B2 - Reboost

B-RBT-200

Backend

PHP - Object Oriented Programming - Inheritance

1.1

{EPITECH.}

Backend

repository name: RBT_back_d04_$ACADEMICYEAR
repository rights: anthony1.palasse@epitech.eu fabien-luc1.lallemand@epitech.eu kevin.cazal@epitech.eu nicolas1.ah-leung@epitech.eu william.grondin@epitech.eu

- Your repository must contain the totality of your source files, but no useless files (binary, temp files, obj files,...).
- Error messages have to be written on the error output, and the program should then exit with the 84 error code (0 if there is no error).

+ Introduction

Today we will continue to go deeper into OOP, you will keep using all the notions from the previous days, and you will also discover a few new things:

- static keyword with methods (also called "static method")
- inheritance
- protected visibility

Inheritance is probably the most important notion in OOP.
Here is a quick example: let's imagine an "Apple" class inheriting from a class "Fruit". This "Apple" class will have its own methods like "Peel" or "Make an applepie" but will also possess all the methods and attributes that weren't private (protected or public) of her mother class (Fruit), like for example a method "Eat".
We can then imagine another Class "Banana" also inheriting from out class "Fruit" that will also possess all the attributes and methods that weren't private in the mother class, and its own methods and attributes.
Finally, there is no limit to inheritance, in this example we could have a "Golden Apple" class inheriting from the class "Apple", it would inherit all the properties that weren't private in the class "Apple" (which mean that it will obviously also inherit the "Fruit" class characteristics).
And this could go on …
On a special note, it is sadly not possible (in PHP) to inherit from two clases at the same time directly (you actually need to trick the language … because PHP) which mean that we could not represent directly a Tangelo (a cross between Tangerines and Grapefruits, google it).
We ***strongly*** advise you to look attentively (even more than usual) at the example on the internet before starting the Task 03.

Unless specified otherwise, all messages must be followed by a newline.
Unless specified otherwise, the getter and setter name will always be following this format "getAttribute" or "setAttribute", if my attribute name is "someWeirdNameAttribute" its getter will be "getSomeWeirdNameAttribute" (for your general knowledge, it's known as "CamelCase").

+ TASK 01

Turn in: task_01/Animal.php
Restrictions: None.

Create a new class "Animal".
This new class will need to have three **const** attributes initialized like this:

- MAMMAL = 0
- FISH = 1
- BIRD = 2

These variables will be used to pass to the constructor the type of our Animal.
The constructor will take three parameters.

- The first parameter will be the name of our animal.
- The second parameter will be the number of legs of our animal.
- The third will be one of the three attributes created previously (MAMMAL, FISH, BIRD) representing the type of our animal.

These three parameters will obviously need to be stocked inside new attributes, respectively named "name", "legs" and "type". You will need to create a getter for each of this attributes.
Even though accidents could happen, we are generous Gods (well, at least, I am!) and will consider that our animals will never lose a leg after creation (And also, because I love cats), that's why there is no setter for this attribute. For the other you will need to think about it yourself, because there are some logical reasons for it.
Be careful, the getter for "type" is a bit tricky, you will need to make it work like in this example (and guess for the others cases).
All those parameters are mandatory.
Finally, but not least, during its creation out Animal will say:

Terminal

```
My name is [name] and I am a [type]!
```

Where [name] is the name if the Animal and [type], its type.
Example:

Terminal

```
~/B-RBT-200> cat index.php
<?php
include_once("Animal.php");
$isidore = new Animal("Isidore", 4, Animal::MAMMAL);
echo $isidore->getName() . " has " . $isidore->getLegs() . " legs and is a " .
$isidore->getType() . ".\n";
```

Terminal

```
~/B-RBT-200> php index.php
My name is Isidore and I am a mammal!
Isidore has 4 legs and is a mammal.
```

+ TASK 02

Turn in: task_02/Animal.php
Restrictions: None.

Copy your "Animal.php" file from the previous Task.
You shall implement a static method in this class called "getNumberOfAnimalsAlive" returning the number of instances of "Animal" and displaying it in a sentence like this:

```
Terminal
There [is/are] currently [x] animal[s] alive in our world.
```

Where [x] should be replaced by the number of animals currently alive, and [s] and [is/are] should be used when necessary.
It is up to you to find how to handle this part.

remember static?

You will also implement three variants of this function "getNumberOfMammals", "getNumberOfFishes" and "getNumberOfBirds", returning respectively the number of mammals, fishes and birds alive in the world, and displaying the following message:

```
Terminal
There [is/are] currently [x] [type][s] alive in our world.
```

Where [x] should be replaced by the number of animals currently alive of this type, and [type] by the type of animal of the function being called and [s] and [is/are] should be used when necessary. An example would be ("There are currently 3 mammals alive in our world.").

In english, 0 is plurial, only 1 is singular. Be careful, here, "0 birds" **but** "0 fish".

+ TASK 03

Turn in: task_03/Cat.php
task_03/Animal.php
Restrictions: None.

Copy your "Animal.php" from the previous Task.
Create a new class "Cat", which shall inherit from "Animal". This class must have a private attribute "color" with a getter and setter.
When creating a "Cat", it must display:

Terminal

```
[name]: MEEEOOWWWW
```

Where [name] shall be replaced by … the name of the cat!
The name of the cat should always be specified as the first parameter of its constructor, however the color can be specified as its second parameter but is not mandatory. Its number of legs will be set to 4 and its type to MAMMAL by default (not parameters).
Add a public method "meow" to your class "Cat". This method does not take any parameter, when calling it, you must display the following message:

Terminal

```
[name] the [color] kitty is meowing.
```

Where [name] shall be replaced by the name of the cat … and [color] by its color.
If no color has been specified during creation, the default color will be grey.
Example:

Terminal

```
~/B-RBT-200> cat index.php
<?php
include_once("Cat.php");
$isidore = new Cat("Isidore", "orange");
echo $isidore->getName() . " has " . $isidore->getLegs() . " legs and is a " .
$isidore->getType() . ".\n";
$isidore->meow();
```

Terminal

```
~/B-RBT-200> php index.php
My name is Isidore and I am a mammal!
Isidore: MEEEOOWWWW
Isidore has 4 legs and is a mammal.
Isidore the orange kitty is meowing.
```

+ TASK 04

Turn in: task_04/Cat.php
task_04/Animal.php
task_04/Shark.php
task_04/Canary.php
Restrictions: None

Copy your classes from the previous Task.
Create a class "Shark" and "Canary" both inheriting from "Animal".
When creating a "Shark", you must display:

```
Terminal
A KILLER IS BORN!
```

The class "Shark" will also have a private attribute named "frenzy".
When the class will be constructed it will receive its name as parameter, its number of legs should be set to 0 and its type to "FISH", its attribute "frenzy" should be on "false" by default.
You must also add a method "smellBlood" to it, it will take a Boolean as parameter and return nothing. This method will change the value of "frenzy" to the value passed as parameter.
Finally you must add a method "status", it will display one of the two following messages depending on frenzy:

- If "frenzy" is true:

  ```
  Terminal
  [name] is smelling blood and wants to kill.
  ```

  where [name] will be replaced by the shark's name.

- If "frenzy" is false:

  ```
  Terminal
  [name] is swimming peacefully.
  ```

  where [name] will be replaced by the shark's name.

Your class "Canary" must have a private attribute "eggs", indicating how many eggs it laid in its life. During initialization "Canary" will only take its name. "legs" will be initialized to 2, "type" to BIRD and "eggs" to 0.
During its creation it will say:

```
Terminal
Yellow and smart? Here I am!
```

You must add a method "getEggsCount" returning the number of eggs laid by the Canary.
And a method "layEgg" that will simply increment the number of egg laid by 1.

Example :

Terminal

```
~/B-RBT-200> cat index.php
<?php
include_once("Shark.php");
include_once("Canary.php") ;
$titi = new Canary("Titi");
$willy = new Shark("Willy"); // Yes , Willy is a shark here!
$willy->status();
$willy->smellBlood(true);
$willy->status();
$titi->layEgg();
echo $titi->getEggsCount() . "\n";
```

Terminal

```
~/B-RBT-200> php index.php
My name is Titi and I am a bird!
Yellow and smart? Here I am!
My name is Willy and I am a fish!
A KILLER IS BORN!
Willy is swimming peacefully.
Willy is smelling blood and wants to kill.
1
```

+ TASK 05

Turn in: task_05/Cat.php
task_05/Animal.php
task_05/Shark.php
task_05/Canary.php
Restrictions: None

Copy your classes from the previous Task.
Willy … our dear shark need to eat.
Add a public method "eat" to your class "Shark". This method will take an "Animal" as parameter and return nothing. When the method is called you should display:

Terminal

```
"[Shark's name] ate a [type's name] named [Animal's name]."
```

Where [Shark's name] will be replaced by the name of the shark, [type's name] will be replaced by the name of the type of the animal being eaten (mammal, fish, bird) and [Animal's name] will be replaced by the name of the animal being eaten.
If the parameter can't be eaten your shark will say:

Terminal

```
[name]: It's not worth my time.
```

Where [name] should be replaced by the name of the shark.
If the attribute "frenzy" on the "Shark" is true and our Shark has successfully eaten it must be set to false after calling this method.
Tip: Your shark will never be self-cannibal.

What happens to an eaten animal?

+ TASK 06

Turn in: task_06/Cat.php
task_06/Animal.php
task_06/Shark.php
task_06/Canary.php
task_06/BlueShark.php
task_06/GreatWhite.php
Restrictions: None

For this Task you need to create two new classes "BlueShark" and "GreatWhite" both inheriting from the "Shark" class. They will both take their name during their construction as mandatory argument.
They will be almost identical to the original class "Shark", except that they are picky eaters…
"BlueShark" will refuse to eat anything except other "Fish" and "GreatWhite" will refuse to eat "Canary", even worse if you try to feed a "Canary" to a "GreatWhite" it will say:

Terminal

```
[name]: Next time you try to give me that to eat, I'll eat you instead.
```

Where [name] should be replaced by the name of the "GreatWhite".

Except for those minor differences, their eat function will do the exact same thing as for the "Shark" Class.

Oh … Almost forgot!
If a "GreatWhite" eat another "Shark" it will display:

Terminal

```
[name]: The best meal one could wish for.
```

Where [name] should be replaced by the name of the "GreatWhite".