METHOD CALLS WITH RETURN VALUES

When calling a method that returns a value, you generally, but not necessarily, want to use the value that it returns. This is most easily accomplished by embedding the call within an assignment statement.

**Example**
This main method shows a call to `in.nextDouble()` in line 6 embedded within an assignment statement. The value returned by the method is saved into the variable `weight`.

```java
public static void main( String [] args )
{
    double weight;  // patient's weight
    Scanner in = new Scanner( System.in );
    System.out.print( "Enter patient's weight: " );
    weight = in.nextDouble();
    System.out.println( "weight = " + weight );
}
```

You can embed the method call within any expression that expects a value.

**Example**
This main method embeds the call to `in.nextDouble` directly within the `println` (line 5).

```java
public static void main( String [] args )
{
    Scanner in = new Scanner( System.in );
    System.out.print( "Enter patient's weight: " );
    System.out.println( "weight = " + in.nextDouble() );
}
```
If you’re not interested in keeping the return value, then make the method call a statement in and of itself.

**Example**

This main method wants the program to pause and wait for the user to respond. This is done by line 5 where the call to `in.nextLine` is written as a standalone statement. The programmer has not saved the return value.

```
1 public static void main( String [] args )
2 {
3     Scanner in = new Scanner( System.in );
4     System.out.print( "Press the ENTER key when ready." );
5     in.nextLine( );
6     . . .
```

When you do use the method’s return value, make sure that the context of the method call is consistent with the data type of the return value.

**Example**

Given this method specification:

```
double toYen( double dollars )
// Return the amount of Japanese yen equivalent to the
// given amount of U.S. dollars.
```

Each of the following calls to `toYen` are legal:

```
double jpy = toYen( 100.00 );
System.out.println( toYen( 100.00 ) );
double total = toYen( 100.00 ) + COMMISSION;
```

As is this call, even though it doesn’t make much sense because the return value is lost:

```
toYen( 100.00 );
```

But this call is not legal because the `double` return value cannot be stored in an `int`:

```
int jpy = toYen( 100.00 );
```
Exercises

Following is a Java method specification for a method that calculates and returns a worker’s wages. The worker’s hourly wage and the number of hours worked must be passed to it.

```
double wages( double hourlyWage, int hoursWorked )
// Calculate and return a worker's wages.
```

1. List the following facts revealed by this specification.

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>The number of arguments it requires.</td>
</tr>
<tr>
<td></td>
<td>The data type of each argument.</td>
</tr>
<tr>
<td></td>
<td>The purpose of each argument.</td>
</tr>
<tr>
<td></td>
<td>A description of its return value.</td>
</tr>
<tr>
<td></td>
<td>The data type of its return value.</td>
</tr>
</tbody>
</table>

For each of the following calls to the `wages` method, circle what's wrong and explain. Some of the calls are correct. Assume that your application has declared and initialized these variables:

- `rate` a double variable holding the worker’s hourly wage
- `hours` an int variable holding the number of hours worked

2. `double w = wages( rate, hours );`
3. `double w = wages( hours, rate );`
4. `wages( rate, hours );`
5. `System.out.println( wages( rate, hours ) );`
6. `int w = wages( rate, hours );`
Following is a Java method specification for a method that calculates and returns the amount withheld from a worker’s wages for income and Social Security taxes. The worker’s wages and number of allowances (declared on his or her federal W-4 form) must be passed to it.

double withheld( double wages, int w4Allowances )
// Calculate and return the amount withheld from a worker's wages given the number of allowances he or she declares on the W-4 form.

7. List the following facts revealed by this specification.

- The number of arguments it requires.
- The data type of each argument.
- The purpose of each argument.
- A description of its return value.
- The data type of its return value.

For each of the following code fragments, circle what's wrong with any method call and explain. Some of the code fragments are completely correct.

Assume that your application has declared and initialized these variables:

- rate a double variable holding the worker’s hourly wage
- hours an int variable holding the number of hours worked
- allowances an int variable holding the number of allowances

8. double wg = wages( rate, hours );
   double held = withheld( wg, allowances );

9. wages( rate, hours );
   withheld( wages, allowances );

10. double held = withheld( wages( rate, hours ), allowances );

11. int held = withheld( wages( rate, hours ), allowances );
12. `double held = withheld( allowances, wages( rate, hours ) );`

13. `System.out.print( withheld( wages( rate, hours ), allowances ) );`