Запись на курсы по HTML, CSS, JavaScript, PHP, фреймворкам и CMS,
а также: помощь в поиске работы и заказов, стажировка на реальных проектах→

Бесплатные курсы по React для новичков. Начало 4-го ноября→ Конкурс CSS картинок. Тема: Хэллоуин. Призовой фонд: 100$. Подробности→

Использование классов внутри других классов

Бывает такое, что мы хотели бы использовать методы одного класса внутри другого, но не хотели бы наследовать от этого класса.

Почему мы не хотим наследовать?

Во-первых, используемый класс может являться вспомогательным и по логике нашего кода может не подходить на роль родителя.

Во-вторых, мы можем захотеть использовать несколько классов внутри другого класса, а с наследованием это не получится, ведь в PHP у класса может быть только один родитель.

Давайте посмотрим на практическом примере. Пусть у нас дан следующий класс **Arr**, в объект которого мы можем добавлять числа с помощью метода **add**:

```php
<?php
  class Arr
  {
    private $nums = []; // массив чисел

    // Добавляем число в массив:
    public function add($num)
    {
      $this->nums[] = $num;
    }
  }
?>
```

Давайте теперь добавим в наш класс метод, который будет находить сумму квадратов элементов и прибавлять к ней сумму кубов элементов.

Пусть у нас уже существует класс **SumHelper**, имеющий методы для нахождения сумм элементов массивов:

```php
<?php
  class SumHelper
  {
    // Сумма квадратов:
    public function getSum2($arr)
    {
      return $this->getSum($arr, 2);
    }

    // Сумма кубов:
    public function getSum3($arr)
    {
      return $this->getSum($arr, 3);
    }

    // Вспомогательная функция для нахождения сумм:
     private function getSum($arr, $power) {
      $sum = 0;

      foreach ($arr as $elem) {
        $sum += pow($elem, $power);
      }

      return $sum;
    }
```

```
  }
?>
```

Логично будет не реализовывать нужные нам методы еще раз в классе **Arr**, а воспользоваться методами класса **SumHelper** внутри класса **Arr**.

Для этого в классе **Arr** создадим объект класса **SumHelper** внутри конструктора и запишем его в свойство **sumHelper**:

```
<?php
  class Arr
  {
    private $nums = []; // массив чисел
    private $sumHelper; // сюда запишется объ
      ект класса SumHelper

    // Конструктор класса:
    public function __construct()
    {
      // Запишем объект вспомогательного класса в свойство:
      $this->sumHelper = new SumHelper;
    }

    // Добавляем число в массив:
    public function add($num)
    {
      $this->nums[] = $num;
    }
  }
?>
```

Теперь внутри **Arr** доступно свойство **$this->sumHelper**, в котором хранится объект класса **SumHelper** с его публичными методами и свойствами (если бы публичные свойства были, сейчас их там нет, только публичные методы).

Создадим теперь в классе **Arr** метод **getSum23**, который будет находить сумму квадратов элементов и прибавлять к ней сумму кубов элементов, используя методы класса **SumHelper**:

```
<?php
  class Arr
  {
    private $nums = []; // массив чисел
    private $sumHelper; // сюда запишется объ
      ект класса SumHelper

    // Конструктор класса:
    public function __construct()
    {
      $this->sumHelper = new SumHelper;
    }

    // Находим сумму квадратов и кубов элементов мас
      сива:
     public function getSum23()
    {
      // Для краткости запишем $this->nums в пер
        еменную:
      $nums = $this->nums;

      // Найдем описанную сумму:
      return $this->sumHelper->getSum2($nums) + $
        this->sumHelper->getSum3($nums);
     }

    // Добавляем число в массив:
    public function add($number)
    {
      $this->nums[] = $number;
```

```
27      }
28    }
29  ?>
```

Давайте воспользуемся созданным классом **Arr**:

```php
<?php
  $arr = new Arr(); // создаем объект

  $arr->add(1); // добавляем в массив число 1
  $arr->add(2); // добавляем в массив число 2
  $arr->add(3); // добавляем в массив число 3

  // Находим сумму квадратов и кубов:
  echo $arr->getSum23();
?>
```

Задача 24.1

Самостоятельно повторите описанные мною классы **Arr** и **SumHelper**.

Задача 24.2

Создайте класс **AvgHelper** с методом **getAvg**, который параметром будет принимать массив и возвращать среднее арифметическое этого массива (сумма элементов делить на количество).

Задача 24.3

Добавьте в класс **AvgHelper** еще и метод **getMeanSquare**, который параметром будет принимать массив и возвращать среднее квадратичное этого массива (квадратный корень, извлеченный из суммы квадратов элементов, деленной на количество).

Задача 24.4

Добавьте в класс **Arr** метод **getAvgMeanSum**, который будет находить сумму среднего арифметического и среднего квадратичного из массива **$this->nums**.