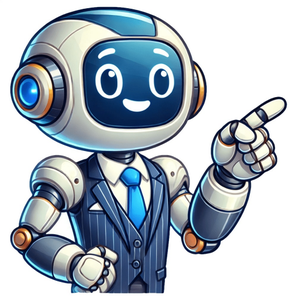
Continue



Math ceil java

21 Mar 2025 | 2 min readThe java.lang.Math.ceil () is used to find the smallest integer value that is greater than or equal to the argument or mathematical integer.Syntaxpublic static double ceil(double x) x= a value This method returns smallest floating-point value that is greater than or equal to the argument and is equal to a mathematical integer. If the argument is positive or negative double value, this method will return the ceil value.If the argument is NaN, this method will return same argument.If the argument is Infinity, this method will return Infinity with the same sign as the argument.If the argument is positive or negative Zero, this method will return Zero with same sign as the argument.If the argument is less than Zero but greater than -1.0, this method will return Negative Zero as output.public class CeilExample1 { public static void main(String[] args) { double x = 83.56; // Input positive value, Output ceil value of x System.out.println(Math.ceil(x)); } } Output:Example 2public class CeilExample2 { public static void main(String[] args) { double x = -94.73; // Input negative value, Output ceil value of x System.out.println(Math.ceil(x)); } } Output:Example 3public class CeilExample3 { public static void main(String[] args) { double x = -1.0 / 0; // Input negative infinity, Output negative infinity System.out.println(Math.ceil(x)); } } Output:Example 4public class CeilExample4 { public static void main(String[] args) { double x = 0.0; // Input positive zero, Output positive zero System.out.println(Math.ceil(x)); } } Output:Example 5public class CeilExample5 { public static void main(String[] args) { double x = -0.25; // Input less than zero but greater than -1.0, Output negative zero System.out.println(Math.ceil(x)); } } Output: Next TopicJava Math The Java Math ceil(double a) returns the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Special cases − If the argument value is already equal to a mathematical integer, then the result is the same as the argument. If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument. If the argument value is less than zero but greater than -1.0, then the result is negative zero. Note that the value of Math.ceil(x) is exactly the value of -Math.floor(-x). Following is the declaration for java.lang.Math.ceil() method public static double ceil(double a) Parameters a − a value. Return Value This method returns the smallest (closest to negative infinity) floating-point value that is greater than or equal to the argument and is equal to a mathematical integer. Exception NA Getting Smallest Value Greater Than or Equal to a Positive Number Example The following example shows the usage of Math ceil() method. package com.tutorialspoint; public class MathDemo { public static void main(String[] args) { // get a double number double x = 10.7; // print the ceil of the number System.out.println("Math.ceil(" + x + ")=" + Math.ceil(x)); } } Output Let us compile and run the above program, this will produce the following result − Math.ceil(10.7)=11.0 Getting Smallest Value Greater Than or Equal to Zero Example The following example shows the usage of Math ceil() method of zero value. package com.tutorialspoint; public class MathDemo { public static void main(String[] args) { // get a double number double x = 0.0; // print the ceil of the number System.out.println("Math.ceil(" + x + ")=" + Math.ceil(x)); } } Output Let us compile and run the above program, this will produce the following result − Math.ceil(-0.0)=0.0 Getting Smallest Value Greater Than or Equal to a Negative Number Example The following example shows the usage of Math ceil() method of a negative number. package com.tutorialspoint; public class MathDemo { public static void main(String[] args) { // get a double number double x = -10.7; // print the ceil of the number System.out.println("Math.ceil(" + x + ")=" + Math.ceil(x)); } } Output Let us compile and run the above program, this will produce the following result − Math.ceil(-10.7)=-10.0 java_lang_math.htm In this example, we will learn about the Math.ceil Java method. Java math class contains methods for performing basic numeric operations such as the elementary exponential, logarithm, square root, and trigonometric functions. Some of the most important of Math class methods are min(), max(), avg(), sin(), cos(). You could take a look at all methods of Math class in the java doc.But today we will be familiar with the ceil method of Math class.Math.ceil method always rounds a number up to the next largest double or in other words returns the smallest double value that is greater than or equal to the argument. For example: System.out.println(Math.ceil(1.2)); // expected result: 2.0 System.out.println(Math.ceil(2.0001)); // expected result: 3.0 System.out.println(Math.ceil(-2.01)); // expected result: -2.0 The following snippet shows the syntax of Math.ceil method. public static double ceil(double a) If the input is an infinity then the result is infinity.If the input is a positive zero then the result is a positive zero.If the input is negative zero then the result is negative zero.If the input is less than zero but greater than -1 then the result is negative zero.public class CeilExample { public static void main(String[] args) { // Integer number System.out.println(Math.ceil(2)); System.out.println(Math.ceil(1.2)); System.out.println(Math.ceil(2.001)); // Infinity example System.out.println(Math.ceil(1.0/0)); // Positive zero System.out.println(Math.ceil(0)); // Negative zero System.out.println(Math.ceil(-0.0)); // Negative number less than zero but greater than -1 System.out.println(Math.ceil(-0.001)); // Negative number System.out.println(Math.ceil(-1.02)); } }After running the above code in any IDE of your choice you'll receive the following output: 2.0 2.0 3.0 Infinity 0.0 -0.0 -0.0 -1.0 In this article we reviewed the ceil() method from java.lang.Math class. Math.ceil method always rounds a number up to the next largest double.That was a Math.ceil Java Example.Download You can download the full source code of this example here: Math.ceil Java Example Related Articles Premium Read: Access my best content on Medium member-only articles — deep dives into Java, Spring Boot, Microservices, backend architecture, interview preparation, career advice, and industry-standard best practices. Some premium posts are free to read — no account needed. Follow me on Medium to stay updated and support my writing. Top 10 Udemy Courses (Huge Discount): Explore My Udemy Courses — Learn through real-time, project-based development. ▶ Subscribe to My YouTube Channel (172K+ subscribers): Java Guides on YouTube The Math.ceil() method in Java is used to return the smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Table of Contents Introduction ceil() Method Syntax Understanding ceil() Examples Basic Usage Using ceil() with Different Values Real-World Use Case Conclusion Introduction The Math.ceil() method is a part of the Math class in Java and is used to round a number up to the nearest integer value. The returned value is of type double and represents the smallest integer greater than or equal to the input value. ceil() Method Syntax The syntax for the ceil() method is as follows: public static double ceil(double a) Parameters: a: The value to be rounded up. Returns: The smallest (closest to negative infinity) double value that is greater than or equal to the argument and is equal to a mathematical integer. Understanding ceil() The Math.ceil() method calculates the ceiling of a given value, which means it rounds the value up to the nearest integer. This method is useful when you need to ensure that a value is rounded up, regardless of its fractional part. Examples Basic Usage To demonstrate the basic usage of ceil(), we will calculate the ceiling of a few values. Example public class CeilExample { public static void main(String[] args) { double value1 = 2.3; double value2 = -2.3; double value3 = 3.0; double result1 = Math.ceil(value1); double result2 = Math.ceil(value2); double result3 = Math.ceil(value3); System.out.println("Ceiling of " + value1 + " is " + result1); System.out.println("Ceiling of " + value2 + " is " + result2); System.out.println("Ceiling of " + value3 + " is " + result3); } } Output: Ceiling of 2.3 is 3.0 Ceiling of -2.3 is -2.0 Ceiling of 3.0 is 3.0 Using ceil() with Different Values You can use the ceil() method with various values to calculate their ceilings. Example public class CeilDifferentValuesExample { public static void main(String[] args) { double[] values = {1.1, 1.5, 1.9, -1.1, -1.5, -1.9, 0.0}; for (double value : values) { double result = Math.ceil(value); System.out.println("Ceiling of " + value + " is " + result); } } } Output: Ceiling of 1.1 is 2.0 Ceiling of 1.5 is 2.0 Ceiling of 1.9 is 2.0 Ceiling of -1.1 is -1.0 Ceiling of -1.5 is -1.0 Ceiling of -1.9 is -1.0 Ceiling of 0.0 is 0.0 Real-World Use Case Rounding Up Prices In real-world scenarios, the Math.ceil() method can be used to round up prices to the nearest whole number. This is useful in financial applications where you need to ensure that prices are rounded up to avoid fractional values. Example public class RoundUpPriceExample { public static void main(String[] args) { double price = 19.95; double roundedPrice = Math.ceil(price); System.out.println("The rounded up price is $" + roundedPrice); } } Output: The rounded up price is $20.0 Conclusion The Math.ceil() method in Java provides a way to round a given value up to the nearest integer. By understanding how to use this method, you can perform various rounding operations and solve problems that require values to be rounded up. Whether you are working with simple rounding tasks or complex financial calculations, the ceil() method offers a reliable tool for ensuring that values are rounded up correctly. ❮ Math Methods Round numbers up to the nearest integer: System.out.println(Math.ceil(0.60)); System.out.println(Math.ceil(0.40)); System.out.println(Math.ceil(5)); System.out.println(Math.ceil(5.1)); System.out.println(Math.ceil(-5.1)); System.out.println(Math.ceil(-5.9)); Try it Yourself » Definition and Usage The ceil() method rounds a number UP to the nearest integer. Tip: To round a number DOWN to the nearest integer, look at the floor() method. Tip: To round a number to the nearest integer in either direction, look at the round() method. Syntax public static double ceil(double number) Parameter Values Parameter Description number Required. A number to round up. Technical Details Returns: A double value representing the nearest integer greater or equal to a number. Java version: Any ❮ Math Methods The java.lang.Math.ceil() returns the double value that is greater than or equal to the argument and is equal to the nearest mathematical integer. Note: If the argument is Integer, then the result is Integer.If the argument is NaN or an infinity or positive zero or negative zero, then the result is the same as the argument.If the argument value is less than zero but greater than -1.0, then the result is negative zero. Syntax: public static double ceil(double a) a : the argument whose ceil value is to be determined Returns : This method returns the double value that is greater than or equal to the argument and is equal to the nearest mathematical integer. Example 01:To show working of java.lang.Math.ceil() method. java // Java program to demonstrate working // of java.lang.Math.ceil() method import java.lang.Math; class Gfg { // driver code public static void main(String args[]) { double a = 4.3; double b = 1.0 / 0; double c = 0.0; double d = -0.0; double e = -0.12; System.out.println(Math.ceil(a)); // Input Infinity, Output Infinity System.out.println(Math.ceil(b)); // Input Positive Zero, Output Positive Zero System.out.println(Math.ceil(c)); // Input Negative Zero, Output Negative Zero System.out.println(Math.ceil(d)); // Input less than zero but greater than -1.0 // Output Negative zero System.out.println(Math.ceil(e)); } } Output: 5.0 Infinity 0.0 -0.0 -0.0 Example 02: To show the working of ceil() with a positive double value Java import java.io.*; class GFG { public static void main (String[] args) { double number = 3.5; double result = Math.ceil(number); System.out.println(result); // Output: 4.0 } } Output : 4.0 The ceil() method rounds the specified double value upward and returns it. The rounded value will be equal to the mathematical integer. That is, the value 3.24 will be rounded to 4.0 which is equal to integer 4. Example class Main { public static void main(String[] args) { double a = 3.24; System.out.println(Math.ceil(a)); } } // Output: 4.0 The syntax of the ceil() method is: Math.ceil(double value) Here, ceil() is a static method. Hence, we are accessing the method using the class name, Math. ceil() Parameters The ceil() method takes a single parameter. value - number which is to be rounded upward ceil() Return Value returns the rounded value that is equal to the mathematical integer Note: The returned value will be the smallest value that is greater than or equal to the specified argument. Example: Java Math.ceil() class Main { public static void main(String[] args) { // Math.ceil() method // value greater than 5 after decimal double a = 1.878; System.out.println(Math.ceil(a)); // 2.0 // value equals to 5 after decimal double b = 1.5; System.out.println(Math.ceil(b)); // 2.0 // value less than 5 after decimal double c = 1.34; System.out.println(Math.ceil(c)); // 2.0 } } Also Read: Java Math.floor() Java Math.round() In Java, the process of rounding numbers is a frequently performed operation in a range of applications, spanning from mathematical computations to formatting output for presentation. Let us delve into a practical approach to exploring Java Math Ceil, Floor and Round methods.1. Understanding Math.ceil(), Math.floor(), and Math.round() in JavaIn Java, the Math class provides several methods for rounding numbers to different precision levels. Three commonly used methods are Math.ceil(), Math.floor(), and Math.round().1.1 Math.ceil()The Math.ceil() method returns the smallest integer greater than or equal to the specified numeric value. It effectively rounds up the number to the nearest whole number, regardless of the decimal part.double result = Math.ceil(7.25); // Result: 8.0 1.2 Math.floor()The Math.floor() method, on the other hand, returns the largest integer less than or equal to the specified numeric value. It effectively rounds down the number to the nearest whole number, ignoring the decimal part.double result = Math.floor(7.75); // Result: 7.0 1.3 Math.round()The Math.round() method rounds a floating-point value to the nearest integer. If the decimal part is .5 or greater, it rounds up; otherwise, it rounds down.long result = Math.round(7.5); // Result: 8 2. Using Math.ceil() for "Rounding Up" a NumberThe Math.ceil() method in Java is commonly used for rounding up a number to the nearest integer or, more specifically, to the smallest integer greater than or equal to the original value. Here's an example:package com.javacodegeek; public class RoundingUpExample { public static void main(String[] args) { double originalNumber = 6.75; // Using Math.ceil() to round up the number double roundedUpNumber = Math.ceil(originalNumber); // Displaying the results System.out.println("Original Number: " + originalNumber); System.out.println("Rounded Up Number: " + roundedUpNumber); } } In this example, the Math.ceil() method is applied to originalNumber, which is 6.75. The result is then stored in the variable roundedUpNumber. When you run this program, you'll get output similar to the following:Original Number: 6.75 Rounded Up Number: 7.0 3. Using Math.floor() for "Rounding Down" a NumberThe Math.floor() method in Java is commonly used for rounding down a number to the nearest integer or, more precisely, to the largest integer less than or equal to the original value. Here's an example:package com.javacodegeek; public class RoundingDownExample { public static void main(String[] args) { double originalNumber = 8.25; // Using Math.floor() to round down the number double roundedDownNumber = Math.floor(originalNumber); // Displaying the results System.out.println("Original Number: " + originalNumber); System.out.println("Rounded Down Number: " + roundedDownNumber); } } In this example, the Math.floor() method is applied to originalNumber, which is 8.25. The result is then stored in the variable roundedDownNumber. When you run this program, you'll get output similar to the following:Original Number: 8.25 Rounded Down Number: 8.0 4. Using Math.round() for "Rounding to the Nearest Integer"The Math.round() method in Java is commonly used for rounding a number to the nearest integer. It uses the "round half to even" strategy, also known as "bankers' rounding," where values equidistant to the two nearest integers are rounded to the even integer. Here's an example:package com.javacodegeek; public class RoundingToNearestIntegerExample { public static void main(String[] args) { double originalNumber = 5.75; // Using Math.round() to round to the nearest integer long roundedNumber = Math.round(originalNumber); // Displaying the results System.out.println("Original Number: " + originalNumber); System.out.println("Rounded Number: " + roundedNumber); } } In this example, the Math.round() method is applied to originalNumber, which is 5.75. The result is then stored in the variable roundedNumber. When you run this program, you'll get output similar to the following:Original Number: 5.75 Rounded Number: 6 5. ConclusionIn conclusion, when working with numerical values in Java, the Math class provides three essential methods for rounding: Math.ceil() for rounding up to the nearest integer, Math.floor() for rounding down to the nearest integer, and Math.round() for rounding to the nearest integer using the "round half to even" strategy. These methods are invaluable in a variety of applications, from mathematical calculations to formatting output for display. Whether adjusting data for specific requirements or handling mathematical operations that demand integer values, these rounding methods offer precise control over how numbers are processed in Java. Depending on the context and the desired outcome, developers can choose the appropriate method to ensure accurate and efficient handling of numerical data in their applications. Java has had several advanced usage application including working with complex calculations in physics, architecture/designing of structures, working with Maps and corresponding latitudes/longitudes, etc. All such applications require using complex calculations/equations that are tedious to perform manually. Programmatically, such calculations would involve usage of logarithms, trigonometry, exponential equations, etc. Now, you cannot have all the log or trigonometry tables hard-coded somewhere in your application or data. The data would be enormous and complex to maintain. Java provides a very useful class for this purpose. It is the Math java class (java.lang.Math). This class provides methods for performing the operations like exponential, logarithm, roots and trigonometric equations too. Let us have a look at the methods provided by the Java Math class. The two most fundamental elements in Math are the 'e' (base of the natural logarithm) and 'pi' (ratio of the circumference of a circle to its diameter). These two constants are often required in the above calculations/operations. Hence the Math class java provides these two constants as double fields. Math.E – having a value as 2.718281828459045 Math.PI – having a value as 3.141592653589793 A) Let us have a look at the table below that shows us the Basic methods and its description Method Description Arguments abs Returns the absolute value of the argument Double, float, int, long round Returns the closed int or long (as per the argument) double or float ceil Math ceil function in Java returns the smallest integer that is greater than or equal to the argument Double floor Java floor method returns the largest integer that is less than or equal to the argument Double min Returns the smallest of the two arguments Double, float, int, long max Returns the largest of the two arguments Double, float, int, long Below is the code implementation of the above methods: Note: There is no need to explicitly import java.lang.Math as its imported implicitly. All its methods are static. Integer Variable int i1 = 27; int i2 = -45; Double(decimal) variables double d1 = 84.6; double d2 = 0.45; Java Math abs() method with Example Java Math abs() method returns the absolute value of the argument. public class Guru99 { public static void main(String args[]) { int i1 = 27; int i2 = -45; double d1 = 84.6; double d2 = 0.45; System.out.println("Absolute value of i1: " + Math.abs(i1)); System.out.println("Absolute value of i2: " + Math.abs(i2)); System.out.println("Absolute value of d1: " + Math.abs(d1)); System.out.println("Absolute value of d2: " + Math.abs(d2)); } } Expected Output: Absolute value of i1: 27 Absolute value of i2: 45 Absolute value of d1: 84.6 Absolute value of d2: 0.45 Java Math.round() method with Example Math.round() method in Java returns the closed int or long as per the argument. Below is the example of math.round Java method. public class Guru99 { public static void main(String args[]) { double d1 = 84.6; double d2 = 0.45; System.out.println("Round off for d1: " + Math.round(d1)); System.out.println("Round off for d2: " + Math.round(d2)); } } Expected Output: Round off for d1: 85 Round off for d2: 0 Java Math.ceil and Math.floor method with Example The Math.ceil and Math.floor in Java methods are used to return the smallest and largest integer that are greater than or equal to the argument. Below is the Math floor and ceiling Java example. public class Guru99 { public static void main(String args[]) { double d1 = 84.6; double d2 = 0.45; System.out.println("Ceiling of '" + d1 + "' = " + Math.ceil(d1)); System.out.println("Floor of '" + d1 + "' = " + Math.floor(d1)); System.out.println("Ceiling of '" + d2 + "' = " + Math.ceil(d2)); System.out.println("Floor of '" + d2 + "' = " + Math.floor(d2)); } } We will get the below output of the math.ceil in Java example. Expected Output: Ceiling of '84.6' = 85.0 Floor of '84.6' = 84.0 Ceiling of '0.45' = 1.0 Floor of '0.45' = 0.0 Java Math.min() method with Example The Java Math.min() method returns the smallest of the two arguments. public class Guru99 { public static void main(String args[]) { int i1 = 27; int i2 = -45; double d1 = 84.6; double d2 = 0.45; System.out.println("Minimum out of '" + i1 + "' and '" + i2 + "' = " + Math.min(i1, i2)); System.out.println("Maximum out of '" + i1 + "' and '" + i2 + "' = " + Math.max(i1, i2)); System.out.println("Minimum out of '" + d1 + "' and '" + d2 + "' = " + Math.min(d1, d2)); System.out.println("Maximum out of '" + d1 + "' and '" + d2 + "' = " + Math.max(d1, d2)); } } Expected Output: Minimum out of '27' and '-45' = -45 Maximum out of '27' and '-45' = 27 Minimum out of '84.6' and '0.45' = 0.45 Maximum out of '84.6' and '0.45' = 84.6 B) Let us have a look at the table below that shows us the Exponential and Logarithmic methods and its description- Method Description Arguments exp Returns the base of natural log (e) to the power of argument Double Log Returns the natural log of the argument double Pow Takes 2 arguments as input and returns the value of the first argument raised to the power of the second argument Double floor Java math floor returns the largest integer that is less than or equal to the argument Double Sqrt Returns the square root of the argument Double Below is the code implementation of the above methods: (The same variables are used as above) public class Guru99 { public static void main(String args[]) { double d1 = 84.6; double d2 = 0.45; System.out.println("exp(" + d2 + ") = " + Math.exp(d2)); System.out.println("log(" + d2 + ") = " + Math.log(d2)); System.out.println("pow(5, 3) = " + Math.pow(5.0, 3.0)); System.out.println("sqrt(16) = " + Math.sqrt(16)); } } Expected Output: exp(0.45) = 1.568312185490169 log(0.45) = -0.7985076962177716 pow(5, 3) = 125.0 sqrt(16) = 4.0 C) Let us have a look at the table below that shows us the Trigonometric methods and its description- Method Description Arguments Sin Returns the Sine of the specified argument Double Cos Returns the Cosine of the specified argument double Tan Returns the Tangent of the specified argument Double Atan2 Converts rectangular coordinates (x, y) to polar(r, theta) and returns theta Double toDegrees Converts the arguments to degrees Double Sqrt Returns the square root of the argument Double toRadians Converts the arguments to radians Double Default Arguments are in Radians Below is the code implementation: public class Guru99 { public static void main(String args[]) { double angle_30 = 30.0; double radian_30 = Math.toRadians(angle_30); System.out.println("sin(30) = " + Math.sin(radian_30)); System.out.println("cos(30) = " + Math.cos(radian_30)); System.out.println("tan(30) = " + Math.tan(radian_30)); System.out.println("Theta = " + Math.atan2(4, 2)); } } Expected Output: sin(30) = 0.49999999999999994 cos(30) = 0.8660254037844387 tan(30) = 0.5773502691896257 Theta = 1.1071487177940904 Now, with the above, you can also design your own scientific calculator in java.