Object Oriented Php

In this exercise you will play around with the Object Oriented Principals in Php. You will look at code examples and modify them to include new classes

You will work with:
Php
OOP
Classes
Inheritance
Interfaces
Encapsulation

Level:
Medium

Recommended Knowledge:
Php Basics
Variables
HTML
var_dump()

Step 1

Looking at a class

1) Look at a class

Open ***"index1.php"***.

In this file there is a class defined. This class is called "Person".

What properties does the class contain?

2) Look at the class instance

Just before the <body> starts, a new ***instance*** of the class is created and stored in the variable $john. Later in the HTML the properties of $john are access and echo'ed out.

```
$john = new Person;
<h2>Name: <?php echo $john->name;?></h2>
```

Notice how the code jumps in and out of Php.

3) Make a new instance

Now try to make a new instance of the class and save it in a variable called $kurt.

Copy/paste the <article> and echo out the name and age of $kurt.

Discuss: Is there a problem having the name and age values defined in the class?

Step 2

Creating an interface for the class

1) The __construct()

Open the file **index2.php**.

Look at the public $name and $age properties. These properties belong to the class.

Discuss: What does it mean that they are public?

Resource: http://www.techflirt.com/tutorials/oop-in-php/visibility-in-php-classes.html

2) Public properties

Look at the defined function inside the class. What is it?

Discuss: How will you describe what the __construct() function does?

Resource: read the first paragraph from http://php.net/manual/en/language.oop5.decon.php

3) Make a function that access the properties

Inside of the class, make a new function called **get_name()**. Make this a public function by added the keyword **public** in front of it.

```
public function get_name() {
}
```

Inside this function, make it return the value of the $name property of the class.
Use the **return** keyword for this. This return makes php go back to where it where and take the value with it.

```
return $this->name;
```

Discuss: What does the return keyword do? What is referred to by the keyword $this?

Resource: http://stackoverflow.com/questions/1523479/what-does-the-variable-this-mean-in-php

Step 3

Making the properties private

1) Change the public properties to private

Change the public $name and $age properties to private.

```
private $name;
private $age;
```

2) Try accessing the private properties

To see how private affects the way php allows you to access the properties of the class, try to echo out the value of the $name property.

```
$kurt = new Person("Kurt Kreol", 42);
echo $kurt->name;
```

Discuss: What does the error message tell you?

3) Add a get_age method

Add another method that will allow you to access the $age property. Look at the get_name() method as basis for this.

4) Write out the name and age

Create two instances of the class and write out their name and age.

5) Try to set the name and age

Try to access and change the value of the name property after you have created the instance.

```
$bob = new Person("Bob Hansen", 26);
$bob->name = "Henry";
```

Discuss: Why can't you do this? Why is this a good thing?

Step 4 [Optional]

Create a new class that extends the Person class.

1) Create a Employee class

Create a new class called Employee. This class must inherit all the methods and properties from the Person class. In Php this is done through the principle of ***inheritance***, where you tell Php to extend the properties and methods of one class to another.

To create a new class that extends an existing class, use the keyword **extends**, e.g.

```
class Employee extends Person {
}
```

2) Create a private $position property

On the Employee class, create a new private property called **$position**.

3) Fill the property from the constructor

In the __construct method, add a parameter that accepts a value for position and assign this parameter to the $position property.

```
public function __construct($name, $age, $position) {
	$this->position = $position;
	$this->name = $name;
	$this->age = $age;
}
```

4) Make a get method for $position

Finally, create a get_position() method that accesses and returns the $position property.

5) Create a new employee

Change the $kurt variable to be an instance of the Employee class instead of Person. Send along a third parameter with his position.

```
$kurt = new Employee("Kurt Kreol", 42, "Regional Sales Manager");
```

Use the get_position() method to echo out his position.

```
<article>
	<h2>Name: <?php echo $kurt->get_name();?></h2>
	<h3>Age: <?php echo $kurt->get_age();?></h3>
	<h3>Position: <?php echo $kurt->get_position();?></h3>
</article>
```