Function calls

You have seen *function calls* since high school, like

```
sqrt(25)      // square root of 25. Its value is 5.0.
abs(-7)       // absolute value of -7. Its value is 7.
cos(0)        // cosine of angle 0. It value is 1.0.
min(5, 7)     // minimum of 5 and 7. Its value is 5.
```

You will learn to write functions in Java later. Here we just look at how to write *calls* on functions.

A function call has the form

<*function-name*> (<*arguments*>)

where <*arguments*> is a list of expressions separated by commas. The first three function calls you see here have one argument. The last function call has two arguments —the expressions `5` and `7`— and they are separated by a comma. You have even seen a function with no arguments: function `length` (remember the lecture on type `String`).

Please remember the term *argument* of a function call. You may see it called the *actual parameter*. We do not use this terminology because it is confusing, as you will see when we discuss writing functions themselves.

A function call is an expression, and you can put one anywhere an expression could be placed, as you can see in this expression:

```
1 + min(5, 7) * 2
```

Secondly, since a function call is an expression, it can be *evaluated* to *yield a result*. The value the expression `min(5, 7)` is `5`, so the value of the expression given above is `11`.

Since Java is strongly typed, arguments of function calls have to have certain types. For example, you can't ask for the square root of a boolean value. And the value of a function call also has a specific, known type.

Java has thousands of built-in functions for you to use. Here, we tell you about a few of the mathematical ones. Their names begin with "`Math.`" —for reasons that will become clear later.

Function `Math.sqrt` yields the square root of its argument. Its argument is a **`double`** value —if it is a narrower type, like **`int`**, the argument will be promoted to a **`double`**. And it yields a **`double`** result.

```
Math.sqrt(double)          // Yields a double.
    Math.sqrt(25.0)        // The square root of 25.0. Its value is 5.0.
    Math.sqrt(5.0)         // The square root of 5.0. Its value is 2.23606797749979.
    Math.sqrt(25)          // The square root of 25.0. Its value is 5.0.
    Math.sqrt(5)           // The square root of 5.0. Its value is 2.23606797749979.
```

Function `Math.abs` yields the absolute value of its **`int`** argument —found simply changing the sign of the argument, if necessary, so that the answer is positive.

```
Math.abs(int)              // Yields an int.
    Math.abs(5)            // The absolute value of 5. Its value is 5.
    Math.abs(-5)           // The absolute value of -5. Its value is 5.
```

Actually, there are several absolute functions, depending on the argument type. For example,

```
Math.abs(double)           // Yields a double.
    Math.abs(5.0)          // The absolute value of 5.0. Its value is 5.0.
    Math.abs(-5.0)         // The absolute value of -5.0. Its value is 5.0.
```

Function `Math.min` yields the minimum of its two **`int`** arguments. And, you can see by testing in the interactions pane that there are two min methods, one for **`int`** arguments and another for **`double`** arguments. Note that if one argument is a **`double`** and the other is an **`int`**, the **`int`** value will be automatically promoted to type **`double.`**

```
Math.min(int, int)         // Yields an int.
    Math.min(5, 7)         // minimum of 5 and 7. Its value is 5.
```

```
        Math.min(5, -7)         // minimum of 5 and -7. Its value is -7.

    Math.min(double, double)// Yields a double.
        Math.min(5.0, 7.0)      // minimum of 5.0 and 7.0. Its value is 5.0.
        Math.min(5.0, -7)       // minimum of 5.0 and -7. Its value is -7.0.
```

Three useful functions are

`Math.round(`**`double`**`)`, `Math.floor(`**`double`**`)`, `Math.ceil(`**`double`**`)`.

Math.round rounds its argument to the nearest integer:

```
        Math.round(5.6)         // Its value is the int 6.
        Math.round(5.4)         // Its value is the int 5.
```

`Math.floor` and `Math.ceil` are the floor and ceiling function. Think of one integer, say 5, as being the floor, the next integer, say 6, being the ceiling, and all the real numbers between 5 and 6 being between the floor and ceiling.

A more mathematical definition is that

`floor(x)` is the largest integer that is at most `x`.
`ceil(x)` is the smallest integer that is at least `x`.

Note that the result, although an integer, is of type **`double`**.

```
    Math.floor(5.6)     // Its value is the double 5.0.
    Math.floor(5.4)     // Its value is the double 5.0.
    Math.ceil(5.6)      // Its value is the double 6.0.
    Math.ceil(5.4)      // Its value is the double 6.0.
```

There are many more functions like this. We just want to give you an idea of what is available. You can find more of them listed in a table on page 24 of Gries/Gries, and later, we will show you how to access the specifications of all the functions that begin with “`Math.`”. Also, we wanted to acquaint you with our terminology for dealing with functions.