Method Calls

The arithmetic operators we've used thus far are simple. We can combine them together to combine complex expressions, but that also makes our code more difficult to read. More complicated operations are often handled by *methods* which put a simple name to a potentially complex operation. Methods also allow Java to be *extensible*, in that you can create your own methods to perform custom tasks (we will see how to do this in future chapters).

We formed arithmetic expressions with arithmetic operators. Similarly, we can form *method call* expressions by providing operands to a method. When the method call expression is evaluated, the method will calculate a value based on the arguments it was given (arguments, in this case, are values derived from other expressions -- similar to the operands we supplied to operators in section 1.3). In this context, the term *function* is also applicable. Here is an example of a simple function call:

Example 1. Calling the Square Root Method

We will see many types of methods in future chapters, but the methods we're calling in this chapter are called *static* methods. In order to use some of the methods that come standard with Java, we should first import them. Typing in the following line will make the square root method available (note that this method is named `sqrt`).

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

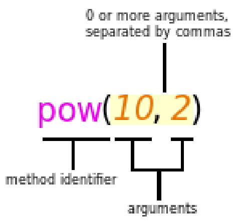
Note that `Math` must be capitalized and the line should end in a semicolon. Once you've typed this in, you're allowed to make calls to the square root function at will. For example:

```
sqrt(4)
```

This expression should yield the value `2.0`. Note that the method returns a double value, even though `4` is a perfect square. Try calling this method with some other, messier values.

To call a method, first give the name of the method you wish to call. Following the name should be a pair of parenthesis, within which is a comma-separated list of expressions for each argument. In the last example, there was only one argument, but a method could take more than one argument, or could even take zero arguments.

Figure 1. Anatomy of a Method Call



In the diagram above, we're calling a method named pow which takes two arguments: the base and the power. The method returns the result of raising the specified base to the specified power (that is to say, this expression evaluates to the value 100).