时，结果与`f`具有相同的符号。

特殊情况：

- 如果第一个参数是NaN，则返回NaN。
- 如果第一个参数为无穷大，则返回相同符号的无穷大。
- 如果第一个参数为零，则返回相同符号的零。

参数

`f` – `f`两个幂来缩放的数字。

`scaleFactor` – 2的功率用于缩放 `f`

结果

`f` × 2^`scaleFactor`

从以下版本开始：

`1.6`