BRAINSTER

Workshop 18: PHP - OOP

Brainster Web Development Academy

Exercise 1

Create a class called Bicycle.
The class should have 5 public properties: $brand, $model, $year, $description and $weight.

Default value for the $description property is “Used bicycle” (hint: you can set it either when declaring the property or through the constructor).

Create getInfo method (a getter) that will return information about the bike in the following format: *“$brand $model ($year)”*.

Create getWeight method that will return weight in grams. Make this method configurable so that it accepts one argument which by default is false. If it is true, the weight should be returned in kilograms and if it is false (default), it should return weight in grams.

Create a setter method for the weight property. The weight property stores the weight in grams.

Create two objects from the Bicycle class and set values for all properties.
Print each bike’s information.
Print each bike’s weight in kilograms.
Print each bike’s weight in grams.

Exercise 2

Create a class called Student.
The class should have:

- 3 public properties: $name, $surname and $country;
- 1 private property: $tuition;
- 1 protected property: $indexNumber.

Create getter methods for the name and the surname of the student.
Create a public method helloWorld() that will return “Hello World” string.
Create a protected method helloFamily() that will return “Hello Family” string.
Create a private method helloMe() that will return “Hello me!” string.
Create a private getter method getTuition() that will print the value of the tuition property.

Do not use a constructor with arguments.

Create a subclass PartTimeStudent.
Add a public method helloParent() that will call the method helloFamily() from the Student class.

Create objects from both the Student and the PartTimeStudent classes, and call all the methods within.

Exercise 3

Create a class called Product.
The class should have 3 properties: $description, $quantity and $price

Create constructor method accepting 3 arguments ($description, $quantity and $price). In the constructor, when setting these arguments, check if the description is a string and if the quantity and price are numbers. If they are not, print an error message.

Create setter and getter methods for the $description, $quantity and $price properties.

Create a method called calculatePrice() that will return the product's price as: $quantity * $price;

Create an object from the Product class. Print all properties in newlines and then print the result from the calculatePrice() method.

Exercise 4

Create an interface HasInfo that will have one abstract method called getInfo().

Create a class called Address that implements the HasInfo interface.
The class should have 3 public properties: street, number and city. Set them through the constructor.
The method getInfo() in this class should return: "Address: street $street, number $number, city $city".

Create a class called Phone that implements the HasInfo interface.
The class should have 2 public properties: prefix and number. Set them through the constructor.
The method getInfo() in this class should return: "Number: $prefix / $number".

Create a class called User that implements the HasInfo interface.
The class should have 2 public properties: name and surname.
The class should have 2 private properties: address and phone (instances from the classes above).
The getInfo() method in this class should call the getInfo() methods from the Address and Phone class respectively. The output of this method should be:

```
"User: $firstName $lastName
Address: street $street, number $number, city $city
Number: $prefix / $number"
```

Create 1 objects from each class.
Call the getInformation method from the User object to see the above output.

Exercise 5

Create 3 classes – User, AdminUser and Customer.

class User

- Should have 3 protected properties: $name , $surname and $username;
- Set their values using a constructor method;
- Add 1 protected property $is_admin. Its default value should be false;
- Create a method that checks if the user is admin;
- Create a method that prints the user's full name. If the user is admin, print `(admin)` at the end.

class Customer

- Should extend the User class;
- Add 3 private properties: $city, $state, $country;
- The Customer's class constructor should have the same parameters as the parent constructor;
- For the other properties create setter and getter methods;
- Create a method location() that returns '$city, $state, $country'.

class AdminUser

- Should extend the User class;
- The constructor should have the same parameters as the parent constructor;
- The constructor should set the value of the $is_admin property to true.

Create objects from each class. Print the full name and is_admin values for each object, and additionally the location (city, state and country) for the customer objects only.

Break a leg

If you haven't finished all exercises, please try to finish them at home.

Hint on using one object as property for another object.

```
class One
{
    public name;

    public function print()
    {
        echo $this->name;
    }
}
```

```
class Two
{
    public $name;
    public $obj;

    public function print()
    {
        echo "{$this->name},"
        $this->obj->print();
    }
}
```

```
$one = new One();
$one->name = "One";

$two = new Two();
$two->name = "Two";
$two->obj = $one;
$two->print();
//Two, One
```