As mentioned previously, a class contains properties. *Properties* are also called *variables*. Properties include characteristics of the class. When an instance of a class is created, the properties are unique for that object. The operating system reserves a space in memory to hold the properties. The operating system handles memory management for us, including cleaning up properties that are no longer needed. In PHP, anytime a closing bracket (}) is reached, properties that have been created are scheduled for removal by the garbage collector of the operating system. The program can no longer access the property at that point.

Properties can hold many different types of data. In most languages, when a property is created you must also include a data type to describe the kind of data being stored (such as string). However, PHP does not require the defining of a data type. PHP determines the type of data to be stored in a property the first time data is placed in the property. Properties are created with an initial $ and the name of the property. Property names can include alphabetic characters, numbers, and the _. The _ can be used at the beginning of the property (after the $) or between words. No spaces are allowed. Properties are commonly created with lowercase letters. However, PHP does allow uppercase letters. PHP is case-sensitive and will consider a lowercase property (`speak`) and an uppercase property (`Speak`) two different properties.

Example 3-3. Basic class structure with properties in dog.php file

```
<?php
class Dog
{
private $dog_size = 0;
private $dog_breed = "no breed";
private $dog_color = "no color";
private $dog_name = "no name";
}
?>
```

> *Program design recommendation—Properties are created on the fly in PHP. Properties are created the first time you use them. This can both be a help and a pain. If you misspell a property name, PHP will not produce an error. Instead it will create a new property with the misspelled name. It is recommended (when possible) that properties be created with initial values at the top of your program (or method, or class) to more easily determine what your property name is and whether it has been created.*

As seen in Example 3-3, each of the properties (except `$dog_size`) for the `Dog` class has been declared `private` and initially set with a string (text). The `$dog_size` property has been set to the number zero (we know it is a number and not a string because there are no `""` around the zero). The operating system will store the strings values in the properties (`$dog_breed`, `$dog_color`, and `$dog_name`) in ASCII format (combinations of zeros and ones to represent each character) and will store the value in the `$dog_size` property in a numerical format in memory. The operating system creates memory tables to look up the actual memory address of the value in a variable when it is used in a program.

■ **Note** The format of the code must also include a semicolon at the end of the code statements (all code lines that are executed must include semicolons).

As you may note, at this point the example is not very useful. Even if you were to create an instance of the class, you could not access anything in the class or display the values in the properties. Let's add a method to the class to allow you to display what is contained in the class. To do so, you need to create a method to display the values using the `print` statement. You can also take this opportunity to build one single string of output using *string concatenation*.