| | [**Overview**](http://java.sun.com/j2se/1.3/docs/api/overview-summary.html) | [**Package**](http://java.sun.com/j2se/1.3/docs/api/java/lang/package-summary.html) | **Class** | [**Use**](http://java.sun.com/j2se/1.3/docs/api/java/lang/class-use/Double.html) | [**Tree**](http://java.sun.com/j2se/1.3/docs/api/java/lang/package-tree.html) | [**Deprecated**](http://java.sun.com/j2se/1.3/docs/api/deprecated-list.html) | [**Index**](http://java.sun.com/j2se/1.3/docs/api/index-files/index-1.html) | [**Help**](http://java.sun.com/j2se/1.3/docs/api/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***JavaTM 2 Platform***  ***Std. Ed. v1.3.1*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Compiler.html)   [**NEXT CLASS**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Float.html) | [**FRAMES**](http://java.sun.com/j2se/1.3/docs/api/index.html)    [**NO FRAMES**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) |
| SUMMARY:  INNER | [FIELD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#field_summary) | [CONSTR](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#constructor_summary) | [METHOD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#method_summary) | DETAIL:  [FIELD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#field_detail) | [CONSTR](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#constructor_detail) | [METHOD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#method_detail) |

## **java.lang**

Class Double

[java.lang.Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html)  
 |  
 +--[java.lang.Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html)  
 |  
 +--**java.lang.Double**

**All Implemented Interfaces:** [Comparable](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html), [Serializable](http://java.sun.com/j2se/1.3/docs/api/java/io/Serializable.html)public final class **Double** extends [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) implements [Comparable](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html)

The Double class wraps a value of the primitive type double in an object. An object of type Double contains a single field whose type is double.

In addition, this class provides several methods for converting a double to a String and a String to a double, as well as other constants and methods useful when dealing with a double.

**Since:**  JDK1.0 **See Also:**  [Serialized Form](http://java.sun.com/j2se/1.3/docs/api/serialized-form.html#java.lang.Double)

| **Field Summary** | |
| --- | --- |
| static double | [**MAX\_VALUE**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#MAX_VALUE)            The largest positive finite value of type double. |
| static double | [**MIN\_VALUE**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#MIN_VALUE)            The smallest positive value of type double. |
| static double | [**NaN**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#NaN)            A Not-a-Number (NaN) value of type double. |
| static double | [**NEGATIVE\_INFINITY**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#NEGATIVE_INFINITY)            The negative infinity of type double. |
| static double | [**POSITIVE\_INFINITY**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#POSITIVE_INFINITY)            The positive infinity of type double. |
| static [Class](http://java.sun.com/j2se/1.3/docs/api/java/lang/Class.html) | [**TYPE**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#TYPE)            The Class object representing the primitive type double. |

| **Constructor Summary** | |
| --- | --- |
| [**Double**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#Double(double))(double value)            Constructs a newly allocated Double object that represents the primitive double argument. |
| [**Double**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#Double(java.lang.String))([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)            Constructs a newly allocated Double object that represents the floating- point value of type double represented by the string. |

| **Method Summary** | |
| --- | --- |
| byte | [**byteValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#byteValue())()            Returns the value of this Double as a byte (by casting to a byte). |
| int | [**compareTo**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#compareTo(java.lang.Double))([Double](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) anotherDouble)            Compares two Doubles numerically. |
| int | [**compareTo**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#compareTo(java.lang.Object))([Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) o)            Compares this Double to another Object. |
| static long | [**doubleToLongBits**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#doubleToLongBits(double))(double value)            Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. |
| static long | [**doubleToRawLongBits**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#doubleToRawLongBits(double))(double value)            Returns a representation of the specified floating-point value according to the IEEE 754 floating-point "double format" bit layout. |
| double | [**doubleValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#doubleValue())()            Returns the double value of this Double. |
| boolean | [**equals**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#equals(java.lang.Object))([Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) obj)            Compares this object against the specified object. |
| float | [**floatValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#floatValue())()            Returns the float value of this Double. |
| int | [**hashCode**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#hashCode())()            Returns a hashcode for this Double object. |
| int | [**intValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#intValue())()            Returns the integer value of this Double (by casting to an int). |
| boolean | [**isInfinite**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#isInfinite())()            Returns true if this Double value is infinitely large in magnitude. |
| static boolean | [**isInfinite**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#isInfinite(double))(double v)            Returns true if the specified number is infinitely large in magnitude. |
| boolean | [**isNaN**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#isNaN())()            Returns true if this Double value is the special Not-a-Number (NaN) value. |
| static boolean | [**isNaN**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#isNaN(double))(double v)            Returns true if the specified number is the special Not-a-Number (NaN) value. |
| static double | [**longBitsToDouble**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#longBitsToDouble(long))(long bits)            Returns the double-float corresponding to a given bit represention. |
| long | [**longValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#longValue())()            Returns the long value of this Double (by casting to a long). |
| static double | [**parseDouble**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#parseDouble(java.lang.String))([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)            Returns a new double initialized to the value represented by the specified String, as performed by the valueOf method of class Double. |
| short | [**shortValue**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#shortValue())()            Returns the value of this Double as a short (by casting to a short). |
| [String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) | [**toString**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#toString())()            Returns a String representation of this Double object. |
| static [String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) | [**toString**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#toString(double))(double d)            Creates a string representation of the double argument. |
| static [Double](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) | [**valueOf**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#valueOf(java.lang.String))([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)            Returns a new Double object initialized to the value represented by the specified string. |

| **Methods inherited from class java.lang.**[**Object**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) |
| --- |
| [clone](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#clone()), [finalize](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#finalize()), [getClass](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#getClass()), [notify](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#notify()), [notifyAll](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#notifyAll()), [wait](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#wait()), [wait](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#wait(long)), [wait](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#wait(long,%20int)) |

| **Field Detail** |
| --- |

### POSITIVE\_INFINITY

public static final double **POSITIVE\_INFINITY**

The positive infinity of type double. It is equal to the value returned by Double.longBitsToDouble(0x7ff0000000000000L).

### NEGATIVE\_INFINITY

public static final double **NEGATIVE\_INFINITY**

The negative infinity of type double. It is equal to the value returned by Double.longBitsToDouble(0xfff0000000000000L).

### NaN

public static final double **NaN**

A Not-a-Number (NaN) value of type double. It is equal to the value returned by Double.longBitsToDouble(0x7ff8000000000000L).

### MAX\_VALUE

public static final double **MAX\_VALUE**

The largest positive finite value of type double. It is equal to the returned by:

Double.longBitsToDouble(0x7fefffffffffffffL)

### MIN\_VALUE

public static final double **MIN\_VALUE**

The smallest positive value of type double. It is equal to the value returned by Double.longBitsToDouble(0x1L).

### TYPE

public static final [Class](http://java.sun.com/j2se/1.3/docs/api/java/lang/Class.html) **TYPE**

The Class object representing the primitive type double. **Since:**  JDK1.1

| **Constructor Detail** |
| --- |

### Double

public **Double**(double value)

Constructs a newly allocated Double object that represents the primitive double argument. **Parameters:** value - the value to be represented by the Double.

### Double

public **Double**([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)  
 throws [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html)

Constructs a newly allocated Double object that represents the floating- point value of type double represented by the string. The string is converted to a double value as if by the valueOf method. **Parameters:** s - a string to be converted to a Double. **Throws:** [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html) - if the string does not contain a parsable number. **See Also:**  [valueOf(java.lang.String)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#valueOf(java.lang.String))

| **Method Detail** |
| --- |

### toString

public static [String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) **toString**(double d)

Creates a string representation of the double argument. All characters mentioned below are ASCII characters.

* If the argument is NaN, the result is the string "NaN".
* Otherwise, the result is a string that represents the sign and magnitude (absolute value) of the argument. If the sign is negative, the first character of the result is '-' ('-'); if the sign is positive, no sign character appears in the result. As for the magnitude *m*:
* If *m* is infinity, it is represented by the characters "Infinity"; thus, positive infinity produces the result "Infinity" and negative infinity produces the result "-Infinity".
* If *m* is zero, it is represented by the characters "0.0"; thus, negative zero produces the result "-0.0" and positive zero produces the result "0.0".
* If *m* is greater than or equal to 10-3 but less than 107, then it is represented as the integer part of *m*, in decimal form with no leading zeroes, followed by '.' (.), followed by one or more decimal digits representing the fractional part of *m*.
* If *m* is less than 10-3 or not less than 107, then it is represented in so-called "computerized scientific notation." Let *n* be the unique integer such that 10n<=*m*<10n+1; then let *a* be the mathematically exact quotient of *m* and 10n so that 1<=*a*<10. The magnitude is then represented as the integer part of *a*, as a single decimal digit, followed by '.' (.), followed by decimal digits representing the fractional part of *a*, followed by the letter 'E' (E), followed by a representation of *n* as a decimal integer, as produced by the method [Integer.toString(int)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Integer.html#toString(int)).

How many digits must be printed for the fractional part of *m* or *a*? There must be at least one digit to represent the fractional part, and beyond that as many, but only as many, more digits as are needed to uniquely distinguish the argument value from adjacent values of type double. That is, suppose that *x* is the exact mathematical value represented by the decimal representation produced by this method for a finite nonzero argument *d*. Then *d* must be the double value nearest to *x*; or if two double values are equally close to *x*, then *d* must be one of them and the least significant bit of the significand of *d* must be 0.

**Parameters:** d - the double to be converted. **Returns:** a string representation of the argument.

### valueOf

public static [Double](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) **valueOf**([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)  
 throws [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html)

Returns a new Double object initialized to the value represented by the specified string. The string s is interpreted as the representation of a floating-point value and a Double object representing that value is created and returned.

If s is null, then a NullPointerException is thrown.

Leading and trailing whitespace characters in s are ignored. The rest of s should constitute a *FloatValue* as described by the lexical rule:

*FloatValue:  
   
 Signopt FloatingPointLiteral*

where *Sign* and *FloatingPointLiteral* are as defined in �3.10.2 of the [Java Language Specification](http://java.sun.com/docs/books/jls/html/). If it does not have the form of a *FloatValue*, then a NumberFormatException is thrown. Otherwise, it is regarded as representing an exact decimal value in the usual "computerized scientific notation"; this exact decimal value is then conceptually converted to an "infinitely precise" binary value that is then rounded to type double by the usual round-to-nearest rule of IEEE 754 floating-point arithmetic. Finally, a new object of class Double is created to represent the double value. **Parameters:** s - the string to be parsed. **Returns:** a newly constructed Double initialized to the value represented by the string argument. **Throws:** [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html) - if the string does not contain a parsable number.

### parseDouble

public static double **parseDouble**([String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) s)  
 throws [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html)

Returns a new double initialized to the value represented by the specified String, as performed by the valueOf method of class Double. **Parameters:** s - the string to be parsed. **Returns:** the double value represented by the string argument. **Throws:** [NumberFormatException](http://java.sun.com/j2se/1.3/docs/api/java/lang/NumberFormatException.html) - if the string does not contain a parsable double. **Since:**  1.2 **See Also:**  [valueOf(String)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#valueOf(java.lang.String))

### isNaN

public static boolean **isNaN**(double v)

Returns true if the specified number is the special Not-a-Number (NaN) value. **Parameters:** v - the value to be tested. **Returns:** true if the value of the argument is NaN; false otherwise.

### isInfinite

public static boolean **isInfinite**(double v)

Returns true if the specified number is infinitely large in magnitude. **Parameters:** v - the value to be tested. **Returns:** true if the value of the argument is positive infinity or negative infinity; false otherwise.

### isNaN

public boolean **isNaN**()

Returns true if this Double value is the special Not-a-Number (NaN) value. **Returns:** true if the value represented by this object is NaN; false otherwise.

### isInfinite

public boolean **isInfinite**()

Returns true if this Double value is infinitely large in magnitude. **Returns:** true if the value represented by this object is positive infinity or negative infinity; false otherwise.

### toString

public [String](http://java.sun.com/j2se/1.3/docs/api/java/lang/String.html) **toString**()

Returns a String representation of this Double object. The primitive double value represented by this object is converted to a string exactly as if by the method toString of one argument. **Overrides:** [toString](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#toString()) in class [Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) **Returns:** a String representation of this object. **See Also:**  [toString(double)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#toString(double))

### byteValue

public byte **byteValue**()

Returns the value of this Double as a byte (by casting to a byte). **Overrides:** [byteValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#byteValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Since:**  JDK1.1

### shortValue

public short **shortValue**()

Returns the value of this Double as a short (by casting to a short). **Overrides:** [shortValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#shortValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Since:**  JDK1.1

### intValue

public int **intValue**()

Returns the integer value of this Double (by casting to an int). **Overrides:** [intValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#intValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Returns:** the double value represented by this object is converted to type int and the result of the conversion is returned.

### longValue

public long **longValue**()

Returns the long value of this Double (by casting to a long). **Overrides:** [longValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#longValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Returns:** the double value represented by this object is converted to type long and the result of the conversion is returned.

### floatValue

public float **floatValue**()

Returns the float value of this Double. **Overrides:** [floatValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#floatValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Returns:** the double value represented by this object is converted to type float and the result of the conversion is returned. **Since:**  JDK1.0

### doubleValue

public double **doubleValue**()

Returns the double value of this Double. **Overrides:** [doubleValue](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html#doubleValue()) in class [Number](http://java.sun.com/j2se/1.3/docs/api/java/lang/Number.html) **Returns:** the double value represented by this object.

### hashCode

public int **hashCode**()

Returns a hashcode for this Double object. The result is the exclusive OR of the two halves of the long integer bit representation, exactly as produced by the method [doubleToLongBits(double)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#doubleToLongBits(double)), of the primitive double value represented by this Double object. That is, the hashcode is the value of the expression:

(int)(v^(v>>>32))

where v is defined by:

long v = Double.doubleToLongBits(this.doubleValue());

**Overrides:** [hashCode](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#hashCode()) in class [Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) **Returns:** a hash code value for this object.

### equals

public boolean **equals**([Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) obj)

Compares this object against the specified object. The result is true if and only if the argument is not null and is a Double object that represents a double that has the identical bit pattern to the bit pattern of the double represented by this object. For this purpose, two double values are considered to be the same if and only if the method [doubleToLongBits(double)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#doubleToLongBits(double)) returns the same long value when applied to each.

Note that in most cases, for two instances of class Double, d1 and d2, the value of d1.equals(d2) is true if and only if

d1.doubleValue() == d2.doubleValue()

also has the value true. However, there are two exceptions:

* If d1 and d2 both represent Double.NaN, then the equals method returns true, even though Double.NaN==Double.NaN has the value false.
* If d1 represents +0.0 while d2 represents -0.0, or vice versa, the equal test has the value false, even though +0.0==-0.0 has the value true. This allows hashtables to operate properly.

**Overrides:** [equals](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) **Parameters:** obj - the object to compare with. **Returns:** true if the objects are the same; false otherwise.

### doubleToLongBits

public static long **doubleToLongBits**(double value)

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

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L.

If the argument is negative infinity, the result is 0xfff0000000000000L.

If the argument is NaN, the result is 0x7ff8000000000000L.

In all cases, the result is a long integer that, when given to the [longBitsToDouble(long)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#longBitsToDouble(long)) method, will produce a floating-point value equal to the argument to doubleToLongBits.

**Parameters:** value - a double precision floating-point number. **Returns:** the bits that represent the floating-point number.

### doubleToRawLongBits

public static long **doubleToRawLongBits**(double value)

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

Bit 63 (the bit that is selected by the mask 0x8000000000000000L) represents the sign of the floating-point number. Bits 62-52 (the bits that are selected by the mask 0x7ff0000000000000L) represent the exponent. Bits 51-0 (the bits that are selected by the mask 0x000fffffffffffffL) represent the significand (sometimes called the mantissa) of the floating-point number.

If the argument is positive infinity, the result is 0x7ff0000000000000L.

If the argument is negative infinity, the result is 0xfff0000000000000L.

If the argument is NaN, the result is the long integer representing the actual NaN value. Unlike the doubleToLongBits method, doubleToRawLongBits does not collapse NaN values.

In all cases, the result is a long integer that, when given to the [longBitsToDouble(long)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#longBitsToDouble(long)) method, will produce a floating-point value equal to the argument to doubleToRawLongBits.

**Parameters:** value - a double precision floating-point number. **Returns:** the bits that represent the floating-point number.

### longBitsToDouble

public static double **longBitsToDouble**(long bits)

Returns the double-float corresponding to a given bit represention. The argument is considered to be a representation of a floating-point value according to the IEEE 754 floating-point "double precision" bit layout. That floating-point value is returned as the result.

If the argument is 0x7ff0000000000000L, the result is positive infinity.

If the argument is 0xfff0000000000000L, the result is negative infinity.

If the argument is any value in the range 0x7ff0000000000001L through 0x7fffffffffffffffL or in the range 0xfff0000000000001L through 0xffffffffffffffffL, the result is NaN. All IEEE 754 NaN values of type double are, in effect, lumped together by the Java programming language into a single value called NaN. Distinct values of NaN are only accessible by use of the Double.doubleToRawLongBits method.

In all other cases, let *s*, *e*, and *m* be three values that can be computed from the argument:

int s = ((bits >> 63) == 0) ? 1 : -1;  
 int e = (int)((bits >> 52) & 0x7ffL);  
 long m = (e == 0) ?  
 (bits & 0xfffffffffffffL) << 1 :  
 (bits & 0xfffffffffffffL) | 0x10000000000000L;

Then the floating-point result equals the value of the mathematical expression *s*�*m*�2e-1075. **Parameters:** bits - any long integer. **Returns:** the double floating-point value with the same bit pattern.

### compareTo

public int **compareTo**([Double](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) anotherDouble)

Compares two Doubles numerically. There are two ways in which comparisons performed by this method differ from those performed by the Java language numerical comparison operators (<, <=, ==, >= >) when applied to primitive doubles:

* Double.NaN is considered by this method to be equal to itself and greater than all other double values (including Double.POSITIVE\_INFINITY).
* 0.0d is considered by this method to be greater than -0.0d.

This ensures that Double.compareTo(Object) (which inherits its behavior from this method) obeys the general contract for Comparable.compareTo, and that the *natural order* on Doubles is *total*. **Parameters:** anotherDouble - the Double to be compared. **Returns:** the value 0 if anotherDouble is numerically equal to this Double; a value less than 0 if this Double is numerically less than anotherDouble; and a value greater than 0 if this Double is numerically greater than anotherDouble. **Since:**  1.2 **See Also:**  [Comparable.compareTo(Object)](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html#compareTo(java.lang.Object))

### compareTo

public int **compareTo**([Object](http://java.sun.com/j2se/1.3/docs/api/java/lang/Object.html) o)

Compares this Double to another Object. If the Object is a Double, this function behaves like compareTo(Double). Otherwise, it throws a ClassCastException (as Doubles are comparable only to other Doubles). **Specified by:**  [compareTo](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html#compareTo(java.lang.Object)) in interface [Comparable](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html) **Parameters:** o - the Object to be compared. **Returns:** the value 0 if the argument is a Double numerically equal to this Double; a value less than 0 if the argument is a Double numerically greater than this Double; and a value greater than 0 if the argument is a Double numerically less than this Double. **Throws:** ClassCastException - if the argument is not a Double. **Since:**  1.2 **See Also:**  [Comparable](http://java.sun.com/j2se/1.3/docs/api/java/lang/Comparable.html)

| | [**Overview**](http://java.sun.com/j2se/1.3/docs/api/overview-summary.html) | [**Package**](http://java.sun.com/j2se/1.3/docs/api/java/lang/package-summary.html) | **Class** | [**Use**](http://java.sun.com/j2se/1.3/docs/api/java/lang/class-use/Double.html) | [**Tree**](http://java.sun.com/j2se/1.3/docs/api/java/lang/package-tree.html) | [**Deprecated**](http://java.sun.com/j2se/1.3/docs/api/deprecated-list.html) | [**Index**](http://java.sun.com/j2se/1.3/docs/api/index-files/index-1.html) | [**Help**](http://java.sun.com/j2se/1.3/docs/api/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***JavaTM 2 Platform***  ***Std. Ed. v1.3.1*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Compiler.html)   [**NEXT CLASS**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Float.html) | [**FRAMES**](http://java.sun.com/j2se/1.3/docs/api/index.html)    [**NO FRAMES**](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html) |
| SUMMARY:  INNER | [FIELD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#field_summary) | [CONSTR](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#constructor_summary) | [METHOD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#method_summary) | DETAIL:  [FIELD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#field_detail) | [CONSTR](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#constructor_detail) | [METHOD](http://java.sun.com/j2se/1.3/docs/api/java/lang/Double.html#method_detail) |

[Submit a bug or feature](http://java.sun.com/cgi-bin/bugreport.cgi)

For further API reference and developer documentation, see [Java 2 SDK SE Developer Documentation](http://java.sun.com/products/jdk/1.3/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Java, Java 2D, and JDBC are trademarks or registered trademarks of Sun Microsystems, Inc. in the US and other countries.

Copyright 1993-2001 Sun Microsystems, Inc. 901 San Antonio Road

Palo Alto, California, 94303, U.S.A. All Rights Reserved.