Documentation

The Java™ Tutorials

Trail: Learning the Java Language
Lesson: Numbers and Strings
Section: Numbers

The Java Tutorials have been written for JDK 8. Examples and practices described in this page don't take advantage of improvements introduced in later releases and might use technology no longer available.
See Java Language Changes for a summary of updated language features in Java SE 9 and subsequent releases.
See JDK Release Notes for information about new features, enhancements, and removed or deprecated options for all JDK releases.

Beyond Basic Arithmetic

The Java programming language supports basic arithmetic with its arithmetic operators: +, -, *, /, and %. The `Math` class in the `java.lang` package provides methods and constants for doing more advanced mathematical computation.

The methods in the `Math` class are all static, so you call them directly from the class, like this:

```
Math.cos(angle);
```

Note: Using the `static import` language feature, you don't have to write `Math` in front of every math function:

```
import static java.lang.Math.*;
```

This allows you to invoke the `Math` class methods by their simple names. For example:

```
cos(angle);
```

Constants and Basic Methods

The `Math` class includes two constants:

- `Math.E`, which is the base of natural logarithms, and
- `Math.PI`, which is the ratio of the circumference of a circle to its diameter.

The `Math` class also includes more than 40 static methods. The following table lists a number of the basic methods.

Basic Math Methods

Method	Description

`double abs(double d)`
`float abs(float f)`
`int abs(int i)`
`long abs(long lng)`
	Returns the absolute value of the argument.

`double ceil(double d)`	Returns the smallest integer that is greater than or equal to the argument. Returned as a double.
`double floor(double d)`	Returns the largest integer that is less than or equal to the argument. Returned as a double.
`double rint(double d)`	Returns the integer that is closest in value to the argument. Returned as a double.
`long round(double d)` `int round(float f)`	Returns the closest long or int, as indicated by the method's return type, to the argument.

`double min(double arg1, double arg2)`
`float min(float arg1, float arg2)`
`int min(int arg1, int arg2)`
`long min(long arg1, long arg2)`
	Returns the smaller of the two arguments.

Method	Description

`double max(double arg1, double arg2)`
`float max(float arg1, float arg2)`
`int max(int arg1, int arg2)`
`long max(long arg1, long arg2)`
	Returns the larger of the two arguments.

The following program, `BasicMathDemo` , illustrates how to use some of these methods:

```
public class BasicMathDemo {
    public static void main(String[] args) {
        double a = -191.635;
        double b = 43.74;
        int c = 16, d = 45;

        System.out.printf("The absolute value " + "of %.3f is %.3f%n",
                          a, Math.abs(a));

        System.out.printf("The ceiling of " + "%.2f is %.0f%n",
                          b, Math.ceil(b));

        System.out.printf("The floor of " + "%.2f is %.0f%n",
                          b, Math.floor(b));

        System.out.printf("The rint of %.2f " + "is %.0f%n",
                          b, Math.rint(b));

        System.out.printf("The max of %d and " + "%d is %d%n",
                          c, d, Math.max(c, d));

        System.out.printf("The min of of %d " + "and %d is %d%n",
                          c, d, Math.min(c, d));
    }
}
```

Here's the output from this program:

```
The absolute value of -191.635 is 191.635
The ceiling of 43.74 is 44
The floor of 43.74 is 43
The rint of 43.74 is 44
The max of 16 and 45 is 45
The min of 16 and 45 is 16
```

Exponential and Logarithmic Methods

The next table lists exponential and logarithmic methods of the `Math` class.

Exponential and Logarithmic Methods

Method	Description
`double exp(double d)`	Returns the base of the natural logarithms, e, to the power of the argument.
`double log(double d)`	Returns the natural logarithm of the argument.
`double pow(double base, double exponent)`	Returns the value of the first argument raised to the power of the second argument.
`double sqrt(double d)`	Returns the square root of the argument.

The following program, `ExponentialDemo`, displays the value of e, then calls each of the methods listed in the previous table on arbitrarily chosen numbers:

```
public class ExponentialDemo {
    public static void main(String[] args) {
        double x = 11.635;
        double y = 2.76;

        System.out.printf("The value of " + "e is %.4f%n",
                          Math.E);
```

```
            System.out.printf("exp(%.3f) " + "is %.3f%n",
                              x, Math.exp(x));

            System.out.printf("log(%.3f) is " + "%.3f%n",
                              x, Math.log(x));

            System.out.printf("pow(%.3f, %.3f) " + "is %.3f%n",
                              x, y, Math.pow(x, y));

            System.out.printf("sqrt(%.3f) is " + "%.3f%n",
                              x, Math.sqrt(x));
        }
    }
```

Here's the output you'll see when you run `ExponentialDemo`:

```
The value of e is 2.7183
exp(11.635) is 112983.831
log(11.635) is 2.454
pow(11.635, 2.760) is 874.008
sqrt(11.635) is 3.411
```

Trigonometric Methods

The `Math` class also provides a collection of trigonometric functions, which are summarized in the following table. The value passed into each of these methods is an angle expressed in radians. You can use the `toRadians` method to convert from degrees to radians.

Trigonometric Methods

Method	Description
`double sin(double d)`	Returns the sine of the specified double value.
`double cos(double d)`	Returns the cosine of the specified double value.
`double tan(double d)`	Returns the tangent of the specified double value.
`double asin(double d)`	Returns the arcsine of the specified double value.
`double acos(double d)`	Returns the arccosine of the specified double value.
`double atan(double d)`	Returns the arctangent of the specified double value.
`double atan2(double y, double x)`	Converts rectangular coordinates `(x, y)` to polar coordinate `(r, theta)` and returns `theta`.
`double toDegrees(double d)` `double toRadians(double d)`	Converts the argument to degrees or radians.

Here's a program, `TrigonometricDemo`, that uses each of these methods to compute various trigonometric values for a 45-degree angle:

```
public class TrigonometricDemo {
    public static void main(String[] args) {
        double degrees = 45.0;
        double radians = Math.toRadians(degrees);

        System.out.format("The value of pi " + "is %.4f%n",
                          Math.PI);

        System.out.format("The sine of %.1f " + "degrees is %.4f%n",
                          degrees, Math.sin(radians));

        System.out.format("The cosine of %.1f " + "degrees is %.4f%n",
                          degrees, Math.cos(radians));

        System.out.format("The tangent of %.1f " + "degrees is %.4f%n",
                          degrees, Math.tan(radians));

        System.out.format("The arcsine of %.4f " + "is %.4f degrees %n",
                          Math.sin(radians),
                          Math.toDegrees(Math.asin(Math.sin(radians))));

        System.out.format("The arccosine of %.4f " + "is %.4f degrees %n",
```

```
                                Math.toDegrees(Math.acos(Math.cos(radians))));

            System.out.format("The arctangent of %.4f " + "is %.4f degrees %n",
                                Math.tan(radians),
                                Math.toDegrees(Math.atan(Math.tan(radians))));
        }
    }
```

The output of this program is as follows:

```
The value of pi is 3.1416
The sine of 45.0 degrees is 0.7071
The cosine of 45.0 degrees is 0.7071
The tangent of 45.0 degrees is 1.0000
The arcsine of 0.7071 is 45.0000 degrees
The arccosine of 0.7071 is 45.0000 degrees
The arctangent of 1.0000 is 45.0000 degrees
```

Random Numbers

The `random()` method returns a pseudo-randomly selected number between 0.0 and 1.0. The range includes 0.0 but not 1.0. In other words: `0.0 <= Math.random() < 1.0`. To get a number in a different range, you can perform arithmetic on the value returned by the random method. For example, to generate an integer between 0 and 9, you would write:

```
int number = (int)(Math.random() * 10);
```

By multiplying the value by 10, the range of possible values becomes `0.0 <= number < 10.0`.

Using `Math.random` works well when you need to generate a single random number. If you need to generate a series of random numbers, you should create an instance of `java.util.Random` and invoke methods on that object to generate numbers.

Previous page: Formatting Numeric Print Output
Next page: Summary of Numbers