This requires changing these lines in the dog.php file to accept an array and then use the values in the array to set each property.

Notice that $properties_array in the first line does not have anything that declares it as an array (except the name). Remember that PHP properties determine what data type they are by what is passed in the property. The same is true with arrays. If an array is passed into $properties_array, it becomes an array.

The address of the array is passed into the method. The property that accepts the address ($properties_array) actually points to the location of the array in memory. This is very efficient. If you needed to pass 50 items into a method, you could pass them all one at a time with different properties declared in the methods signature for each of these 50 items. However, instead, you can create an array that holds the 50 items and pass the array. You are only passing one value using this approach, which is the address of the array in memory. Also, as noted with this example, using an array allows you to be flexible in the amount of items you want to pass into a method signature.

```
$name_error = $this->set_dog_name($properties_array[0]) == TRUE ? 'TRUE,' : 'FALSE,';
```

To copy an item from an array, you refer to the item using the array's name ($properties_array) and the position that the item exists in the array ([0]). Notice that you use [] brackets for declaring the position. The position is commonly called the *subscript*. Arrays subscripts begin with position 0, not position 1.

```
class GetBreeds {
function __construct($properties_array)
{ //get_breeds constructor
if (!(method_exists('dog_container', 'create_object')))
{
exit;
}
}
```

You must adjust the get_breeds program. GetBreeds is now a class. You also create the signature of the GetBreeds constructor to accept an array. However, as you can see from this code, you actually will ignore anything that is passed into the array. This allows you to use the same method in dog_container to create an object of either class (or actually of any class that accepts an array). All you need to find is the name of the class (dog or GetBreeds) so you can make an instance of it.

```
function create_object($properties_array)
{
  $dog_loc = $this->get_dog_application();
  if(($dog_loc == FALSE) || (!file_exists($dog_loc)))
  {
    return FALSE;
  }
```

The create_object method then calls the get_dog_application method and places the value returned (the location of the file and file name) into $dog_loc. If the value in $dog_loc is FALSE or is not an existing file, the method returns FALSE to the program that called it.

```
else
  {
    require_once($dog_loc);
    $class_array = get_declared_classes();
        $last_position = count($class_array) - 1;
```