ООП. Модуль 3. Домашнее задание

Мы с вами прошли практически все особенности работы с классами в php, давай-те применим их 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. Корзина с мячами

a. Наша программа будет класть в корзину мячики, при этом корзина не знает сколько в ней мячей - ведь чтобы нам узнать сколько мячей в корзине - их нужно пересчитать вручную, т..е. снова достать и положить, либо считать при складывании. Но у нас есть учет мячей, перед тем как положить мяч в корзину - мяч маркируется.
b. Создайте класс Box - корзина, в этом классе будет только один метод, никаких свойств классу добавлять нельзя
    - public function putBall(Ball $ball) - должен вывести на экран фразу: 'В корзину добавлен мяч'
c. Создайте класс Ball - мяч, создайте статическое свойство $count. При создании нового мяча - количество должно увеличиваться.
d. Создайте корзину, положите в нее случайное число мячиков (функция rand()), а затем выведите на экран количество мячей, которое лежит в корзине.

2. Фабрика котов

a. Да, котов производят на фабрике!

b. Создайте класс Cat - кот, у кота должны быть три свойства имя, цвет, возраст, все они должны быть инициализированы в конструкторе
c. Создайте класс CatFactory - этот класс будет создавать котов, с помощью именованных конструкторов, например вот такой:

```
public static function createBlack8YearsOldCat($name)
 {
    return new Cat($name, 'black', 8);
 }
```

d. Обратите внимание название конструктора фабрики точно совпадает с параметрами создаваемого кота.
e. Создайте еще 6 различных статичных конструкторов в фабрике, каждый из которых будет создавать котов с разными параметрами. Должны быть конструкторы, которые предустанавливают только 1 параметр, например createBlackCat(подумайте какие здесь параметры), при создании кота все свойства должны быть проинициализированы значениями
f. Создайте массив котов, заполните его, используя все созданные конструкторы и выведите на экран (например функцией print_r())

3. В мире животных

a. Создайте классы Рыба, Тигр, Медведь, Лось, Змея, Курица, Верблюд, Слон, Дельфин
b. Придумайте для указанных животных два уровня абстракции, базовый уровень - Animal - животное
c. Уровни абстракции - должны быть абстрактными классами.

d. Правильно расставьте наследование в созданных классах
e. Класс Animal должен содержать абстрактный метод move()
f. Все животные должны реализовать метод move() - который выводит каким образом передвигается указанное животное

4. Импорт/Экспорт с интерфейсами

a. Создайте следующие интерфейсы
    - Reader - драйвер чтения - содержит один метод read(): array - который читает и возвращает данные в виде массива
    - Writer - драйвер записи - содержит один метод write(array $data) - который принимает данные в виде массива для записи
    - Converter - конвертация строки данных - содержит один метод convert($item) - конвертирует один элемент массива и возвращает результат конвертации.
b. Создайте класс импорта Import, со свойствами $reader, $writer и $converters = []
    - public function from(Reader $reader) - устанавливает значение свойства $reader и возвращает $this
    - public function to(Writer $writer) - устанавливает значение свойства $writer и возвращает $this
    - public function with(Converter $converter) - Добавляет конвертер в свойство $converters и возвращает $this
    - public function execute() - производит импорт/экспорт данных из $reader в $writer
c. Реализуйте описанные методы.

d. Согласно разработанному коду импорт можно произвести, примерно, такой конструкцией
```
(new Import()) // Создаем новый объект - импорт
    ->from(new YourReader())    // Регистрируем в импорте reader - как будем читать
    ->to(new YourWriter())          // Регистрируем в импорте writer - куда будем писать
      ->with(new YourConverter()) // Регистрируем в импорте сколько угодно конвертеров - как данные будут обработаны перед записью
    ->with(new YourConverter())
    ->execute()
;
```
e. В конструкции классы YourReader, YourWriter и YourConverter - это названия для демонстрации кода, под ними подразумеваются ваши реализации соответствующих интерфейсов. Названия ваших реализация должны быть осмысленными, например ArrayReader, но точно не YourReader.
f. Создайте свои реализации Reader и Writer, например читать из файла или массива, и писать в файл, в массив, в строку, в сессию куда угодно. Создайте свои реализации Converter'а.
g. Проведите свой импорт/экспорт.