Once you have completed all changes to the array (as requested by the business rules tier), you will return the information to the storage location in the destructor.

```
private $dogs_array = array(); // defined as an empty array initially
function __construct()
        {
               $xmlfile = file_get_contents(get_dog_application("datastorage"));
               $xmlstring = simplexml:load_string($xmlfile);
               $json = json_encode($xmlstring);
               $this->dogs_array = json_decode($json,TRUE);
        }
function __destruct()
{
            $xmlstring = '<?xml version="1.0" encoding="UTF-8"?>';
            $xmlstring .= "\n<dogs>\n";
            foreach ($this->dogs_array as $dogs=>$dogs_value) {
                foreach ($dogs_value as $dog => $dog_value)
                {
                        $xmlstring .="<$dogs>\n";
                        foreach ($dog_value as $column => $column_value)
                        {
                        $xmlstring .= "<$column>" . $dog_value[$column] . "</$column>\n";
                        }
                        $xmlstring .= "</$dogs>\n";
                }
    }
$xmlstring .= "</dogs>\n";  file_put_contents(get_dog_application("datastorage"),$xmlstring);
}
```

There are many ways that you can create XML data in PHP. The previous example takes a simplistic approach by supplying the XML tags from the array. As seen in the structure, there are three sets of arrays in this multidimensional array. The first `foreach` loop is used to flow through the first array (dogs). The second `foreach` loop handles the dog arrays (rows). Once inside this loop, the third `foreach` loop controls the columns in each dog array (each row).

The third loop retrieves the column names (from `$column`) and places them in XML tags. `$column` is also used to pull the value in the column (`$dog_value[$column]`). The `$xmlstring` supplies the same tags and structure as in the original XML file. Note that each line includes a newline character (`\n`) to display different lines in the file. The structure would work without this addition. However, if makes the file more readable in a text editor.

Once the `$xmlstring` has been created, the code uses a combination of the PHP `file_put_contents` method and the `get_dog_application` method (from Chapter 4) to open the XML file, replace the contents with the string contained in `$xmlstring`, and close the file.

You need to make one more final adjustment to the constructor to allow it to handle XML parsing errors. A parsing error occurs when something is wrong with the XML structure. The previous `dog_breed` and `dog_application` XML files are not updated by the application and are fairly stable. However, the XML file for the dog's information will be updated frequently. You need to handle any problems that may occur. You will raise a general error, which will be treated by `dog_interface` as an important error that is logged and e-mailed to support personnel. It will also display a `"System currently not available please try again later"` message to the users.