**MySql**

Пройти курс <https://php.zone/php-i-mysql-s-nulya/rabota-s-mysql>

Залить на гит результат работы в курсе (выборка из базы данных пользователей)   
В гите должен быть php файл с логикой + дамп базы данных

**ООП в php**

Ознакомится с теорией ООП в php

Далее выполнить задания

**Задание 1**

Пять небольших задач на работу с классами и объектами. Каждое задание выполните в отдельном файле, все классы в этом файле должны быть в уникальном пространстве имён.

    1.1 Домашний зоопарк

**Описание классов**

Опишите следующие классы:

* Cat — кот.
* Dog — собака.
* Fish — рыбка.

Каждый класс должен содержать одно свойство **public** $name — кличка животного. Это свойство должно быть проинициализировано в конструкторе. При этом никаких значений по умолчанию у него быть не должно.

**Клиентский код**

Создайте двух котов, двух собак и одну рыбку. Поместите их в массив и выведите клички всех животных.

    1.2. Голодный кот

**Описание классов**

Опишите класс HungryCat — голодный кот.  Этот класс должен иметь три свойства: кличка, цвет и любимая еда. Все эти свойства должны быть проинициализированы в конструкторе и не должны иметь значений по умолчанию.

**Клиентский код**

Создайте и реализуйте метод **public function** **eat**($food). Внутри этого метода должна выводиться строка такого формата:

"Голодный кот <кличка кота>, особые приметы: цвет - <цвет кота>, съел <название еды>"

Если кот съел любимую еду, то нужно к этой строке добавить в конце:

"  и замурчал 'мррррр' от своей любимой еды"

Создайте двух совершенно разных голодных котов и накормите каждого тремя разными блюдами. Оба котика обязательно должны помурчать.

    1.3. Фабрика игрушек

**Описание классов**

Опишите два класса:

* ToyFactory — фабрика игрушек. У этого класса не должно быть никаких свойств.
* Toy — игрушка. У игрушки должно быть два свойства: название и цена. Эти свойства должны быть проинициализированы в конструкторе и не должны иметь значений по умолчанию.

Реализуйте метод создания игрушки на фабрике **public function createToy**($name), который должен возвращать новый экземпляр игрушки со случайной ценой.

**Клиентский код**

Создайте массив строк с названиями игрушек, содержащий пять разных названий.

Создайте случайное количество игрушек (воспользуйтесь функцией rand(int min, int max)): не менее пяти, но не более 20 штук. Название игрушки должно браться случайным образом из массива с названиями игрушек.

Выведите на экран для каждой игрушки следующую строку:

"<Название игрушки> - <Стоимость игрушки>"

А в конце выведите на экран итоговую строку:

"Итого - <Сумма стоимостей всех игрушек>"

    1.4. Уведомления для клиента

**Описание классов**

Опишите классы:

* User — пользователь, содержит четыре свойства: Ф. И. О., email, телефон и возраст. Эти свойства должны быть проинициализированы в конструкторе, при этом поля «телефон» и «возраст» не обязательны для заполнения.
* Censor — цензура. У этого класса нет никаких свойств.
* Notification — уведомление. У него одно свойство — канал уведомления, которое должно быть проинициализировано в конструкторе.

**Класс User**

Этот класс должен содержать следующие методы:

* **public function getName**() — метод должен возвращать Ф. И. О. пользователя.
* **public function getEmail**() — метод должен возвращать email пользователя.
* **public function getPhone**() — метод должен возвращать телефон пользователя.
* **public function isAdult**() — метод должен возвращать boolean-значение, есть ли пользователю 18 лет или нет.

**Класс Censor**

Этот класс должен содержать следующий метод:

* **public function censor**($message) — метод должен производить цензуру переданного ему сообщения и возвращать результат этой обработки.

**Класс Notification**

Этот класс должен содержать следующие методы:

* **public function sendTo**($name, $contact, $message) — метод должен формировать и выводить строку вида:

"Уведомление клиенту: <name> на <канал уведомления> (<contact>): <message>"

**Клиентский код**

Создайте функцию notify(User $user, $message). Эта функция принимает два параметра:

* User $user — объект класса User.
* $message — текст уведомления.

Эта функция отправляет уведомление (или уведомления) переданному в качестве параметра пользователю. Если пользователю нет 18 лет, то функция должна применить цензуру для сообщения, создав объект класса Censor и вызвав его метод. После этого функция должна создать новый объект уведомления с каналом email и отправить уведомление пользователю. Если у пользователя также указан телефон, то функция должна создать ещё одно уведомление и отправить его на телефон.

Создайте двух разных пользователей и отправьте им уведомления (вызывая функцию notify). Для одного укажите телефон и возраст больше 18 лет, а для другого не указывайте ни телефон, ни возраст.

1.5 Интернет-магазин

**Описание классов**

* Order — заказ, содержит два свойства: корзина и цена доставки. Эти свойства должны быть проинициализированы в конструкторе.
* Basket — корзина, содержит одно свойство: массив позиций товаров. При создании объекта это свойство содержит пустой массив.
* BasketPosition — позиция одного товара в заказе, содержит два свойства: товар и количество. Эти свойства должны быть проинициализированы в конструкторе.
* Product — товар, содержит два свойства: название и цена. Эти свойства должны быть проинициализированы в конструкторе.

**Класс Order**

Этот класс должен содержать следующие методы:

* **public function getBasket**() — возвращает объект-корзину, которая хранится в заказе.
* **public function getPrice**() — возвращает общую стоимость заказа. Сумма стоимости корзины и стоимости доставки заказа.

**Класс Basket**

Этот класс должен содержать следующие методы:

* **public function addProduct**(Product $product, $quantity) — создаёт новую позицию и добавляет её в корзину, объединение количества товаров можно не реализовывать.
* **public function getPrice**() — возвращает стоимость всех позиций в корзине.
* **public function describe**() — выводит или возвращает информацию о корзине в виде строки:

"<Наименование товара> - <Цена одной позиции> - <Количество>"

**Класс BasketPosition**

Этот класс должен содержать следующие методы:

* **public function getProduct**() — возвращает наименование товара в этой позиции.
* **public function getQuantity**() — возвращает количество товаров в этой позиции.
* **public function getPrice**() — возвращает стоимость позиции.

**Класс Product**

Этот класс должен содержать следующие методы:

* **public function getName**() — возвращает наименование товара.
* **public function getPrice**() — возвращает стоимость товара.

Опишите все требуемые классы и реализуйте их методы.

**Клиентский код**

Создайте корзину, заполните её товарами. Создайте заказ на основе этой корзины таким образом:

$order = new Order($basket);

Выведите информацию о корзине этого заказа и выведите общую стоимость заказа.

"Заказ, на сумму: <сумма заказа> Состав: <информация о корзине>"

Создайте нового клиента — «Николай Николаича» и отправьте ему уведомление, используя классы из предыдущего задания (подключив их с помощью оператора require; если у вас выводится текст из предыдущего задания, то просто не обращайте на это внимание. Если хотите, чтобы текст не выводился, то нужно объявление классов в задании 4 вынести в отдельный файл и их использование в другой). Текст уведомления:

"Для вас создан заказ, на сумму: <сумма заказа> Состав: <информация о корзине>"

**Задание 2**

Три задачи на применение принципов ООП на практике. Каждое задание выполните в отдельном файле, все классы в этом файле должны быть в уникальном пространстве имён.

    2.1. Ферма. Абстракция

**Описание классов**

* Animal — животное.
* Bird — птица, наследник класса Animal.
* Farm — ферма, содержит одно свойство $animals — массив животных на ферме.
* BirdFarm — птичья ферма, наследник класса Farm.
* Farmer — фермер.

**Класс Animal**

Этот класс должен содержать следующие методы:

* **public function getName**() — возвращает название вида животного, например: конь, курица, собака.
* **public function isBird**() — возвращает boolean-значения, является ли животное птицей. Для класса Animal всегда возвращает false.

**Класс Bird**

Этот класс должен переопределить метод isBird:

* **public function isBird**() — возвращает boolean-значения, является ли животное птицей. Для класса Bird всегда возвращает true.

**Класс Farm**

Этот класс должен содержать следующие методы:

* **public function addAnimal**(Animal $animal) — добавляет животное на ферму.
* **public function rollCall**() — вызывает перекличку животных. Перекличка выполняется для всех животных на ферме в случайном порядке. Для каждого животного должна выводиться строка:

"На ферме обитает: <вид животного>"

**Класс BirdFarm**

Этот класс должен переопределить метод rollCall:

* **public function rollCall**() — должен сначала вывести строку с количеством птиц на птицеферме, а затем провести перекличку. Строка с количеством птиц должна выглядеть следующим образом:

"Птиц на птицеферме: <кол-во птиц>"

**Класс Farmer**

Этот класс — некий менеджер, который передаёт наши команды на фермы. Должен содержать следующие методы:

* **public function addAnimal**(Farm $farm, Animal $animal) — добавляет животное на указанную ферму.
* **public function rollCall**(Farm $farm) — вызывает перекличку животных на указанной ферме.

**Наследники классов Animal и Bird**

Создайте несколько классов, наследников животных и птиц.

**Клиентский код**

Создайте по одному экземпляру классов Farm, BirdFarm, Farmer. Создайте массив с животными, в нём должны быть вперемешку и птицы, и не птицы.

Расселите одним циклом всех животных на фермы с помощью фермера. Если животное — это птица, то фермер должен поселить его на птицеферму. Если животное не птица, то фермер должен поселить его на обычную ферму.

Затем попросите фермера провести перекличку на этих фермах.

2.2 Чёрный ящик. Инкапсуляция

**Описание классов**

* BlackBox — чёрный ящик. Содержит одно приватное свойство  **private** array $data = []. В нём будет храниться лог записей полёта самолёта.
* Plane — самолёт. Содержит одно приватное свойство **private** BlackBox $blackBox. В конструкторе это свойство должно быть заполнено новым экземпляром класса BlackBox.
* Engineer — инженер. Содержит одно приватное свойство **private** int $accessLevel. Это свойство должно быть проинициализировано в конструкторе.

**Класс BlackBox**

Этот класс должен содержать следующие методы:

* **public function addLog**(string $message) — добавляет новую запись в лог, при этом к строке $message в начале должен дописаться порядковый номер этой записи в логе.
* **public function getData**(int $accessLevel): array — проверяет уровень доступа и возвращает массив с логами ящика.

Если уровень доступа меньше или равен единице, то тогда возвращается только одна запись:

"Доступ запрещен"

Если уровень доступа больше единицы, но меньше или равен трём, то тогда возвращается две записи: первая запись из лога ящика и ещё одна запись:

"Ваш уровень доступа не позволяет получить больше данных"

В остальных случаях возвращаются все записи из чёрного ящика.

**Класс Plane**

Этот класс должен содержать следующие методы:

* **public function flyAndCrush**() — имитирует процесс полёта самолёта и его крушения (запишите различные строковые записи в лог чёрного ящика).
* **private function addLog**(string $message) — добавляет запись в чёрный ящик. При этом в начале каждой записи дописывается метка времени в формате: "Y.m.d H:i:s".
* **public function getBlackBox**(): BlackBox — возвращает значение свойства blackBox.

**Класс Engineer**

Этот класс должен содержать следующий метод:

* **public function decodeBox**(BlackBox $blackBox) — пробует дешифровать чёрный ящик и построчно выводит результат дешифровки.

**Клиентский код**

Создайте самолёт, пусть он полетает и разобьётся. После этого возьмите из него чёрный ящик. Создайте инженера со случайным значением уровня доступа от 1 до 5 и попробуйте расшифровать чёрный ящик из разбившегося самолёта.

2.3. Домна. Полиморфизм

Описание классов

* Forge — печь.
* BlueFlame, RedFlame, Smoke — голубое пламя, красное пламя и дым.

**Класс Forge**

Этот класс должен содержать один метод:

* **public function burn**($object) — метод сжигает указанный предмет, вызывая его метод burnWith($flame), а в качестве параметра передавая случайным образом один из объектов BlueFlame, RedFlame, Smoke.

**Классы BlueFlame, RedFlame, Smoke**

Эти классы должны содержать один метод:

* **public function getFlame**() — метод возвращает одну из строк соответственно для каждого класса:
  + "горит синим пламенем"
  + "ярко горит"
  + "лишь задымился"

**Другие классы**

Создайте несколько классов, не являющихся ничьими наследниками. Постарайтесь придумать никак не связанные между собой объекты, например: пианино, жизнь и больная голова. Каждый из этих классов должен содержать один метод:

* **public function burnWith**($flame) — метод должен выводить строку вида:

"<название предмета> <как горит>"

например:

"Домашнее задание горит синим пламенем"

**Клиентский код**

Создайте массив из предметов и сожгите их все в печи.

**Задание 3**

Ознакомится с ключевым словом static   
<https://www.php.net/manual/ru/language.oop5.static.php>

В рамках этого задания ничего выполнять не нужно, просто ознакомьтесь для понимания синтаксиса

**Битрикс**   
Изучить курс

https://academy.1c-bitrix.ru/training/course/7785/  
Затем в отдельном файле я пришлю задание на реализацию простого корп сайта на битриксе