ООП. Модуль 5

Наконец-таки мы познакомились со всеми основами ООП, и можем в полной мере приступить к нашей CMS.

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

1. Ленивый калькулятор

a. Создайте класс Calculator, содержащий только один метод public static function calculate($number1, $number2, callable $callback) - метод должен возвращать результат вычисления для чисел, при этом сам ничего не считая.
b. создайте массив $callbacks, содержащий 4-ре различных вида значений для типа callable (повторяться не должны, например может быть только одна анонимная функция и т.п). Каждый из этих callable должен выполнять одно из 4-х математических операций (сложение, вычитание, умножение, деление).
c. Создайте несколько пар чисел и выведите результат каждой из математической операции, используя Calculator::calculate()
d. Пример выполнения программы:
   Пара чисел: 5 10
   15
   -5
   50
   0.5

2. Валидация с помощью исключений

a. Создайте простую html-страницу с веб-формой, в форме должны быть поля name - имя, age - возраст, email - email.
b. При отправке формы если валидация пройдена, то должно быть выведено сообщение об успешном изменении, при ошибке валидации - должна быть выведена соответствующая ошибка
c. Код для обработки запроса формы должен выглядеть так:

```
$success = false;
if (! empty($_POST)) {
    try {

        $success = (new UserFormValidator())->validate($_POST);

    } catch (\Exception $e) {
        $error = $e->getMessage();
    }
}
```

d. Создайте класс UserFormValidation - реализуйте метод validate.
e. Требования к валидации:
    - имя должно быть не пустым
    - возраст должен быть не менее 18 лет
    - email - должен быть заполнен и соответствовать формату email

3. Валидация 2.0

a. Добавим нашей программе созданной в задании выше имитацию работы с базой данных.
b. Создайте класс User, класс должен содержать два метода
    - public function load($id) - метод должен формировать исключение если $id - не найден в базе данных (придумайте условие на $id, для имитации этой ошибки)
    - public function save($data) - метод имитирует сохранение в базе данных возвращает true или false при ошибке - для имитации работы метод должен возвращать случайное значение
c. Добавьте в форму поле id
d. Доработайте код
    - Перед валидацией нужно проверить, что пользователь существует в базе данных
    - после успешной валидации, необходимо попробовать сохранить пользователя в базе данных.