block. This process will continue until either a `catch` block has been discovered or it has been determined that the environment itself must handle the exception.

Using this process, you can throw exceptions in the `dog_container`, dog, and `get_breeds` modules without using catches. In the `dog_interface`, you can create a `try` block around calls to these files. Multiple `catch` blocks (or one with a `switch` statement) could be created in the interface to handle the exceptions from both the interface and all the other modules. This satisfies one of the requirements of three-tier programming. The business rules tier (and data tier) pass messages to the interface tier. The interface tier then determines how to handle those messages. It could display them to the users, place them in a log file (which you will look at later in this chapter), or ignore them (if it does not adversely affect the operation of the application).

Before you change the `Dog` application code, let's look at an example of exceptions being handled by the hierarchy.

Example 5-1. testerror.php with error and exception-producing methods

```
<?php
class testerror {
   function produceerror() {
       trigger_error( "User Error", E_USER_ERROR);
       echo "This line will not display";  }
   function throwexception() {
       throw new userException("User Exception");
       echo "This line will not display";  }  }
?>
```

Example 5-2. The handleerror.php file captures error or exception

```
<?php
function errorHandler($severity, $message, $file, $line) {
            throw new errorException($message, 0, $severity, $file, $line);
   }
class userException extends Exception { }
Set_error_handler('errorHandler');
try {
require_once("testerror.php");
$tester = new testerror();
$tester->produceerror();
echo "This line does not display";
$tester->throwexception(); // will not execute if produceerror() is executed
echo "This line does not display"; }
catch (errorException $e ){
           echo $e->getMessage(); }
catch (userException $e) {
  echo $e->getMessage(); }
catch (Exception $e) {
           echo $e->getMessage(); }
echo "This line will display";
?>
```