METHOD CALLS WITH ARGUMENTS

A method specification should include a list of its parameters, which are variables within the method that receive data from arguments in the method call. A well-written specification tells you:

1. The number of parameters.
2. The data type of each.
3. The purpose of each.

Example

```java
int totalMinutes( int hours, int minutes )
// Calculate and return the total minutes from the given
// number of hours and minutes.
```

The method specification above tells you:

1. The method has two parameters.
2. Both are int variables.
3. The first is the number of hours and the second the number of minutes.

Your method call must provide arguments that are consistent with the parameters to which they are being passed. An argument can be a constant, variable or expression.

Examples

Each of these is a correctly formed call to the totalMinutes method given above.

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>totalMinutes( 2, 30 )</td>
<td>This says to call the method passing 2 to hours and 30 to minutes.</td>
</tr>
<tr>
<td>totalMinutes( hr, min )</td>
<td>This says to call the method passing the value of hr to hours and min to minutes. An argument need not have the same identifier as a parameter.</td>
</tr>
<tr>
<td>totalMinutes( 0, hr*60 )</td>
<td>This says to call the method passing 0 to hours and hr*60 to minutes.</td>
</tr>
</tbody>
</table>
The number of arguments and the data type of each must match those of the parameters in the method specification.

**Examples**

This call to method `totalMinutes`:

```java
m = totalMinutes();
```

gives this compiler diagnostic:

```
method totalMinutes cannot be applied to given types;
   m = totalMinutes();
   ^
required: int, int
found: no arguments
reason: actual and formal argument lists differ in length
```

The large red arrow marks the key lines in the diagnostic, which tell you that the call to `totalMinutes` requires two arguments whereas this call has none.

This call to method `totalMinutes`:

```java
m = totalMinutes(2.5, 30.25);
```

gives this compiler diagnostic:

```
method totalMinutes cannot be applied to given types;
   m = totalMinutes(2.5, 30.25);
   ^
required: int, int
found: double, double
reason: actual argument double cannot be converted to int
```

Again the red arrow marks the key lines, which tell you that the call to `totalMinutes` requires two `int` arguments and this call has two `double` arguments.
The order of the arguments must match the order of the parameters in the specification.

**Examples**
If you want to calculate the total number of minutes in 3 hours and 45 minutes.

<table>
<thead>
<tr>
<th>Call</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>totalMinutes( 3, 45 )</code></td>
<td>This call is correct, passing 3 to <strong>hours</strong> and 45 to <strong>minutes</strong>.</td>
</tr>
<tr>
<td><code>totalMinutes( 45, 3 )</code></td>
<td><strong>Wrong.</strong> This passes 45 to <strong>hours</strong> and 3 to <strong>minutes</strong>.</td>
</tr>
</tbody>
</table>

Don’t make the novice mistake of mimicking the declaration syntax used in the method specification when writing a method call. A method specification tells you the characteristics of the method. Your method call is to abide by, but not mimic, this information.

**Examples**
Each of the following calls to method `totalMinutes` are syntactically incorrect. The programmer has taken the method specification too literally when writing his or her call.

<table>
<thead>
<tr>
<th>Call</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>m = int totalMinutes( 2, 30 );</code></td>
</tr>
<tr>
<td><code>m = totalMinutes( int 2, int 30 );</code></td>
</tr>
<tr>
<td><code>m = totalMinutes( int hours, int minutes );</code></td>
</tr>
</tbody>
</table>
**Exercises**

The Body Mass Index (BMI) is an internationally used measure of obesity. It was originally developed by the Belgium statistician Adolphe Quetelet (1796-1874). You can calculate it from your height (in inches) and your weight (in pounds) using the formula:

\[
BMI = \frac{weight \times 703}{height^2}
\]

Following is a Java method specification for a method that calculates and returns a person’s BMI. The person’s weight and height must be passed to it.

```java
public double bmi( int weight, double height )
// Calculate and return a patient's Body Mass Index given
// his or her weight and height.
```

1. List the following three facts revealed by this specification.

   - The number of arguments it requires.
   - The data type of each argument.
   - The purpose of each argument.

For each of the following calls to the `bmi` method, circle what's wrong and explain. Some of the calls are correct.

Assume that your application has declared and initialized these variables:

- `w` an `int` variable holding the patient’s weight
- `h` a `double` variable holding the height.

2. `bmi( 120, 66.0 )`
3. `bmi( )`
4. `BMI( 120, 66.0 )`
5. `bmi( 120, 66 )`
6. `bmi( 120.0, 66.0 )`
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>7.</td>
<td><code>double bmi(int 120, double 66.5)</code></td>
</tr>
<tr>
<td>8.</td>
<td><code>bmi(w, h)</code></td>
</tr>
<tr>
<td>9.</td>
<td><code>double bmi(int w, double h)</code></td>
</tr>
<tr>
<td>10.</td>
<td><code>bmi(h, w)</code></td>
</tr>
</tbody>
</table>