```
    function display_properties()
    {
        print "Dog weight is $this->dog_weight. Dog breed is $this->dog_breed. Dog color is
        $this->dog_color.";
    }
}
?>
```

> *Program design recommendation—As seen in Example 3-4, it becomes much more important to be aware of the proper use of the opening and closing brackets ({ and }). For every open bracket, there must be a closing bracket. It can become difficult to determine if you are missing something. Editors (discussed in Chapter 1) can help color code everything to make it easier to see. Also, indenting (as seen in Example 3-4) can help visually line up the brackets. The PHP engine ignores extra spacing (called whitespace). So the programmer can make the code more visually pleasing and easier to debug.*

This class now has the ability to perform an action (via the `display_properties` method). So you can finally test its functionality. In order to do so, the code from Example 3-4 must be placed in the `dog.php` file (same as the class name) in the same location as the program that will use it.

We now need to create a program that will pull in this library (via the `require_once` statement). The program will then need to make an instance of the class (`Dog`). Finally, the program will need to call the method (`display_properties`) to display the contents of the properties.

> *Program design recommendation—PHP will allow the `include`, `include_once`, `require`, and `require_once` statements to be used anywhere in your program code. This could cause potential issues if it's used in the incorrect location or used more than once. It is strongly recommended that these statements be include together as close to the top of your code as possible for easy review to determine if libraries have already been installed.*
>
> *Security and reliability—PHP has several methods available to pull libraries into PHP programs. The `include` method will attempt to pull in a library. However, if the library does not exist, the program will continue to run (or crash). The `include` method also does not concern itself with possibility that the library might already have been attached to the code. It is possible that a large program might accidentally try to pull in the same library more than once (which would crash the program due to duplicate method and/or class names). The `include_once` method eliminates the possibility of attempting to pull in a library more than once. If the library has already been included, the statement will not execute. The `require` method does not allow the program to continue running if the library cannot be found. However, like the `include` method, it could attempt to pull in the same library more than once. The `require_once` method solves these potential problems by shutting the program down if the library cannot be found and by only installing the library if it has not already been installed.*

The format of the `require_once` statement is simple. The keyword `require_once` is followed with the library name (`dog.php`). The statement should be included near the top of your code and before you make an actual instance of the class (`Dog`).

```
require_once("dog.php");
```