```
        $class_name = $class_array[$last_position];
        $dog_object = new $class_name($properties_array);
        return $dog_object;
        } } }
?>
```

If the path and file name are valid, the else portion of the code will execute. The value in $dog_loc is used in the require_once statement (which pulls the contents of the file into this method).

You must now determine the name of the class that exists in the file (dependence injection requires you to not only discover the file names and file path, but also the class names).

The PHP method get_declared_classes returns an array of all the classes that currently exist in the program. The classes are in order from first to last. Thus, since you just included the file with a class (either the dog or get_breeds class), you can hunt for the last entry in the array created. You have placed the array created by get_declared_classes in $class_array. You can now determine the size of the array. The PHP method count will return the size of an array. Remember the size of an array is the number of items in the array not the last position. If an array has a size of ten, the actual subscripts are 0 through 9, not 1 through 10.

```
$last_position = count($class_array) - 1;
```

For more information on get_declared_classes, visit
http://php.net/manual/en/function.get-declared-classes.php

With this in mind, this statement determines the size of $class_array and then subtracts 1 from the size and places that value in $last_position. Thus, if the array had a size of 10, the number 9 would be stored in $last_position (since the array subscripts would be 0 through 9, not 10).

```
$class_name = $class_array[$last_position];
```

You can then use the value in $last_position to pull the last class created from the array and place it into $class_name. As you can see, you can actually pass the property $last_position between the [] subscript brackets to indicate to the program you want the value of whatever is in the position located in $last_position. This allows you to be able to pull the information from the last position in any size array (since you have no idea of the size of the $class_array).

You can now create an instance of the class, because you have the class name in $class_name.

```
$dog_object = new $class_name($properties_array);
```

This line of code will now create an instance of any class (that accepts an array into the constructor) that has just been included (require_once) in the program. The method will create an instance of either the dog class or getBreeds class.

```
return $dog_object;
```

Finally, the object (which is either a dog object or a getBreeds object) is returned to the program that called the method. By returning the object, the calling program has complete access to the object and its properties and methods, even though it did not actually create the object (dog_interface can use the object even though dog_container created it).