ООП. Модуль 2.

В этом модуле мы с вами познакомились с принципами ООП и с наследованиям, давайте применим знания на практике.

Домашнее задание состоит из нескольких небольших отдельных не связанных между собой задач. Каждое задание делается в отдельном файле с уникальным пространством имен.

1. Ферма - абстракция

a. Опишите следующие классы
   - Cow - Коровка
   - Pig - Хрюшка
   - Chicken - Курица
   - Farm - Ферма

b. Ферма должна содержать свойство public $animals - массив с животными, а также метод public function addAnimal($animal) - который добавит животное на ферму.

c. Каждое животное умеет говорить public function say() и ходить public function walk(), создайте и реализуйте эти методы. Метод say() - должен выводить голос каждого животного. Метод walk() - должен выводить "топ-топ", для каждого из животных.

d. При заселении на ферму животное должно пойти.

e. Создайте метод перекличка на ферме public function rollCall() - в котором каждое животное на ферме покажет свой голос в случайном порядке.

f. Создайте класс - абстракцию для данной задачи. Обратите внимание на условие в этой задачи, волей-неволей эта абстракция в явном виде присутствует в условии задачи. Вынесите как можно больше функционала в абстракцию - а реализации сделайте наследниками этой абстракции.
g. Поселите корову, двух хрюшек и курицу на ферму. Проведите их перекличку.

2. Ферма 2.0

a. Добавим к созданным в предыдущем задании классам следующие классы:
    - Goose - Гусь
    - Turkey - Индюк
    - Horse - Лошадь
    - BirdFarm - Птичья ферма
    - Farmer - фермер
b. Теперь у нас есть две фермы, одна будет для птиц, другая для животных с копытцами (hoof).
c. Создайте дополнительные уровни абстракции для животных (копытные и птицы). При этом птицы при ходьбе должны пытаться взлететь, а не топать как другие животные. У них должен быть метод function tryToFly(), который выводит "Вжих-Вжих-топ-топ". Скорректируйте наследования всех животных на более точные, и реализуйте недостающие методы
d. На птицеферме у нас строгий учет свободных мест, поэтому сразу после заселения туда новой птицы, Ферма должна сообщать суммарное количество птиц на ней, для этого добавьте метод function showAnimalsCount(), который выводит на экран "Птиц на ферме: <кол-во птиц>"
e. За расселение животных и учет на фермах теперь частично отвечает фермер. Добавьте фермеру методы

- function addAnimal(Farm $farm, Animal $animal) - метод должен заселить на ферму $farm, животное $animal
- function rollCall(Farm $farm) - метод должен вызывать перекличку на ферме $farm

f. Создайте объекты: фермер, ферма, птицеферма. Создайте в указанном порядке и прикажите фермеру заселить следующих животных: Корову, две хрюшки, Курицу, три Индейки, двух Лошадей и одного Гуся

g. После этого заставьте этого лентяя устроить перекличку всех животных на обеих фермах.

3. Черный ящик - Инкапсуляция

a. Создайте классы
   - BlackBox - черный ящик, у него должно быть закрытое от всех свойство private $data
   - Plane - Самолет, должен содержать закрытое свойство private $blackBox, в конструкторе это свойство должно быть проинициализировано новым классом BlackBox
   - Engineer - Инженер - дешифровщик черных ящиков.

b. Класс BlackBox содержит следующие методы:
   - public function addLog($message) - добавляет очередную строку в свое свойство $data
   - public function getDataByEngineer(Engineer $engineer) - возвращает свои данные для инженера.

c. Класс Plane должен содержать методы:
   - public function flyAndCrush()
     {
       $this->flyProcess();

```
        $this->crushProcess();
    }
```

- flyProcess - процесс полета может проходить по-другому для других самолетов, пишет лог в черный ящик, придумайте что будет записано в этом методе в черный ящик.
- crushProcess - процесс крушения переопределен быть не может, пишет лог в черный ящик, придумайте что будет записано в этом методе в черный ящик.
- protected function addLog($message) - передает сообщение для записи в лог черного ящика.
-
```
    public function getBoxForEngineer(Engineer $engineer)
    {
        $engineer->setBox($this->blackBox);
    }
```

d. Реализуйте класс Engineer
   - public function setBox(BlackBox $blackBox) - устанавливает черный ящик для дешифрации у инженера
   - public function takeBox(Plane $plane) - должен доставать черный ящик из самолета (посмотрите какой подходящий метод есть в классе Plane)
   - public function decodeBox() - декодирует черный ящик - выводит на экран лог черного ящика
e. Реализуйте методы без изменения области видимости методов и свойств.
f. Создайте самолет, устройте ему полет с крушением.
g. Создайте инженера, возьмите черный ящик из упавшего самолета и дешифруйте его.

h. Создайте новый вид самолета (наследуйтесть от Plane). Самолет должен вести другой лог во время полета. Но, к сожалению, путь его тот же что и для предыдущего самолета. Дешифруйте и его лог.

4. Домна - полиморфизм

a. Была, кажется, такая игра - где в печи нужно было сжигать вещи, и чем лучше это все горит, тем больше дают очков, давайте сделаем частичку этой игры. Создайте следующие классы
   - Forge - печь, должен содержать один метод:

```
public function burn($object)
{
    $flame = $object->burn();
    echo $flame->render((string)$object) . PHP_EOL;
}
```

   - BlueFlame, RedFlame, Smoke - голубое пламя, красное пламя и дым. Должны содержать один метод public function render($name), возвращающий текст (каждый класс со своим текстом), например для класса Smoke:

```
public function render($name)
{
    return $name . " лишь задымился";
}
```

b. Создайте не менее 5 различных классов, которые будут сжигаться в печи, каждый класс должен реализовывать два метода:

- public function burn() - должен возвращать один из объектов BlueFlame, RedFlame, Smoke, на ваш выбор
- public function __toString() - должен возвращать строку - название текущего объекта.

  Постарайтесь придумать никак не связанные между собой объекты, например, пианино и больная голова ;) чтобы ощутить именно эффект полиморфизма, а не наследования.

c. Создайте печь и посжигайте в ней свои объекты. А затем создайте новый класс по той же схеме и насладитесь мощью полиморфизма - спалив и его экземпляр в печи.