

More about the Math Class

The Java `Math` class is a collection of static methods for performing specific mathematical operations. This class is in the `java.lang` package, so there is no need for an `import` statement to use it.

In Chapter 2, you were introduced to the `Math.pow` method, which returns the value of a number raised to a power. Table F-1 describes several of the `Math` class's methods.

Table F-1 Several `Math` class methods

Method	Example Usage	Description
<code>abs</code>	<code>y = Math.abs(x);</code>	Returns the absolute value of the argument. This method can accept and return values of the <code>double</code> , <code>float</code> , <code>int</code> , and <code>long</code> data types.
<code>acos</code>	<code>y = Math.acos(x);</code>	Returns the arc-cosine of the argument. The argument should be the cosine of an angle. (The argument's value must be in the range from <code>-1.0</code> through <code>1.0</code> .) The return type and the argument are <code>doubles</code> .
<code>asin</code>	<code>y = Math.asin(x);</code>	Returns the arc-sine of the argument. The argument should be the sine of an angle. (The argument's value must be in the range from <code>-1.0</code> through <code>1.0</code> .) The return type and the argument are <code>doubles</code> .
<code>atan</code>	<code>y = Math.atan(x);</code>	Returns the arc-tangent of the argument. The argument should be the tangent of an angle. The return type and the argument are <code>doubles</code> .
<code>cbrt</code>	<code>y = Math.cbrt(x);</code>	Returns the cube root of the argument. The return type and argument are <code>doubles</code> . Note: This method was introduced in Java 5.

Method	Example Usage	Description
<code>ceil</code>	<code>y = Math.ceil(x);</code>	Returns the smallest number that is greater than or equal to the argument. The return type and the argument are <code>doubles</code> .
<code>cos</code>	<code>y = Math.cos(x);</code>	Returns the cosine of the argument. The argument should be an angle expressed in radians. The return type and the argument are <code>doubles</code> .
<code>exp</code>	<code>y = Math.exp(x);</code>	Computes the exponential function of the argument, which is e^x . The return type and the argument are <code>doubles</code> .
<code>floor</code>	<code>y = Math.floor(x);</code>	Returns the largest number that is less than or equal to the argument. The return type and the argument are <code>doubles</code> .
<code>hypot</code>	<code>c = Math.hypot(a, b);</code>	Returns the length of the hypotenuse of a right triangle. The arguments <code>a</code> and <code>b</code> are the lengths of the other two sides of the triangle. The return type and the argument are <code>doubles</code> . Note: This method was introduced in Java 5.
<code>log</code>	<code>y = Math.log(x);</code>	Returns the natural logarithm of the argument. The return type and the argument are <code>doubles</code> .
<code>pow</code>	<code>y = Math.pow(x, z);</code>	Returns the value of the first argument raised to the power of the second argument. The return type and the argument are <code>doubles</code> .
<code>round</code>	<code>y = Math.round(x);</code>	Returns the value of the argument, as an integer, rounded to the nearest whole number. The argument is expected to be a <code>double</code> or a <code>float</code> . If the argument is a <code>double</code> , the return type is <code>long</code> . If the argument is a <code>float</code> , the return type is <code>int</code> .
<code>sin</code>	<code>y = Math.sin(x);</code>	Returns the sine of the argument. The argument should be an angle expressed in radians. The return type and the argument are <code>doubles</code> .
<code>sqrt</code>	<code>y = Math.sqrt(x);</code>	Returns the square root of the argument. The return type and argument are <code>doubles</code> .
<code>tan</code>	<code>y = Math.tan(x);</code>	Returns the tangent of the argument. The argument should be an angle expressed in radians. The return type and the argument are <code>doubles</code> .
<code>toDegrees</code>	<code>y = Math.toDegrees(x);</code>	Accepts as an argument an angle in radians. The angle converted to degrees is returned. The argument and return values are both <code>doubles</code> .
<code>toRadians</code>	<code>y = Math.toRadians(x);</code>	Accepts as an argument an angle in degrees. The angle converted to radians is returned. The argument and return values are both <code>doubles</code> .

We will not cover all of these methods in depth, but let's take a closer look at some of them. The `Math.abs` method returns the absolute value of its argument. The following code segment shows an example of how it is used.

```
double x = -4.2, y = 4.2, a, b;
a = Math.abs(x);
b = Math.abs(y);
```

After this code executes, both the variables `a` and `b` will contain the value 4.2. The argument to the `Math.abs` method can be of the `double`, `float`, `int`, or `long` data types. This is because there are several overloaded versions of the method. The method's return value will be of the same data type as the argument.

Here is a program segment that demonstrates the `Math.sqrt` method, which returns the square root of a number:

```
num = 25.0;
s = Math.sqrt(num);
System.out.println("The square root of " + num + " is " + s);
```

The output of this program segment is

```
The square root of 25.0 is 5.0
```

The `Math.round` method returns the value of its argument rounded to the nearest whole number. The following code segment shows an example.

```
double x = 4.2, y = 4.8;
long a, b;
a = Math.round(x);
b = Math.round(y);
```

After this code executes, `a` will contain 4 and `b` will contain 5.

In addition to these and other methods, the `Math` class defines two static `final` variables, which are listed in Table F-2.

Table F-2 The **E** and **PI** constants

Method	Example Usage	Description
E	<code>y = x * Math.E;</code>	This is the mathematical constant known as <i>e</i> (for Euler's number). It is defined as 2.7182818284590452354.
PI	<code>area = Math.PI * radius * radius;</code>	This is the mathematical constant Pi, or π . It is defined as 3.14159265358979323849.

Both of these `final` variables are public, so you can access them directly from the class.