



# PHP OOP

[sd@freematiq.com](mailto:sd@freematiq.com)

<https://github.com/freematiq/php>



# Тип данных Object

Класс - тип переменной

Переменная класса (instance) может содержать набор свойств (значений) и операций (методов, функций) для работы с НИМИ.



# class

```
class User
{
    public $name, $password;

    function isAuthValid($name, $password) {
        if ($name === $this->name &&
            $password === $this->password) {
            return true;
        }
        return false;
    }
}
```



# \$this

Используется для обращения к переменным и методам объекта (инстанса)



# Создание класса

```
$var = new ClassName();
```

Объект работает по ссылке !  
И в передаче функций

```
function f(User $user);  
function f(User $user = null); // значение по умолчанию
```



# Конструктор

```
public function __construct($params, ...) {  
}
```

Параметры по умолчанию:

```
__construct($p1, $p2 = 1, $p3 = 2) {  
    $this->p1 = $p1;  
    $this->p2 = $p2;  
}
```

```
$user = new User('p1', 'p2');
```



# Доступ к методам и переменным

Через переменную:

```
$objInstance->name;
```

```
$objInstance ->login();
```

Внутри класса:

```
$this->name;
```

```
$this->login();
```



# Права доступа

`private` — доступ только из методов класса

`public` — открытый доступ

`protected` — доступ только для наследования

По умолчанию - `public`

```
$name = $user->name;
```





# Getters, setters

```
class User
{
    public $name;

    public function getName() {
        return $this->name;
    }

    public function setName($name) {
        $this->name = $name;
    }
}
```



# Что нельзя

Делать перегрузку методов

Делать перегрузку операторов



# Namespacing

- namespace ...; идёт первой строкой
- Пространства отделяются \
- Имя файла должно совпадать с именем класса
- Имя пространства должно учитывать вложенность директорий
- Для глобальных классов используется пространство \Exception, \PDO



# use

```
use %имя%\%имя;
```

```
use %имя%\..\ as Псевдоним;
```

```
use %имя%\{Class1, Class2};
```



# composer init

Создаём composer.json, composer.lock для проекта

composer install - установка проекта

dev раздел - для библиотек разработки

composer install —no-dev



# composer autoload

```
"autoload": {  
    "psr-4": {  
        "Freematiq\\": "src"  
    }  
},
```

Начать использовать автозагрузку  
`require "vendor/autoload.php";`



# Итого

- ООП надо выучить
- Объекты - хранилища значений и методов
- Обращение к методам и значениям через `$this`
- Создание объекта - ссылка и передача по ссылке
- Использовать `getters & setters`
- Пространства имён для изоляции кода
- Автозагрузка для классов с помощью `composer`
- Использование `composer` для библиотек



# Домашнее задание

Используя решение с прошлого урока (сессии и вход), написать решение с использованием ООП для контроллера, роутера и модели