```
function set_dog_breed($value)
{
$error_message = TRUE;
(ctype_alpha($value) && strlen($value) <= 35) ? $this->dog_breed = $value : $error_message = FALSE;
return $error_message;
}
function set_dog_color($value)
{
$error_message = TRUE;
(ctype_alpha($value) && strlen($value) <= 15) ? $this->dog_color = $value : $error_message = FALSE;
return $error_message;
}
}
function get_properties()
{
return "$this->dog_weight,$this->dog_breed,$this->dog_color.";
}
}
?>
```

> *Program design recommendation—When coding and testing your programs, code just one* `set` *method. Then test the method to correct errors. After you have one successful* `set` *method, copy and paste it in your code and make the necessary changes. Do not attempt to completely code a program before testing it. Program piece by piece, then test. Although you may think that this slows down your coding, actually this is not true. By catching errors with each small addition to your program it will be easier to find them. If you attempt to code a complete program you may have lots of errors and could spend a lot of time trying to hunt down each error. If you are having difficulty finding an error, comment out (using* `//`*) the new lines of code in your program and retest. If all is okay, then gradually (just a few lines at a time) remove the comment lines (*`//`*) from your code lines and retest. This process should help you to find the lines of code that might be causing problems.*
>
> *Security and performance—In a live environment the programmer should not display details to the users as to what caused an update to be unsuccessful. Providing too much information can inform hackers on what can be changed to successfully update properties with invalid information. Pass the details of what caused the unsuccessful update to a secure log file on a server.*

The code is starting to get lengthy. However, each of the `set` functions is very similar. As you code the `set` functions, you will find that this is a common occurrence. It also allows you to quickly create `set` methods once you have a working error free example by copying and pasting working methods and making simple changes. In Example 3-9 different string lengths are determined depending on the type of information being updated. Also, the `$set_dog_weight` method checks for numeric values in the string passed, instead of alphabetic characters. Otherwise, the methods are almost identical.

Figure 3-2 demonstrates the output when valid information is passed into each property. The 'successful' messages display. Also note that the `get_properties` method displays the new updated values for each property. In a live environment you might consider not displaying the successful messages and only displaying the not successful messages.