C1- Code & Go

pool_php_d03

Day 03

Welcome in OOP !

CODING ACADEMY
BY EPITECH

Introduction

Starting from today, you're going to learn Object Oriented Programming (OOP). In some previous exercises you may have approached it, but from now on you will use it every day.

To help you in your researches, here are today's notions (in no particular order):

- Classes definition
- Objects and instantiations
- Constructors
- Destructors
- Methods and attributes
- New operator
- Methods and attributes visibility
- Optional parameters

Exercise 01

2pts

Turn in: pool_php_d03/ex_01/Gecko.php
Restrictions: None.

Create a new class “Gecko”.

Every times a new Gecko is created, it shall automatically display “Hello!” followed by a newline.

example.php :

```php
<?php

include_once("Gecko.php");

$alex = new Gecko();
$raph = new Gecko();
```

Terminal

```
~/pool_php_d03> php example.php
Hello !
Hello !
~/pool_php_d03>
```

Exercise 02

3pts

Turn in: pool_php_d03/ex_02/Gecko.php
Restrictions: None.

Copy your "Gecko.php" from Exercise 01. We're going to add a few things to it.

You shall modify your constructor so that it becomes possible to pass a name as parameter for our new Gecko being created.

If a name is specified, it will now display "Hello <Name>!" followed by a newline.

Otherwise, it will still display the same thing as in exercise 01.

This parameter will need to be stocked in an attribute "name" that will be public.

example.php :

```php
<?php

include_once("Gecko.php");

$thomas = new Gecko("Thomas");
$annonymus = new Gecko();

echo $thomas->name;
echo $annonymus->name . "\n";
```

```
Terminal
~/pool_php_d03> php example.php | cat -e
Hello Thomas !$
Hello !$
Thomas$
~/pool_php_d03>
```

Exercise 03

3pts

Turn in: pool_php_d03/ex_03/Gecko.php
Restrictions: None.

Once again, copy your "Gecko.php" from the previous exercise.

From now on, when your Gecko will ceases to exist, you will need to display "Bye <Name> !" followed by a newline.

If they do not have a name, they will simply say "Bye !", followed by a newline.

Ah, speaking of name... From now on, the "name" attribute should not be public, you will need to find yourself what it should be. However, for the name to be available outside our object, we will create our very first method! It will be called "getName" and it will return... the name of the Gecko!

example.php :

```php
<?php

include_once("Gecko.php");

$thomas = new Gecko("Thomas");
$anonymous = new Gecko();
$serguei = new Gecko("Serguei");

unset($serguei);
echo $thomas->getName() . "\n";
echo $anonymous->getName() . "\n";
```

```
Terminal
~/pool_php_d03> php example.php | cat -e
Hello Thomas !$
Hello !$
Hello Serguei !$
Bye Serguei !$
Thomas$

Bye !$
Bye Thomas !$
~/pool_php_d03>
```

Exercise 04

3pts

Turn in: pool_php_d03/ex_04/Gecko.php
Restrictions: None.

Copy your "Gecko.php" from the previous exercise.

Add a new attribute "Age" to our Gecko class. It should be possible to specify it as a second parameter during the construction of the object.

This attribute will have its own getter and setter, respectively "getAge" and "setAge".

Moreover, you will add a new method "status" to our Gecko. This method will take no parameter, and must display a sentence depending on the Gecko's age. You **must** use a 'switch' statement and you are not allowed to use the 'if' keyword in this method.

The method must display the following sentences:

- "Unborn Gecko" if the age is 0.
- "Baby Gecko" if the age is 1 or 2.
- "Adult Gecko" if the age is between 3 and 10.
- "Old Gecko" if the age is between 11 and 13.
- "Impossible Gecko" otherwise.

Each of these sentences must be followed by a newline.

Exercise 05

4pts

Turn in: pool_php_d03/ex_05/Gecko.php
Restrictions: None.

It is time... to give the gift of speech to our Geckos! Firstly, copy your previous "Gecko.php"

Add a new public method to it, called "hello".

When called with a string, it will display "Hello <string>, I'm <Name>!", with <string> being the string given as parameter and <Name> being the name of the Gecko. If our Gecko doesn't have a name it will only display "Hello <string>!".

However, if an integer is given as parameter it will display "Hello, I'm <Name>!" as often as the number given as parameter. Once again, if our Gecko doesn't have a name it will display something else: "Hello !".

Every message must be followed by a newline.

In all other cases, the method does nothing.

example.php :

```
<?php

include_once("Gecko.php");

$dylan = new Gecko("Dylan");
$dylan->hello("Teddy");
$dylan->hello(2);
```

```
Terminal
~/pool_php_d03> php example.php | cat -e
Hello Dylan!$
Hello Teddy, I'm Dylan!$
Hello, I'm Dylan!$
Hello, I'm Dylan!$
Bye Dylan !$
~/pool_php_d03>
```

Exercise 06

3pts

Turn in: pool_php_d03/ex_06/Gecko.php
Restrictions: None.

Copy your "Gecko.php" from the previous exercise.

Add a new method "eat" to our Gecko. It will take a string as parameter and return nothing.

If the value of the string given in parameter is equal "Meat", the Gecko must display the following:

```
Terminal
~/pool_php_d03> Yummy!
```

If the value is equal to "Vegetable", your Gecko must say

```
Terminal
~/pool_php_d03> Erk!
```

If the value is anything else, the Gecko must display

```
Terminal
~/pool_php_d03> I can't eat this.
```

As usual, every sentence will be followed by a newline.

> The 'eat' method **must** be case insensitive.

Moreover, you will add a new attribute "energy" to our Gecko. By default it will be equal to 100.

You will add a setter and a getter for this attribute (getEnergy and setEnergy).

Every time our Gecko eats something, he will win or lose some energy.

If he eats meat he will win 10 Energy. If he eats vegetable he will lose 10 Energy (A Gecko is carnivorous...). In all the other cases his energy will not be modified.

A gecko's energy should always be between 0 and 100 (included)

Exercise 07

2pts

Turn in: pool_php_d03/ex_07/Gecko.php
Restrictions: None.

Copy your "Gecko.php" file from the previous exercise.

You will implement a new method "work" in our Gecko class. It will take no parameter and return nothing.

On every call, if the Gecko has at least 25 energy, it will display "I'm working T.T" followed by a newline, then it will decrease his energy by 9 (He is working 8 hours by day so he needs enough energy to work the whole day).

If the method is called with less than 25 energy, it will display "Heyyy... I'm too sleepy, better take a nap!" followed by a newline, then it will restore 50 energy to our Gecko.

Bonus

3pts

Turn in: pool_php_d03/ex_08/Gecko.php
Restrictions: None.

Let's implement a new method "fraternize" that will take a parameter.

If the parameter is a Gecko Object, our Gecko will be happy and go drink with his friend. It will cost both of them 30 Energy and they will both say "I'm going to drink with <otherName> !".

If one of them doesn't have enough energy, he will say "Sorry <otherName>, I'm too tired for going out tonight..." and the other will answer "Oh ! That's too bad, another time then !". If both of them are too tired to go out, they will both say "Not today !".

The moulinette will not test the case where one of the gecko doesn't have a name, so you don't need to handle this case.

If the parameter is something else than a Gecko, our Gecko will say "No way."

Except if this something else is a "Snake", then if its energy is superior or equal to 10 it must be set to 0 and he will say "LET'S RUN AWAY !!!", if its energy is inferior to 10 it will play dead and display "..."

Finally, if your Gecko has gone out successfully, until he takes a nap, he will need to throw a 6-faced dice for EVERY action he will try to do, and if the result is 1 he will not do his action and instead display "I'm too drunk for that... hic !"

The moulinette will include its own "Snake" class, you shall not turn it in.

You are free to add any method that you think necessary.

Every messages must be followed by a newline.

You shall not have any call to "srand" in your code, the moulinette will call it itself.