

This specification is not final and is subject to change. Use is subject to [license terms](#).

## Java SE 24 & JDK 24 DRAFT 24-internal-adhoc.darcy.open

### Contents

[Description](#)[Field Summary](#)[Method Summary](#)[Field Details](#)[POSITIVE\\_INFINITY](#)[NEGATIVE\\_INFINITY](#)[NaN](#)[MAX\\_VALUE](#)[MIN\\_NORMAL](#)[MIN\\_VALUE](#)[SIZE](#)[PRECISION](#)[MAX\\_EXPONENT](#)[MIN\\_EXPONENT](#)[BYTES](#)

### Method Details

## Class Float16

[java.lang.Object](#)[java.lang.Number](#)[jdk.incubator.vector.Float16](#)

### All Implemented Interfaces:

[Serializable](#), [Comparable<Float16>](#)

```
public final class Float16
```

```
extends Number
```

```
implements Comparable<Float16>
```

The Float16 is a class holding 16-bit data in IEEE 754 binary16 format.

Binary16 Format:

S EEEEE MMMMMMMMMM

Sign - 1 bit

Exponent - 5 bits

Significand - 10 bits (does not include the *implicit bit* inferred from the exponent, see [PRECISION](#))

Unless otherwise specified, the methods in this class use a *rounding policy* (JLS 15.4 ) of [round to nearest](#).

This is a [value-based](#) class; programmers should treat instances that are [equal](#) as interchangeable and should not use instances for synchronization, or unpredictable behavior may occur. For example, in a future release, synchronization may fail.

### API Note:

The methods in this class generally have analogous methods in either [Float/Double](#) or [Math/StrictMath](#). Unless otherwise specified, the handling of special floating-point values such as [NaN](#) values, [infinities](#), and signed zeros of methods in this class is wholly analogous to the handling of equivalent cases by methods in [Float](#), [Double](#), [Math](#), etc.

**Since:**

24

**See Also:**[Serialized Form](#)

## Field Summary

### Fields

static final <b>Float16</b>	<b>POSITIVE_INFINITY</b>	A constant holding the positive infinity of type <code>Float16</code> .
static final int	<b>PRECISION</b>	The number of bits in the significand of a <code>Float16</code> value, 11.
static final int	<b>SIZE</b>	The number of bits used to represent a <code>Float16</code> value, 16.  finite value of type <code>Float16</code> , $(2-2^{-10})\cdot 2^{15}$ , numerically equal to 65504.0.
static final int	<b>MIN_EXPONENT</b>	Minimum exponent a normalized <code>Float16</code> variable may have, -14.
static final <b>Float16</b>	<b>MIN_NORMAL</b>	A constant holding the smallest positive normal value of type <code>Float16</code> , $2^{-14}$ .
static final <b>Float16</b>	<b>MIN_VALUE</b>	A constant holding the smallest positive nonzero value of type <code>Float16</code> , $2^{-24}$ .
static final <b>Float16</b>	<b>NaN</b>	A constant holding a Not-a-Number (NaN) value of type <code>Float16</code> .
static final <b>Float16</b>	<b>NEGATIVE_INFINITY</b>	A constant holding the negative infinity of type <code>Float16</code> .

## Method Summary

All Methods	Static Methods	Instance Methods	Concrete Methods
static <b>String</b>	<b>toHexString</b> ( <b>Float16</b> f16)		Returns a hexadecimal string representation of the <b>Float16</b> argument.
<b>String</b>	<b>toString</b> ()		Returns a string representation of this <b>Float16</b> .
static <b>String</b>	<b>toString</b> ( <b>Float16</b> f16)		Returns a string representation of the <b>Float16</b> argument.
static <b>Float16</b>	<b>ulp</b> ( <b>Float16</b> f16)		Returns the size of an ulp of the argument.
static <b>Float16</b>	<b>valueOf</b> (double d)		Returns a <b>Float16</b> value rounded from the <b>double</b> argument using the round to nearest rounding policy.
static <b>Float16</b>	<b>valueOf</b> (float f)		Returns a <b>Float16</b> value rounded from the <b>float</b> argument using the round to nearest rounding policy.









## Methods declared in class java.lang.Object

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

## Field Details

### POSITIVE\_INFINITY

`public static final Float16 POSITIVE_INFINITY`

A constant holding the positive infinity of type `Float16`.

**See Also:**

`Float.POSITIVE_INFINITY`, `Double.POSITIVE_INFINITY`

### NEGATIVE\_INFINITY

`public static final Float16 NEGATIVE_INFINITY`

A constant holding the negative infinity of type `Float16`.

**See Also:**

`Float.NEGATIVE_INFINITY`, `Double.NEGATIVE_INFINITY`

### NaN

`public static final Float16 NaN`

A constant holding a Not-a-Number (NaN) value of type `Float16`.

**See Also:**

`Float.NaN`, `Double.NaN`



## MAX\_VALUE

```
public static final Float16 MAX_VALUE
```

A constant holding the largest positive finite value of type `Float16`,  $(2 \cdot 2^{-10}) \cdot 2^{15}$ , numerically equal to 65504.0.

**See Also:**

[Float.MAX\\_VALUE](#), [Double.MAX\\_VALUE](#)

## MIN\_NORMAL

```
public static final Float16 MIN_NORMAL
```

A constant holding the smallest positive normal value of type `Float16`,  $2^{-14}$ .

**See Also:**

[Float.MIN\\_NORMAL](#), [Double.MIN\\_NORMAL](#)

## MIN\_VALUE

```
public static final Float16 MIN_VALUE
```

A constant holding the smallest positive nonzero value of type `Float16`,  $2^{-24}$ .

**See Also:**

[Float.MIN\\_VALUE](#), [Double.MIN\\_VALUE](#)

## SIZE

```
public static final int SIZE
```

The number of bits used to represent a `Float16` value, 16.

**See Also:**

[Float.SIZE](#), [Double.SIZE](#), [Constant Field Values](#)

## PRECISION

```
public static final int PRECISION
```

The number of bits in the significand of a `Float16` value, 11. This corresponds to parameter `N` in section 4.2.3 of *The Java Language Specification*.

**See Also:**

[Float.PRECISION](#), [Double.PRECISION](#), [Constant Field Values](#)

## MAX\_EXPONENT

```
public static final int MAX_EXPONENT
```

Maximum exponent a finite `Float16` variable may have, 15. It is equal to the value returned by `Float16.getExponent(Float16.MAX_VALUE)`.

**See Also:**

[Float.MAX\\_EXPONENT](#), [Double.MAX\\_EXPONENT](#), [Constant Field Values](#)

## MIN\_EXPONENT

```
public static final int MIN_EXPONENT
```

Minimum exponent a normalized `Float16` variable may have, -14. It is equal to the value returned by `Float16.getExponent(Float16.MIN_NORMAL)`.

**See Also:**

[Float.MIN\\_EXPONENT](#), [Double.MIN\\_EXPONENT](#), [Constant Field Values](#)

## BYTES

```
public static final int BYTES
```

The number of bytes used to represent a Float16 value, 2.

**See Also:**

[Float.BYTES](#), [Double.BYTES](#), [Constant Field Values](#)

## Method Details

### toString

```
public static String toString(Float16 f16)
```

Returns a string representation of the Float16 argument.

**Implementation Requirements:**

The current implementation acts as if this Float16 were [converted](#) to float and then the string for that float returned. This behavior is expected to change to accommodate the precision of Float16.

**Parameters:**

f16 - the Float16 to be converted.

**Returns:**

a string representation of the argument.

**See Also:**

[Float.toString\(float\)](#)

### toHexString

```
public static String toHexString(Float16 f16)
```

Returns a hexadecimal string representation of the Float16 argument. The behavior of

this class is analogous to `Float.toHexString(float)` except that an exponent value of "p14" is used for subnormal Float16 values.

**Parameters:**

f16 - the Float16 to be converted.

**Returns:**

a hex string representation of the argument.

**See Also:**

`Float.toHexString(float)`, `Double.toHexString(double)`

**valueOf**

```
public static Float16 valueOf(int value)
```

Returns the value of an int converted to Float16.

**API Note:**

This method corresponds to the `convertFromInt` operation defined in IEEE 754.

**Parameters:**

value - an int value.

**Returns:**

the value of an int converted to Float16

**valueOf**

```
public static Float16 valueOf(long value)
```

Returns the value of a long converted to Float16.

**API Note:**

This method corresponds to the `convertFromInt` operation defined in IEEE 754.

**Parameters:**

value - a long value.

**Returns:**

the value of a long converted to Float16

**valueOf**

```
public static Float16 valueOf(float f)
```

Returns a Float16 value rounded from the float argument using the round to nearest rounding policy.

**API Note:**

This method corresponds to the convertFormat operation defined in IEEE 754.

**Parameters:**

f - a float

**Returns:**

a Float16 value rounded from the float argument using the round to nearest rounding policy

**valueOf**

```
public static Float16 valueOf(double d)
```

Returns a Float16 value rounded from the double argument using the round to nearest rounding policy.

**API Note:**

This method corresponds to the convertFormat operation defined in IEEE 754.

**Parameters:**

d - a double

**Returns:**

a Float16 value rounded from the double argument using the round to nearest

rounding policy

## valueOf

```
public static Float16 valueOf(String s)
    throws NumberFormatException
```

Returns a `Float16` holding the floating-point value represented by the argument string.

### Implementation Requirements:

The current implementation acts as if the string were `parsed` as a `double` and then `converted` to `Float16`. This behavior is expected to change to accommodate the precision of `Float16`.

### Parameters:

`s` - the string to be parsed.

### Returns:

the `Float16` value represented by the string argument.

### Throws:

`NullPointerException` - if the string is null

`NumberFormatException` - if the string does not contain a parsable `Float16`.

### See Also:

`Float.valueOf(String)`

## valueOf

```
public static Float16 valueOf(BigDecimal v)
```

Returns a `Float16` value rounded from the `BigDecimal` argument using the round to nearest rounding policy.

### API Note:

This method corresponds to the `convertFormat` operation defined in IEEE 754.

**Parameters:**

v - a [BigDecimal](#)

**Returns:**

a [Float16](#) value rounded from the [BigDecimal](#) argument using the round to nearest rounding policy

## isNaN

```
public static boolean isNaN(Float16 f16)
```

Returns true if the specified number is a Not-a-Number (NaN) value, false otherwise.

**API Note:**

This method corresponds to the isNaN operation defined in IEEE 754.

**Parameters:**

f16 - the value to be tested.

**Returns:**

true if the argument is NaN; false otherwise.

**See Also:**

[Float.isNaN\(float\)](#), [Double.isNaN\(double\)](#)

## isInfinite

```
public static boolean isInfinite(Float16 f16)
```

Returns true if the specified number is infinitely large in magnitude, false otherwise.

**API Note:**

This method corresponds to the isInfinite operation defined in IEEE 754.

**Parameters:**

f16 - the value to be tested.

**Returns:**

true if the argument is positive infinity or negative infinity; false otherwise.

**See Also:**

[Float.isInfinite\(float\)](#), [Double.isInfinite\(double\)](#)

## isFinite

```
public static boolean isFinite(Float16 f16)
```

Returns true if the argument is a finite floating-point value; returns false otherwise (for NaN and infinity arguments).

**API Note:**

This method corresponds to the isFinite operation defined in IEEE 754.

**Parameters:**

f16 - the Float16 value to be tested

**Returns:**

true if the argument is a finite floating-point value, false otherwise.

**See Also:**

[Float.isFinite\(float\)](#), [Double.isFinite\(double\)](#)

## byteValue

```
public byte byteValue()
```

Returns the value of this Float16 as a byte after a narrowing primitive conversion.

**Overrides:**

[byteValue](#) in class [Number](#)

**Returns:**

the value of this Float16 as a byte after a narrowing primitive conversion

**See Java Language Specification:**



### 5.1.3 Narrowing Primitive Conversion

#### toString

```
public String toString()
```

Returns a string representation of this Float16.

**Overrides:**

[toString](#) in class [Object](#)

**Implementation Requirements:**

This method returns the result of `Float16.toString(this)`.

**Returns:**

a string representation of this Float16

#### shortValue

```
public short shortValue()
```

Returns the value of this Float16 as a short after a narrowing primitive conversion.

**Overrides:**

[shortValue](#) in class [Number](#)

**Returns:**

the value of this Float16 as a short after a narrowing primitive conversion

**See *Java Language Specification*:**

[5.1.3 Narrowing Primitive Conversion](#)

#### intValue

```
public int intValue()
```

Returns the value of this Float16 as an int after a narrowing primitive conversion.

**Specified by:**

[intValue](#) in class [Number](#)

**Returns:**

the value of this Float16 as an int after a narrowing primitive conversion

**See Java Language Specification:**

[5.1.3 Narrowing Primitive Conversion](#)

## longValue

```
public long longValue()
```

Returns value of this Float16 as a long after a narrowing primitive conversion.

**Specified by:**

[longValue](#) in class [Number](#)

**Returns:**

value of this Float16 as a long after a narrowing primitive conversion

**See Java Language Specification:**

[5.1.3 Narrowing Primitive Conversion](#)

## floatValue

```
public float floatValue()
```

Returns the value of this Float16 as a float after a widening primitive conversion.

**Specified by:**

[floatValue](#) in class [Number](#)

**API Note:**

This method corresponds to the `convertFormat` operation defined in IEEE 754.

**Returns:**

the value of this `Float16` as a `float` after a widening primitive conversion

**See Java Language Specification:**

[5.1.2 Widening Primitive Conversion](#)

**doubleValue**

```
public double doubleValue()
```

Returns the value of this `Float16` as a `double` after a widening primitive conversion.

**Specified by:**

`doubleValue` in class `Number`

**API Note:**

This method corresponds to the `convertFormat` operation defined in IEEE 754.

**Returns:**

the value of this `Float16` as a `double` after a widening primitive conversion

**See Java Language Specification:**

[5.1.2 Widening Primitive Conversion](#)

**float16ToRawShortBits**

```
public static short float16ToRawShortBits(Float16 f16)
```

Returns a representation of the specified floating-point value according to the IEEE 754 floating-point binary16 bit layout.

**Parameters:**

`f16` - a `Float16` floating-point number.

**Returns:**

the bits that represent the floating-point number.

**See Also:**

```
Float.floatToRawIntBits(float),  
Double.doubleToRawLongBits(double)
```

## shortBitsToFloat16

```
public static Float16 shortBitsToFloat16(short bits)
```

Returns the Float16 value corresponding to a given bit representation.

**Parameters:**

bits - any short integer.

**Returns:**

the Float16 floating-point value with the same bit pattern.

**See Also:**

[Float.intBitsToFloat\(int\)](#), [Double.longBitsToDouble\(long\)](#)

## compareTo

```
public int compareTo(Float16 anotherFloat16)
```

Compares two Float16 objects numerically. This method imposes a total order on Float16 objects with two differences compared to the incomplete order defined by the Java language numerical comparison operators (<, <=, ==, >=, >) on float and double values.

- A NaN is *unordered* with respect to other values and unequal to itself under the comparison operators. This method chooses to define Float16.NaN to be equal to itself and greater than all other Float16 values (including Float16.POSITIVE\_INFINITY).
- Positive zero and negative zero compare equal numerically, but are distinct and distinguishable values. This method chooses to define positive zero to be greater than negative zero.

**Specified by:**

[compareTo](#) in interface [Comparable<Float16>](#)

**Parameters:**

anotherFloat16 - the Float16 to be compared.

**Returns:**

the value 0 if anotherFloat16 is numerically equal to this Float16; a value less than 0 if this Float16 is numerically less than anotherFloat16; and a value greater than 0 if this Float16 is numerically greater than anotherFloat16.

**See Java Language Specification:**

[15.20.1 Numerical Comparison Operators <, <=, >, and >=](#)

**See Also:**

[Float.compareTo\(Float\)](#), [Double.compareTo\(Double\)](#)

**compare**

```
public static int compare(Float16 f1,  
                          Float16 f2)
```

Compares the two specified Float16 values.

**Parameters:**

f1 - the first Float16 to compare

f2 - the second Float16 to compare

**Returns:**

the value 0 if f1 is numerically equal to f2; a value less than 0 if f1 is numerically less than f2; and a value greater than 0 if f1 is numerically greater than f2.

**See Also:**

[Float.compare\(float, float\)](#),  
[Double.compare\(double, double\)](#)

**max**

```
public static Float16 max(Float16 a,  
                          Float16 b)
```

Returns the larger of two Float16 values.

**API Note:**

This method corresponds to the maximum operation defined in IEEE 754.

**Parameters:**

a - the first operand

b - the second operand

**Returns:**

the greater of a and b

**See Also:**

[BinaryOperator](#),  
[Math.max\(float, float\)](#),  
[Math.max\(double, double\)](#)

## min

```
public static Float16 min(Float16 a,  
                          Float16 b)
```

Returns the smaller of two Float16 values.

**API Note:**

This method corresponds to the minimum operation defined in IEEE 754.

**Parameters:**

a - the first operand

b - the second operand

**Returns:**

the smaller of a and b

**See Also:**

[BinaryOperator](#),  
[Math.min\(float, float\)](#),  
[Math.min\(double, double\)](#)

**add**

```
public static Float16 add(Float16 addend,  
                          Float16 augend)
```

Adds two `Float16` values together as per the `+` operator semantics using the round to nearest rounding policy. The handling of signed zeros, NaNs, infinities, and other special cases by this method is the same as for the handling of those cases by the built-in `+` operator for floating-point addition (JLS [15.18.2](#) ).

**API Note:**

This method corresponds to the addition operation defined in IEEE 754.

**Parameters:**

`addend` - the first operand

`augend` - the second operand

**Returns:**

the sum of the operands

**See *Java Language Specification*:**

[15.4 Floating-point Expressions](#)

**subtract**

```
public static Float16 subtract(Float16 minuend,  
                              Float16 subtrahend)
```

Subtracts two `Float16` values as per the `-` operator semantics using the round to nearest rounding policy. The handling of signed zeros, NaNs, infinities, and other

special cases by this method is the same as for the handling of those cases by the built-in `-` operator for floating-point subtraction (JLS 15.18.2 ).

**API Note:**

This method corresponds to the subtraction operation defined in IEEE 754.

**Parameters:**

`minuend` - the first operand

`subtrahend` - the second operand

**Returns:**

the difference of the operands

**See Java Language Specification:**

[15.4 Floating-point Expressions](#)

## multiply

```
public static Float16 multiply(Float16 multiplier,  
                               Float16 multiplicand)
```

Multiplies two `Float16` values as per the `*` operator semantics using the round to nearest rounding policy. The handling of signed zeros, NaNs, and infinities, other special cases by this method is the same as for the handling of those cases by the built-in `*` operator for floating-point multiplication (JLS 15.17.1 ).

**API Note:**

This method corresponds to the multiplication operation defined in IEEE 754.

**Parameters:**

`multiplier` - the first operand

`multiplicand` - the second operand

**Returns:**

the product of the operands

**See Java Language Specification:**



## 15.4 Floating-point Expressions

### divide

```
public static Float16 divide(Float16 dividend,
                             Float16 divisor)
```

Divides two `Float16` values as per the `/` operator semantics using the round to nearest rounding policy. The handling of signed zeros, NaNs, and infinities, other special cases by this method is the same as for the handling of those cases by the built-in `/` operator for floating-point division (JLS 15.17.2 ).

**API Note:**

This method corresponds to the division operation defined in IEEE 754.

**Parameters:**

`dividend` - the first operand

`divisor` - the second operand

**Returns:**

the quotient of the operands

**See *Java Language Specification*:**

[15.4 Floating-point Expressions](#)

### sqrt

```
public static Float16 sqrt(Float16 radicand)
```

Returns the square root of the operand. The square root is computed using the round to nearest rounding policy. The handling of zeros, NaN, infinities, and negative arguments by this method is analogous to the handling of those cases by `Math.sqrt(double)`.

**API Note:**

This method corresponds to the `squareRoot` operation defined in IEEE 754.

**Parameters:**

radicand - the argument to have its square root taken

**Returns:**

the square root of the operand

**See Also:**

[Math.sqrt\(double\)](#)

**fma**

```
public static Float16 fma(Float16 a,  
                          Float16 b,  
                          Float16 c)
```

Returns the fused multiply add of the three arguments; that is, returns the exact product of the first two arguments summed with the third argument and then rounded once to the nearest Float16. The handling of zeros, NaN, infinities, and other special cases by this method is analogous to the handling of those cases by [Math.fma\(float, float, float\)](#).

**API Note:**

This method corresponds to the fusedMultiplyAdd operation defined in IEEE 754.

**Parameters:**

a - a value

b - a value

c - a value

**Returns:**

$(a \times b + c)$  computed, as if with unlimited range and precision, and rounded once to the nearest Float16 value

**See Also:**

[Math.fma\(float, float, float\)](#),  
[Math.fma\(double, double, double\)](#)

## negate

```
public static Float16 negate(Float16 f16)
```

Returns the negation of the argument. Special cases:

- If the argument is zero, the result is a zero with the opposite sign as the argument.
- If the argument is infinite, the result is an infinity with the opposite sign as the argument.
- If the argument is a NaN, the result is a NaN.

### API Note:

This method corresponds to the negate operation defined in IEEE 754.

### Parameters:

f16 - the value to be negated

### Returns:

the negation of the argument

### See *Java Language Specification*:

[15.15.4 Unary Minus Operator](#) -

## abs

```
public static Float16 abs(Float16 f16)
```

Returns the absolute value of the argument. The handling of zeros, NaN, and infinities by this method is analogous to the handling of those cases by `Math.abs(float)`.

**Parameters:**

f16 - the argument whose absolute value is to be determined

**Returns:**

the absolute value of the argument

**See Also:**

`Math.abs(float)`, `Math.abs(double)`

## getExponent

```
public static int getExponent(Float16 f16)
```

Returns the unbiased exponent used in the representation of a `Float16`.

- If the argument is NaN or infinite, then the result is `MAX_EXPONENT + 1`.
- If the argument is zero or subnormal, then the result is `MIN_EXPONENT - 1`.

**API Note:**

This method is analogous to the `logB` operation defined in IEEE 754, but returns a different value on subnormal arguments.

**Parameters:**

f16 - a `Float16` value

**Returns:**

the unbiased exponent of the argument

**See Also:**

`Math.getExponent(float)`, `Math.getExponent(double)`

## ulp

```
public static Float16 ulp(Float16 f16)
```

Returns the size of an ulp of the argument. An ulp, unit in the last place, of a `Float16` value is the positive distance between this floating-point value and the `Float16` value next larger in magnitude. Note that for non-`NaN`  $x$ , `ulp(-x) == ulp(x)`.

Special Cases:

- If the argument is `NaN`, then the result is `NaN`.
- If the argument is positive or negative infinity, then the result is positive infinity.
- If the argument is positive or negative zero, then the result is `Float16.MIN_VALUE`.
- If the argument is  $\pm$ `Float16.MAX_VALUE`, then the result is equal to  $2^5$ , 32.0.

**Parameters:**

`f16` - the floating-point value whose ulp is to be returned

**Returns:**

the size of an ulp of the argument

## nextUp

```
public static Float16 nextUp(Float16 v)
```

Returns the floating-point value adjacent to `v` in the direction of positive infinity.

Special Cases:

- If the argument is `NaN`, the result is `NaN`.
- If the argument is positive infinity, the result is positive infinity.
- If the argument is zero, the result is `MIN_VALUE`

**API Note:**

This method corresponds to the `nextUp` operation defined in IEEE 754.

**Parameters:**

v - starting floating-point value

**Returns:**

The adjacent floating-point value closer to positive infinity.

**nextDown**

```
public static Float16 nextDown(Float16 v)
```

Returns the floating-point value adjacent to v in the direction of negative infinity.

Special Cases:

- If the argument is NaN, the result is NaN.
- If the argument is negative infinity, the result is negative infinity.
- If the argument is zero, the result is `-MIN_VALUE`

**API Note:**

This method corresponds to the nextDown operation defined in IEEE 754.

**Parameters:**

v - starting floating-point value

**Returns:**

The adjacent floating-point value closer to negative infinity.

**scalb**

```
public static Float16 scalb(Float16 v,  
                             int scaleFactor)
```

Returns  $v \times 2^{\text{scaleFactor}}$  rounded as if performed by a single correctly rounded floating-point multiply. If the exponent of the result is between `MIN_EXPONENT` and `MAX_EXPONENT`, the answer is calculated exactly. If the exponent of the result would be larger than `Float16.MAX_EXPONENT`, an infinity is returned. Note that if the result is subnormal, precision may be lost; that is, when `scalb(x, n)` is subnormal,

`scalb(scalb(x, n), -n)` may not equal `x`. When the result is non-NaN, the result has the same sign as `v`.

Special cases:

- If the first argument is NaN, NaN is returned.
- If the first argument is infinite, then an infinity of the same sign is returned.
- If the first argument is zero, then a zero of the same sign is returned.

**API Note:**

This method corresponds to the `scaleB` operation defined in IEEE 754.

**Parameters:**

`v` - number to be scaled by a power of two.

`scaleFactor` - power of 2 used to scale `v`

**Returns:**

$v \times 2^{\text{scaleFactor}}$

## copySign

```
public static Float16 copySign(Float16 magnitude,  
                               Float16 sign)
```

Returns the first floating-point argument with the sign of the second floating-point argument. This method does not require NaN sign arguments to be treated as positive values; implementations are permitted to treat some NaN arguments as positive and other NaN arguments as negative to allow greater performance.

**API Note:**

This method corresponds to the `copySign` operation defined in IEEE 754.

**Parameters:**

`magnitude` - the parameter providing the magnitude of the result

`sign` - the parameter providing the sign of the result

**Returns:**

a value with the magnitude of magnitude and the sign of sign.

## signum

```
public static Float16 signum(Float16 f)
```

Returns the signum function of the argument; zero if the argument is zero, 1.0 if the argument is greater than zero, -1.0 if the argument is less than zero.

Special Cases:

- If the argument is NaN, then the result is NaN.
- If the argument is positive zero or negative zero, then the result is the same as the argument.

### Parameters:

f - the floating-point value whose signum is to be returned

### Returns:

the signum function of the argument

---

[Report a bug](#) or [suggest an enhancement](#)

For further API reference and developer documentation see the [Java SE Documentation](#) , which contains more detailed, developer-targeted descriptions with conceptual overviews, definitions of terms, workarounds, and working code examples. [Other versions](#).

Java is a trademark or registered trademark of Oracle and/or its affiliates in the US and other countries. Copyright © 1993, 2024, Oracle and/or its affiliates, 500 Oracle Parkway, Redwood Shores, CA 94065 USA. All rights reserved. Use is subject to [license terms](#) and the [documentation redistribution policy](#) .

**DRAFT 24-internal-adhoc.darcy.open**