OOP PHP Lesson 1

For this lesson, we will be working in the folder called *01-vehicles.* Please save your files in this folder, then upload to your web server account (or local server will suffice). There is no database required for OOP lessons.

Create a file called *cars.php* in the root of this folder. Create a PHP block at the top and a basic HTML template below.

Note: If your copy/paste from this, you may have to change certain characters, especially double quotes.

```
<?php


 ?>

 <!DOCTYPE html>
 <html>
 <head>
 	<title></title>
 </head>
 <body>

 </body>
 </html>
```

In the PHP block, we will create a new Class. Remember, this Class will be a template for any new Object that we create from it.

Object Creation (Instantiation) and Properties

We can then declare some Properties in the class and define those properties later on as we Instantiate a new instance of this class to create our Object.

```
<?php
	class vehicle{
		public $color;
		public $num_doors;
		public $price;
		public $shape;
		public $brand;
		public $power;
		public $weight;
	}
 ?>
```

```
<!DOCTYPE html>
<html>
<head>
        <title></title>
</head>
<body>
<?php

        $myCar = new vehicle; // here we create a new object
        $myCar ->brand = "Porsche";
        $myCar ->price = 65000;
        $myCar ->shape = "SUV";
        $myCar ->color = "red";
        $myCar ->num_doors = "2";
        $myCar ->power = 450;
        $myCar ->weight = 3600;

?>
</body>
</html>
```

Methods

We don’t expect to see anything yet. Just setting things up. Let’s go ahead and create some Methods that can actually do something,

First, we will create the Constructor. This is something that will happen anytime a new object is created (instantiated) from this class.

Add this to your class (remember that there are 2 underscores to start the name of the function):

```
public function __construct(){
                        echo "<p> A vehicle is a mobile machine that transports passengers or
cargo.</p>";
                        echo "<p> About this vehicle</p>";
        }
```

Run that and you should see something written to the browser. The Constructor always happens anytime we create a new object.

Let’s try to create some Methods that will return the values of some of our Properties.

```
// add this to your class
public function showPrice(){
        echo "This vehicle costs ". $this->price . "<br>";
}
```

```
// add this to your HTML PHP block
$myCar->showPrice();
```

Note how we retrieve something using the *Object Access Operator* -> (aka, the “arrow thingy”) ☺
This can be both a getter and a setter as we also used it to set property values before. If you think you might have seen this in Wordpress code, then yes you have. All OOP PHP apps will use it.

Let’s do something a bit more robust shall we.

```
// add this to your class
public function showSpecs(){
                if((!$this->brand) || (!$this->num_doors)|| (!$this->shape)){
                        return "Sorry, but we do not have enough data to show the specs.";
                }
                else{
                        if($this->price < 25000){
                                $expense = "a cheap";
                        }elseif($this->price > 25000 && $this->price < 50000){
                                $expense = "a mid priced";
                        }elseif($this->price > 50000){
                                $expense = "an expensive";
                        }

                        if($this->num_doors == "2"){
                                $sport = "sporty";
                        }elseif($this->num_doors == "4"){
                                $sport = "family";
                        }else{
                                $sport = "";
                        }
                        return "This car is ". $expense . " ". $sport . " ". $this->color. " " . $this->shape . 
".<br>";
                }
        }
```

```
// add this to your HTML PHP block
echo "Vehicle description: " . $myCar->showSpecs();
```

Lets do one more. This Method will estimate the acceleration of your vehicle.

```
// add this to your class
public function showPerformance(){
			if((!$this->weight) || (!$this->power)){
				return "Sorry, but we do not have enough data to estimate performance.";
			}
			else{
				$ratio = round($this->power/$this->weight,2);
				$acceleration = round(($this->weight/$this->power)/2, 2);
				return "This vehicle has a Power-to-Weight Ratio of ". $ratio ." hp/lb, and should have a 0 - 100 km/hr time of ". $acceleration . " seconds<br>";
			}
	}
```

```
// add this to your HTML PHP block
echo "Expected performance: " . $myCar->showPerformance();
```

Re-using our Class (that's the point)

Ok, now let's create a separate file for our class, and use that for a whole bunch of vehicles.

Create a file called ***vehicle.class.php*** in the classes subfolder. Yeah, there's 2 dots in the filename. Hey, it's the New Normal.

```
<?php
// vehicle.class.php

class vehicle{
	public $color;
	public $num_doors;
	public $price;// integer please
	public $shape; // coupe, sedan, van, etc.
	public $brand;
	public $power;// hp integer
	public $weight;// lbs integer
	public function __construct(){
		echo "<h1> A vehicle  is a mobile machine that transports passengers or cargo.</h1>";
		echo "<h3> About this vehicle</h3>";
	}
	public function basicInfo(){
		return  $this->brand . " - " . $this->shape . " - $" . $this->price ;
	}
	public function showSpecs(){
		if((!$this->brand) || (!$this->num_doors)|| (!$this->shape)){
			return "Sorry, but we do not have enough data to show the specs.";
		}
		else{
			if($this->price < 25000){
				$expense = "a cheap";
			}elseif($this->price > 25000 && $this->price < 50000){
				$expense = "a mid priced";
			}elseif($this->price > 50000){
				$expense = "an expensive";
			}

			if($this->num_doors == "2"){
				$type = "sporty";
			}elseif($this->num_doors == "4"){
				$type = "family";
			}elseif($this->num_doors == "5"){
				$type = "utilitarian";
			}else{
				$type = "";
			}
			return "This car is ". $expense . " ". $type . " ". $this->color. " " . $this->shape .
".<br>" ;
		}
	}
	public function showPerformance(){
		if((!$this->weight) || (!$this->power)){
			return "Sorry, but we do not have enough data to estimate performance.";
		}
		else{
			$ratio = round($this->power/$this->weight,2);
			$acceleration = round(($this->weight/$this->power)/2, 2);
			return "This vehicle has a Power-to-Weight Ratio of ". $ratio ." hp/lb, and should
have a 0 - 100 km/hr time of ". $acceleration . " seconds<br>";
		}
	}
}//end class vehicle
?>
```

Now create a new file called ***subaru.php*** in the root.

```
<?php
// subaru.php

include("classes/vehicle.class.php"); // first, we include the class file.

$myCar = new vehicle;
 $myCar ->brand = "Subaru";
 $myCar ->price = 35000;
 $myCar ->shape = "SUV";
 $myCar ->color = "blue";
  $myCar ->num_doors = "5";

 $myCar->power = 250;
 $myCar->weight = 3600;

echo “<h1>Subaru</h1>”;
echo "Overview: " . $myCar->showSpecs();
echo "Expected performance: " . $myCar->showPerformance();
?>
```

On your own..

Create 2 more cars (like Subaru.php) . Do a bit of research to find out horsepower and curb weight (in lbs).

Extending our Class

Create a new file called ***motorcycle.class.php*** in the classes subfolder.

```
<?php
// motorcycle.class.php

class motorcycle extends vehicle{
            /*** the number of side cars ***/
            public $category;// roadster, racing, dirtbike

            public $handlebars; // clubman, cruiser, ape hangers

            public function motorCycleInfo(){
              return $this->category . " with ". $this->handlebars . " type handlebars.";
            }

} /*** end of class ***/

?>
```

Now, we create a new file called ***harley.php*** in the root.

```
<?php
// harley.php

include("classes/vehicle.class.php");
include("classes/motorcycle.class.php");

$myCar = new motorcycle;

 $myCar->brand = "Harley Davidson";
 $myCar->price = 45000;
 $myCar->shape = "roadster";
 $myCar->color = "black";
 $myCar->num_doors = "none";
 $myCar->power = 180;
 $myCar->weight = 1500;
 $myCar->category = "cruiser";
 $myCar->handlebars = "ape hangers";


echo "<h1>Harley< h1>";
echo "Overview: " . $myCar->showSpecs();
echo "Expected performance: " . $myCar->showPerformance();
echo "Motorcycle info: " . $myCar->motorCycleInfo();

?>
```

Here we *extend* the base class. The motorcycle class *inherits* all the properties and methods of the vehicle class, but can add some more. That's the point.

Ok. That is our first little exploration into how we can use OOP in our PHP. Hopefully, some of the terminology and the concept is starting to make sense.