Классы и объекты в PHP

Введение

Объектно-ориентированное программирование, или **ООП** — это одна из парадигм разработки. **Парадигмой** называют набор правил и критериев, которые соблюдают разработчики при написании кода. Если представить, что код — это рецепт блюда, то парадигма — то, как рецепт оформлен в кулинарной книге. Парадигма помогает стандартизировать написание кода. Это снижает риск ошибок, ускоряет разработку и делает код более читабельным для других программистов.

Суть понятия объектно-ориентированного программирования в том, что все программы, написанные с применением этой парадигмы, состоят из **объектов**.

Каждый объект — это определённая сущность со своими **данными** (свойствами) и набором доступных **действий** (методами).

> ***Примечание***. Сравните с **таблицей** базы данных, где каждая запись представляет собой объект некоторой сущности, обладающий набором свойств.

Например, для интернет-магазина необходимо вывести каталог товаров. Руководствуясь принципами ООП, в первую очередь нужно создать **объекты**: карточки товаров. Потом заполнить (написать код) эти

карточки **данными**: названием товара и его свойствами. И уже после этого написать код доступных **действий** для объектов (карточек товаров) – взаимодействие с пользователем, обновление, изменение и пр.

> ***Примечание***. Основная идея ООП заключается в том, чтобы представить программу в виде **совокупности взаимодействующих объектов**, каждый из которых является экземпляром определённого **класса**.

В программировании есть несколько основных **парадигм** — подходов, которые определяют, как программисты структурируют и организуют свой код.

Процедурное программирование — самый простой и прямолинейный подход. Программы организованы как цепочки **команд** или **процедур**, выполняемых последовательно. Например, кулинарный рецепт очень похож на программу, написанную в парадигме процедурного программирования: ведь рецепт — это, по сути, набор инструкций, которые нужно выполнить строго по порядку, чтобы получить готовое блюдо.

Есть **функциональное программирование**. Этот подход основан на концепции **пользовательских функций**, которые принимают входные данные и возвращают результат. Допустим, вам необходимо выполнить запрос в базу данных и вывести в браузер карточку искомого товара. В функциональном программировании каждый из этих шагов может быть функцией, причем каждая функция всегда возвращает один и тот же результат при одних и тех же входных данных.

Обычно написать функцию быстрее, чем создавать объекты и прописывать взаимодействие между ними. Но если объём кода большой, работать с разрозненными функциями сложно.

> ***Примечание***. В силу того, что процедурой иногда называют функцию, которая не возвращает значение, **процедурный** и **функциональный** стиль программирования иногда объединяют в одну **парадигму**.

Одним из наиболее популярных подходов сейчас является **объектно-ориентированное программирование**, ООП, где код организуется вокруг объектов и классов. Объектно-ориентированный подход поддерживается многими языками программирования и начинающим разработчикам стоит детальнее разобраться с тем, что же на практике означают страшные слова «полиморфизм», «инкапсуляция» и другие термины, применяемые в этой парадигме.

> ***Примечание***. Часто встречаемая ошибка начинающего разработчика – не зная принципов функционального программирования приступать к изучению объектов. Так делать нельзя!

Практика функционального программирования

Но перед изучением объектов, выполним практику функционального программирования прилагающихся демонстрационных примеров.

Реализуем учебное приложение, реализованное на базе данных **db_distance**. База данных представляет информацию о педагогах и выдаваемых ими курсах некоторого **образовательного заведения**.

Приложение реализовано на связанных между собой **функциях**, выполняющих **поиск** в базе данных и **вывод** в браузер данных по указанной фамилии **педагога** или его **идентификатору**.

В любом языке программирования существуют пользовательские функции. **Функция** — это специальным образом оформленный **именованный** фрагмент кода, к которому можно **многократно обратиться из любого места внутри программы**.

> ***Примечание***. **Алгоритм**, реализующий функционал приложения с помощью набора пользовательских функций называется **процедурным** или **функциональным**.

Последовательно выполним программирование следующих функций:

1. **Функция** поиска педагога по фамилии:
    - **getPerson**($surname)
2. **Функция** подсчета количества образований педагога по его id:
    - **getCountEducations**($id)
3. **Функция** поиска образований педагога по его id:
    - **getEducations**($id)
4. **Функция** подсчета количества курсов, которые ведет педагог по его id:
    - **getCountCourses**($id)
5. **Функция** поиска курсов, которые ведет педагог по его id:
    - **getCourses**($id)
6. **Функция** поиска персональных данных педагога по его id:
    - **getPersonData**($id)

7. **Функция** извлечения персональных данных педагога из JSON-объекта:

    - **getDataFromJSON**($data)

Форматный вывод в данной теме нас не интересует, вывод в браузер организуем через конструкцию **print_r**() (для понимания основных процессов этого будет достаточно).

Даже если вам хорошо понятен принцип реализации функционального программирования, выполнение следующих демонстрационных примеров **обязательно**.

> ***Важно.*** Весь функционал приложения, реализованного с помощью функций, **будет перенесен в методы класса** и реализован с помощью **объектов**.

Функция поиска педагога

Реализуем **функцию** поиска педагога по фамилии:

- **getPerson**($surname);

Для следующих функций будем использовать полученный среди прочих данных **идентификатор педагога** (id_personnel).

example_1.

```php
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
```

```
    $person = getPerson($surname);
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
    echo "</pre>";
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        // доступ к пременной подключения через массив $GLOBALS
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбор строки из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
?>
```

Функция подсчета количества образований педагога

Функция подсчета количества образований педагога по его id:

- **getCountEducations**($id);

example_2.

```
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
```

```php
// критерий поиска - фамилия
$surname = 'Трост';
// сохранение массива данных в переменную
$person = getPerson($surname);
// вывод основной информации по искомому педагогу
echo "<pre>";
echo "<b>Основная информация:</b><br>";
print_r($person);
echo "</pre>";
// вывод количества образований педагога
echo "<pre>";
echo "<b>Количество образований:</b><br>";
print_r(getCountEducations($person['id_personnel']));
echo "</pre>";
// функция поиска педагога по фамилии
function getPerson($surname){
    // выполнение запроса
    // доступ к пременной подключения через массив $GLOBALS
    $result = $GLOBALS['mysqli']->query("SELECT * FROM
    `personnel` WHERE `surname` = '$surname'");
    // выбор строки из набора
    $row = $result->fetch_assoc();
    // возврат значения
    return $row;
};
// функция подсчета количества образований педагога по его id
function getCountEducations($id){
```

```
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT COUNT(*) AS
        `count` FROM `education` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
?>
```

Функция поиска образований педагога

Функция поиска образований педагога по его id:

- **getEducations**($id);

example_3.

```
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
    $person = getPerson($surname);
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
    echo "</pre>";
```

```php
    // вывод образований педагога
    echo "<pre>";
    echo "<b>Образования педагога:</b><br>";
    print_r(getEducations($person['id_personnel']));
    echo "</pre>";
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        // доступ к пременной подключения через массив $GLOBALS
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбор строки из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция поиска образований педагога по его id
    function getEducations($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `education` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
?>
```

Функция подсчета количества курсов, которые ведет педагог

Функция подсчета количества курсов, которые ведет педагог по его id:

- **getCountCourses**($id);

example_4.

```php
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
    $person = getPerson($surname);
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
    echo "</pre>";
    // вывод количества курсов педагога
    echo "<pre>";
    echo "<b>Количество курсов, которые ведет педагог:</b><br>";
    print_r(getCountCourses($person['id_personnel']));
    echo "</pre>";
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        // доступ к пременной подключения через массив $GLOBALS
```

```
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбор строки из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция подсчета количества курсов, которые ведет педагог
    по его id
    function getCountCourses($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT COUNT(*) AS
        `count` FROM `courses` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
?>
```

Функция поиска курсов, которые ведет педагог

Функция поиска курсов, которые ведет педагог по его id:

- **getCourses**($id);

example_5.

```
<?php
    // подключение к серверу СУБД
```

```php
$mysqli = new mysqli('localhost', 'root', '', 'db_distance');
// критерий поиска - фамилия
$surname = 'Трост';
// сохранение массива данных в переменную
$person = getPerson($surname);
// вывод основной информации по искомому педагогу
echo "<pre>";
echo "<b>Основная информация:</b><br>";
print_r($person);
echo "</pre>";
// вывод курсов педагога
echo "<pre>";
echo "<b>Курсы педагога:</b><br>";
print_r(getCourses($person['id_personnel']));
echo "</pre>";
// функция поиска педагога по фамилии
function getPerson($surname){
    // выполнение запроса
    // доступ к пременной подключения через массив $GLOBALS
    $result = $GLOBALS['mysqli']->query("SELECT * FROM
    `personnel` WHERE `surname` = '$surname'");
    // выбор строки из набора
    $row = $result->fetch_assoc();
    // возврат значения
    return $row;
};
// функция поиска курсов, которые ведет педагог по его id
```

```
    function getCourses($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `courses` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
?>
```

Функция поиска персональных данных педагога

Функция поиска персональных данных педагога по его id:

- **getPersonData**($id);

example_6.

```
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
    $person = getPerson($surname);
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
```

```php
    echo "</pre>";
    // вывод персональных данных педагога
    echo "<pre>";
    echo "<b>Персональные данные:</b><br>";
    print_r(getPersonData($person['id_personnel']));
    echo "</pre>";
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция поиска персональных данных педагога по его id
    function getPersonData($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `person_data` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
?>
```

Функция извлечения персональных данных педагога из JSON-объекта

Функция извлечения персональных данных педагога из JSON-объекта:

- **getDataFromJSON**($data);

example_7.

```
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
    $person = getPerson($surname);
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
    echo "</pre>";
    // вывод персональных данных педагога
    echo "<pre>";
    echo "<b>Персональные данные:</b><br>";
    print_r(getPersonData($person['id_personnel']));
    // echo "</pre>";
    // вывод данных текущего педагога вспомогательной функцией
    getDataFromJSON()
    echo "<b>Дополнительная информация:</b><br>";
    echo 'Логин: ' . getDataFromJSON('login') . '<p>';
```

```
    echo 'Пароль:  ' . getDataFromJSON('pwd') . '<p>';
    echo 'E-mail: ' . getDataFromJSON('email');
    echo "</pre>";
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция поиска персональных данных по его id
    function getPersonData($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `person_data` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // создадим и инициализируем
        // вспомогательную глобальную переменную
        $GLOBALS['json_person_data']
        $GLOBALS['json_person_data'] = json_encode($row);
        // возврат значения
        return $row;
    };
    // функция извлечения данных из JSON-объекта вспомогательной
```

```
    переменной $GLOBALS['json_person_data']
    function getDataFromJSON($data){
        // global $json_person_data;
        // $arr = json_decode($json_person_data, 1);
        $arr = json_decode($GLOBALS['json_person_data'], 1);
        // возврат значения
        return $arr[$data];
    }
?>
```

Подводим итоги

Подведем итоги, соберем все функции вместе и выведем в отдельный файл:

- **function.php**

Подобным образом в дальнейшем будем выносить **описание классов в отдельные файлы**.

example_8. index.php

```
<?php
    // подключение к серверу СУБД
    $mysqli = new mysqli('localhost', 'root', '', 'db_distance');
    // подключение файла функций
    include "function.php";
    // критерий поиска - фамилия
    $surname = 'Трост';
    // сохранение массива данных в переменную
    $person = getPerson($surname);
```

```php
    // вывод основной информации по искомому педагогу
    echo "<pre>";
    echo "<b>Основная информация:</b><br>";
    print_r($person);
    // вывод количества образований педагога
    echo "<b>Количество образований:</b><br>";
    print_r(getCountEducations($person['id_personnel']));
    // вывод образований педагога
    echo "<b>Образования:</b><br>";
    print_r(getEducations($person['id_personnel']));
    // вывод количества курсов педагога
    echo "<b>Количество курсов:</b><br>";
    print_r(getCountCourses($person['id_personnel']));
    // вывод курсов педагога
    echo "<b>Курсы:</b><br>";
    print_r(getCourses($person['id_personnel']));
    // вывод персональных данных педагога
    echo "<b>Персональные данные:</b><br>";
    print_r(getPersonData($person['id_personnel']));
    // вывод данных текущего педагога
    echo "<b>Дополнительная информация:</b><br>";
    echo getDataFromJSON('login') . '<p>';
    echo getDataFromJSON('pwd') . '<p>';
    echo getDataFromJSON('email');
    echo "</pre>";
?>
```

example_8. function.php

```php
<?php
    // функция поиска педагога по фамилии
    function getPerson($surname){
        // выполнение запроса
        // доступ к пременной подключения через массив $GLOBALS
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `personnel` WHERE `surname` = '$surname'");
        // выбор строки из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция подсчета количества образований педагога по его id
    function getCountEducations($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT COUNT(*) AS
        `count` FROM `education` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // возврат значения
        return $row;
    };
    // функция поиска образований педагога по его id
    function getEducations($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
```

```
        `education` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
    // функция подсчета количества курсов, которые ведет педагог
    по его id
    function getCountCourses($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT COUNT(*) AS
        `count` FROM `courses` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
    // функция поиска курсов, которые ведет педагог по его id
    function getCourses($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `courses` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_all(MYSQLI_ASSOC);
        // возврат значения
        return $row;
    };
    // функция поиска персональных данных педагога по его id
```

```
    function getPersonData($id){
        // выполнение запроса
        $result = $GLOBALS['mysqli']->query("SELECT * FROM
        `person_data` WHERE `id_personnel` = $id");
        // выбираем строку из набора
        $row = $result->fetch_assoc();
        // создадим и инициализируем
        // вспомогательную глобальную переменную
        $GLOBALS['json_person_data']
        $GLOBALS['json_person_data'] = json_encode($row);
        // возврат значения
        return $row;
    };
    // функция извлечения данных из JSON-объекта вспомогательной
    переменной $GLOBALS['json_person_data']
    function getDataFromJSON($data){
        $arr = json_decode($GLOBALS['json_person_data'], 1);
        // возврат значения
        return $arr[$data];
    };
?>
```

Таким образом, **в хорошо работающем приложении** мы получили набор **слабо связанных** между собой **функций**. И это с учетом того, что происходит обработка данных **одной сущности** – педагога образовательного учреждения.

Если в малом и среднем (по объему кода) приложениях наличие кода разрозненных функций не критично, то на возможность

структурирования больших проектов такое положение вещей скажется самым отрицательным образом.

Администрирование и поддержка проектов масштабов реального приложения с использованием функционального программирования будут весьма затруднительны или невозможны.

Ведь реальное приложение скорее всего потребует:

- учета слушателей курсов;
- системы регистрации как педагогов, так и слушателей;
- наличие системы обратной связи: отзывов, комментариев;
- системы учета оплаты курсов (кредиты, просроченности и пр.);
- и др.

Объектно-ориентированный подход предлагает следующее решение: так же как в базе данных информация **разбивается на сущности** и хранится в разных таблицах, так и в программном коде разбиваем обрабатываемые данные по категориям и **храним в разных объектах**.

Но в отличии от объектов, хранящихся в строках таблицы, каждый объект программного кода будет содержать не только **набор свойств**, но и **набор присущих ему действий**. Например, рассматриваемый нами функциональный код может быть преобразован в объект, который должен уметь выполнять следующие действия:

- поиск педагога по фамилии;
- подсчет количества образований педагога;
- поиск образований педагога;
- и др.

Значит ли сказанное, что каждой таблице базы данных должен соответствовать свой программный объект?

- **Да**, на этапе погружения в объектно-ориентированное программирование, такой подход может быть применен.

- **Нет**, совсем не обязательно. Все зависит от предметной области, поставленных задач, структуры базы данных и прочих зависимостей.

> ***Примечание***. Трудно дать однозначный **совет** по проектированию объектов. Объект это все что угодно из сущностей реального мира. **Объект — это способ мышления** ...

Структура ООП

В коде, написанном по парадигме **объектно-ориентированного программирования** (ООП), выделяют четыре основных элемента:

- **Класс**
- **Объект**
- **Свойство**
- **Метод**

Класс

Класс является ключевым понятием в объектно-ориентированном программировании (ООП). Класс — это общая абстракция, которая описывает структуру объектов.

Класс в объектно-ориентированном программировании, представляет собой **шаблон кода** для создания объектов, обеспечивающий начальные значения состояний:

- инициализация свойств-переменных (атрибутов);
- реализация поведения функций (методов).

> ***Важно***. Шаблон, на базе которого можно построить объект в программировании называется **классом**.

Например, у интернет-магазина может быть класс **Товар**, который описывает набор **свойств** и **методов** товаров магазина. И уже из класса создаются конкретные **объекты** (ноутбук, телефон, планшет ...).

Но товар слишком общее понятие, товар "Ноутбук" и товар "Холодильник" слишком разная **категория** товаров. Поэтому, классы могут **наследоваться** друг от друга.

Например, есть общий класс **Товар** и вложенные классы, или **подклассы**:

- класс **Цифровые устройства**;
- класс **Бытовая техника**;
- класс **Одежда**;
- и др.

В свою очередь класс **Цифровые устройства** может иметь подклассы:

- класс **Телефон**;
- класс **Компьютер**;
- класс **Смартфон**;
- и др.

Класс **Бытовая техника** может иметь свои подклассы:

- класс **Телевизор**;
- класс **Холодильник**;
- класс **Пылесос**;
- и др.

Подкласс может **наследовать** свойства из родительского класса, например: **производителя**, **цену** товара, **скидку** на товар, **количество** штук на складе. При этом может иметь свои свойства, например, диагональ дисплея для класса **Ноутбука** или количество сим-карт для класса **Смартфон**.

Все объекты принадлежат общему классу **stdClass**. **stdClass** – пустой класс общего назначения с **динамическими** свойствами.

Объект

Объект в объектно-ориентированном программировании (ООП) — некоторая сущность в цифровом пространстве, обладающая определённым состоянием и поведением, имеющая определённые:

- свойства (атрибуты);
- операции над свойствами (методы).

Как правило, при рассмотрении объектов выделяется то, что **объекты принадлежат** одному или нескольким **классам**, которые определяют поведение (являются шаблоном) объекта.

Объект – фрагмент кода, который описывает элемент с конкретными характеристиками и функциями. Карточка товара на странице интернет-магазина может быть отдельным объектом.

Кнопки **Перейти в корзину** и **Заказать** — тоже объекты (но другого класса).

В объектно-ориентированном программировании объект обрабатывается так же, как ссылки или указатели. Это значит, что каждая новая переменная содержит **ссылку на объект**, а **не копию всего объекта**.

> ***Важно***. Каждая новая переменная содержит **ссылку на объект**, а **не копию всего объекта** (разберем подробно).

Классы и объекты. Что что?

Для **визуальной** демонстрации понятия **класса** и **объекта** используем автомобильную тему

Создание любого технически сложного изделия начинается с проектной документации, раскрывающей сущность изделия, определяющей архитектурные, конструктивные и инженерно-технические решения этого изделия.

Например, создаваемый автомобиль сначала существует в голове конструктора и на многочисленных **чертежах**, **схемах**, **рисунках**. Так вот – это и есть **класс.** У каждого класса автомобиля есть **техническое описание** множества реализуемых им свойств, характеристик и возможностей: длина, высота, клиренс, колесная база, крутящий момент и прочее.

Компания **Mitsubishi** имеет техническую документацию на большой набор **классов** автомобилей: Pajero, Pajero Sport, Outlander ....

На рисунке представлен фрагмент технического описания **класса Mitsubishi Pajero**.

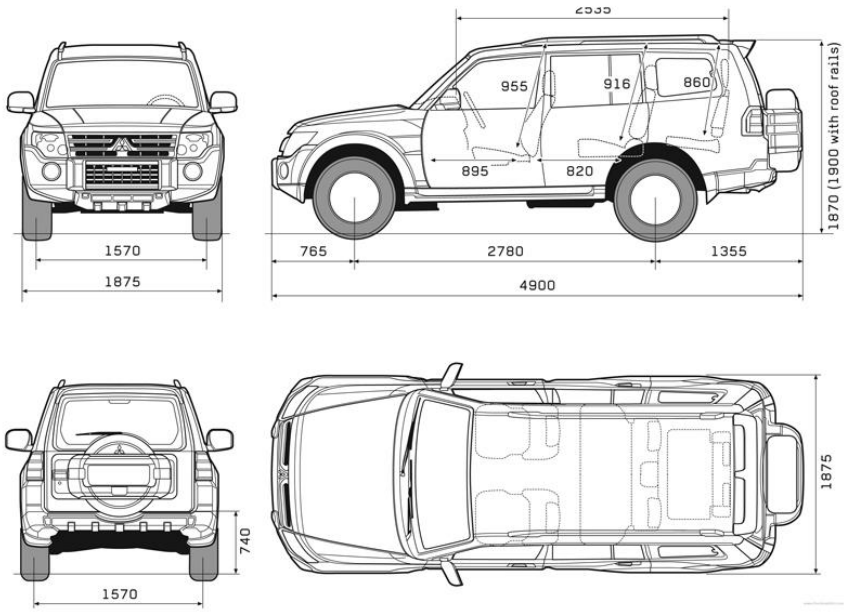

Рис.1. **Класс** Mitsubishi Pajero

Готовый автомобиль называется **объектом**, или говорят – э**кземпляром класса**.

На рисунке 2 представлен **объект** (экземпляр класса) Mitsubishi Pajero.

Рис.2. **Объект** Mitsubishi Pajero

На рисунке 3 еще один **объект** (экземпляр класса) Mitsubishi Pajero.

Рис.3. Другой **объект** Mitsubishi Pajero

Оба объекта выполнены согласно одному техническому описанию, что, впрочем, не мешает им иметь некоторые разные свойства (цвет, комплектация, обивка сидений и другие).

Объектов **Mitsubishi Pajero** может быть сколько угодно. Объект **физически существует**, его можно потрогать, на нем можно ездить (в отличии от похожего и даже очень красивого, но нарисованного на ватмане класса).

Свойство

Свойство (атрибут) – характеристика объекта в программировании — например, цена, производитель или объём оперативной памяти. В классе прописывают, что такие свойства есть, а в объектах с помощью методов заполняют эти свойства данными.

> ***Примечание***. В разных источниках **свойство** объекта может также называться **атрибутом** или **полем**.

Метод

Метод – функция **внутри объекта** или класса, которая позволяет взаимодействовать с текущим объектом или другой частью кода.

В примере с рассматриваемым образовательным учреждением, класс будет содержать следующие методы:

- поиск педагога по фамилии;
- подсчет количества образований педагога;
- поиск образований педагога;
- и др.

В примере с товарами класс может содержать методы:

- заполнение карточки товара нужной информацией;
- обновление количества товара в наличии;
- сравнение выбранных товаров между собой;

- и др.

Основные принципы ООП

Объектно-ориентированное программирование базируется на трёх основных принципах, которые обеспечивают удобство использования этой парадигмы:

- **Инкапсуляция**
- **Наследование**
- **Полиморфизм**

> ***Примечание***. Каждый из принципов будет продемонстрирован большим количеством демонстрационных примеров.

Инкапсуляция

Вся информация, которая нужна для работы конкретного объекта, должна храниться **внутри этого объекта**. Если нужно вносить изменения, методы для этого тоже должны лежать в самом объекте — посторонние объекты и классы этого делать не могут. Для внешних объектов доступны только публичные атрибуты и методы.

Например, метод для заполнения данными карточки товара должен обязательно быть прописан в классе "Карточка товара". А не в классе "Корзина" или "Каталог товаров".

Такой принцип обеспечивает безопасность и не даёт повредить данные внутри какого-то класса со стороны. Ещё он помогает избежать случайных зависимостей, когда из-за изменения одного объекта что-то ломается в другом.

Наследование

В этом принципе — вся суть объектно-ориентированного программирования.

Разработчик создаёт:

- **Класс** с определёнными свойствами;
- **Подкласс** на его основе, который берёт свойства класса и добавляет свои;
- **Объект подкласса**, который также копирует его свойства и добавляет свои.

Каждый дочерний элемент наследует методы и атрибуты, прописанные в родительском. Он может использовать их все, отбросить часть или добавить новые. При этом заново прописывать эти атрибуты и методы не нужно.

Например, в каталоге товаров:

- У класса **Товар** есть следующие атрибуты:
    - **Дата** выпуска
    - **Название**
    - **Цена**
    - **Изображение**

    а также следующие методы:

    - **Вывести карточку**
    - **Обновить цену**

- Подкласс **Цифровые устройства** берёт все атрибуты и методы родительского класса **Товар**, плюс добавляет **свои атрибуты**:

    - **Операционная система**
    - **Центральный процессор**
    - **Графический процессор**

- Подкласс **Ноутбук** берет атрибуты и методы родительского класса **Цифровые устройства** и добавляет свои:

- **Диагональ экрана**
- **Тип экрана**
- **Размер клавиатуры**

Таким образом строится огромная (в некоторых случаях) древовидная структура наследования классов. Принцип наследования рассмотрим отдельной темой.

Полиморфизм

Один и тот же метод может работать по-разному в зависимости от объекта, где он вызван, и данных, которые ему передали. Например, метод **Удалить** при вызове в корзине удалит товар из корзины, а при вызове в карточке товара — удалит саму карточку из каталога.

То же самое с объектами. Можно использовать их публичные методы и атрибуты в других функциях и быть уверенным, что всё сработает нормально.

Этот принцип ООП, как и другие, обеспечивает отсутствие ошибок при использовании объектов.

Действия с объектами

Работа с объектами в PHP включает в себя создание объектов, присваивание объектов переменным, копирование и клонирование объектов, а также проверку типа объекта. Рассмотрим эти аспекты более подробно.

- Создание экземпляра класса (объекта)
- Присваивание объектов переменным
- Копирование и клонирование объектов
- Проверка типа объекта
- Уничтожение объекта

> ***Примечание***. Подробно каждое из действий разберем в следующей теме.

Создание экземпляра класса (объекта)

Создание класса

Каждое определение класса начинается с ключевого слова **class**, затем идёт **имя класса**, а потом пара фигурных скобок, в которых определяют **свойства** и **методы** класса.

Для имени класса разрешается выбирать любое слово, при условии, что слово не входит в **список зарезервированных** слов PHP, начинается с буквы или символа подчёркивания и за которым следует любое количество букв, цифр или символов подчёркивания.

```
<?php
    // определение класса Person
    class Person {
        // PHP код
}
```

Классы могут содержать **переменные** и **константы**, которые в классах называют **свойствами**.

```
<?php
    // определение класса Person
    class Person {
        // свойство класса
        public $id_personnel = "значение 1";
        // константа класса
```

```
        const MYSQLI = "значение 2";
```

Классы также могут содержать **функции**, которые в классах называют **методами**.

```
<?php
    // определение класса Person
    class Person {
        // свойство класса
        public $id_personnel = "значение 1";
        // константа класса
        const MYSQLI = "значение 2";
        // метод класса
        // функция поиска педагога
        public function getPerson() {
            // PHP код
        }
    }
```

Создание объекта

Для создания объекта из класса в PHP используется ключевое слово **new** за которым следует имя класса и круглые скобки.

Для того чтобы создать **объект** класса **Person** используем следующий **синтаксис**:

```
// создание экземпляра класса (объекта)
$person = new Person();
```

example_9.

```
<?php
```

```php
    // определение класса Person
    class Person {
        // свойства класса
        public $id_personnel = "значение 1";
        // константа класса
        const MYSQLI = "значение 2";
        // методы класса
        // функция поиска педагога
        public function getPerson() {
            // PHP код
        }
    }
    // создание экземпляра класса (объекта)
    $person = new Person();
    // вывод объекта
    echo "<pre>";
    var_dump($person);
    // print_r($person);
    echo "</pre>";
?>
```

Тогда полный код **шаблона** рассматриваемого в теме класса **Person** может иметь следующий вид.

example_10.

```php
<?php
    // определение класса Person
    class Person
```

```
{
    // свойства класса
    public $id_personnel = "значение 1";
    public $surname = 'значение 2';
    public $name = 'значение 3';
    // и др.
    // константа класса
    const MYSQLI = "значение ...";
    // методы класса
    // функция поиска педагога
    public function getPerson() {
        // PHP код
    }
    // функция подсчета количества образований
    public function getCountEducations() {
        // PHP код
    }
    // функция поиска образований
    public function getEducations() {
        // PHP код
    }
    // функция подсчета количества курсов
    public function getCountCourses() {
        // PHP код
    }
    // функция поиска курсов
    public function getCourses() {
```

```
            // PHP код
        }
        // функция поиска персональных данных
        public function getPersonData() {
            // PHP код
        }
        // функция извлечения данных из JSON-объекта
        public function getDataFromJSON() {
            // PHP код
        }
    }
    // создание экземпляра класса (объекта)
    $person = new Person();
    // вывод объекта
    echo "<pre>";
    var_dump($person);
    // print_r($person);
    echo "</pre>";
?>
```

К сведению. **Объекты** класса могут быть инициализированы следующими способами:

- оператором **new**;
- созданы с помощью **преобразования** в объект;
- созданы с помощью специальных **функций**.

Подробнее в следующей теме.

Хорошей практикой является хранение классов в **отдельных директориях** и **отдельных файлах**.

example_11. index.php

```php
<?php
    // подключаем класс Team
    include "Team.php";
    // создание экземпляра класса (объекта)
    $person = new Person();
    // вывод объекта
    echo "<pre>";
    print_r($person);
    echo "</pre>";
?>
```

example_11. Team.php

```php
<?php
    // определение класса Person
    class Person
    {
        // свойства класса
        public $id_personnel = "значение 1";
        public $surname = 'значение 2';
        public $name = 'значение 3';
        // и др.
        // константа класса
        const MYSQLI = "знчение ...";
```

```
        // методы класса
        // функция поиска педагога
        public function getPerson() {
        // PHP код
        }
        // функция подсчета количества образований
        public function getCountEducations() {
        // PHP код
        }
        // функция поиска образований
        public function getEducations() {
        // PHP код
        }
        // функция подсчета количества курсов
        public function getCountCourses() {
        // PHP код
        }
        // функция поиска курсов
        public function getCourses() {
        // PHP код
        }
        // функция поиска персональных данных
        public function getPersonData() {
        // PHP код
        }
        // функция извлечения данных из JSON-объекта
        public function getDataFromJSON() {
```

```
        // PHP код

        }

    }
```