COMP2203 Lab 4: Objects and PHP

Objectives
By the end of this lab, you will have:

- Forked (copied) an existing Git repository.
- Used a specification to write a PHP class.
- Read data from a source file to create PHP objects.
- Implemented a filter to find specific PHP objects based on criteria.
- Committed your changes into your Git repository, and documented your code in a reasonable way.

Grades
- You must record the preparation points in your logbook for a D.
- To get a C, additionally complete part 1.
- To get a B, additionally complete part 2.
- To get an A, additionally complete part 3.

When recording answers in your log book, write the section and question number. **Write clearly and legibly** – scruffy work will lose you marks.

Your mark will improve if you use the correct tools, and write readable, commented code. We are happy to help you find the right tools and coding style.

Preparation
Before beginning the lab, we recommend you take some time to review PHP arrays and the use of classes/objects in PHP. You should be familiar with PHP data structures, and have an understanding of the object-oriented approach to programming. You should read and understand the note below.

An Important Note about the ECS PHP Development Server
In previous labs, we've been using your personal ECS webspace to build and test websites in this lab. This webspace does not show any PHP errors that occur, which makes development much harder than it should be.

From this week onward, we will be using the ECS Development Webspace, which will display PHP errors, and make development easier. Here's what you need to do:

- Save files to **linuxproj_html** instead of **public_html.**

- Access your website from http://linuxproj.ecs.soton.ac.uk/~<your-username> instead of http://users.ecs.soton.ac.uk/<your-username>

Part 1: Writing a Class in PHP

In this part of the lab, you are given some array-based data representing the cars Boyd has available for sale.

By analysing the structure of the data, you will write a PHP class that may be used to create an object for each car entry within the data.

Let's start by checking out the lab repo.

a. Fork the lab repo (www.github.com/COMP2203/lab4) into your Github account. Give the new repo a sensible name.

b. Use the Github Client application on the Lab PCs to clone your newly-forked repo into an empty directory in your **ECS Devevelopment Webspace** called comp2203-lab4.
The Clone is your working copy, and will be linked to the original repo at Github.

c. Check that the site is running properly in your webspace, by visiting http://linuxproj.ecs.soton.ac.uk/~<your-username>/comp2203-lab4.

Now you're ready to start editing webpages.

d. Look at carData.php. This file contains a large PHP array, representing all the cars Boyd has for sale.

e. Use `include()` to load the array from carData.php into the cars.php page. Use `print_r()` to verify that the array has loaded correctly.

f. Analyse the carData.php structure and write a PHP class in classes/car.class.php, containing the attributes of cars found in carData.php.

g. Create a new variable called `$cars`, and instantiate it as an array.
E.g. `$cars = array();`

h. Write a `foreach` loop that iterates over the array-based data.
For each item in the array, you should make a new car object, based on the car class you made earlier. Add each new car object into the `$cars` array.
Remember you can use `$cars[] = $newCar;` to quickly add `$newCar` to the `$cars` array.

i. Write a `foreach` loop that iterates over the new `$cars` array.
For each car object, use `echo` to create an advert for the car on the page.

j. Write a method in car.class.php to calculate and return the age of the car in years. Write this data into the advert for each car, by calling the function. Remember to document the function to the PHPDoc standard.

Get your work marked by a demonstrator before proceeding to part 2.

Part 2: Filtering Object–Based PHP Data

In this section, you will write a filter for Boyd's website that allows visitors to show only cars of a certain colour. The filter will run on the array of car objects you created in part 1, and hide/show cars as appropriate.

We have created the filtering form for you. It can be found as commented HTML in cars.php. To enable it, just remove the HTML comments.

Whenever a colour filter is selected on the webpage, cars.php is reloaded with a GET parameter. You can see this at the end of the URL, in the form:
`http://example.com/cars.php?colour=red`

The colour written into the URL depends upon the colour selected in the form.

To read in the selected colour in PHP, use the `$_GET` variable. This is an array of GET parameters available to the page.

For instance, if you loaded the page `http://example.com/test.php?name=Boyd`, you would access the name "Boyd" through the PHP variable `$_GET["name"];`

a. Un-comment the colour filtering form in cars.php and check that it submits. This form submits GET parameters to the server.

b. Select a colour from the form, and use `$_GET` to read the chosen colour into PHP. Echo the selected colour back out to ensure it is loaded correctly.

c. Adapt the `foreach` loop that writes out each car object so that the car object is only written when the colour of the car matches the colour chosen in the filter. Check that this works.

d. Now click the button to clear the colour filter, which will remove the $_GET parameters from the URL.
You will find an error occurs because your `foreach` loop is trying to access the `$_GET` variable, which doesn't exist in this scenario.
Adapt the `foreach` loop further so that it can handle this scenario (i.e. when the filter is disabled).
Some tips: Remember that you can specify multiple conditions in an `if()`, and perhaps look at PHP's `isset()` function. See www.php.net/isset.

Part 3: PHP Exercises

This part of the lab features a research task for a technique you may find useful in the future and ties together our OO-PHP work with some of the UML you may have done before.

Lastly, we have a general code-debugging exercise for you, based upon the current vogue in PHP bugs.

a. Research and document the difference between GET and POST parameters:
   i. What are they?
   ii. When should you use each one?
   iii. How would you would access them in PHP?

b. Draw a UML class diagram for the car class you wrote in part 1.

c. Find three bugs that would produce fatal syntax errors in this code:

```
// Array of our valued employees
$employees = array(
      "U3266543" => "McDonald, R.",
      "U2236642" => "Leghorn, F."
      "U3106621" => “Basset, B.”
);

// Print out each employee
foreach($employees as $id => $name) {
      echo "<p>$name (ID: $id)</p>";
};
```

Now take your log book to your demonstrator for marking.

Your Github settings should remain on the workstation you used this week. So, if you use the same workstation next week, you may save some time.