Класс как набор методов

Часто классы используются просто как набор некоторых методов, сгруппированных вместе. В этом случае нам не нужно создавать много объектов этого класса, а достаточно всего одного.

Для примера давайте сделаем класс **ArraySumHelper**, который предоставит нам набор методов для работы с массивами. Каждый метод нашего класса будет принимать массив, что-то с ним делать и возвращать результат.

Пусть у нас будет следующий набор методов:

```php
<?php
  class ArraySumHelper
  {
    // Сумма элементов массива:
    public function getSum1($arr)
    {

    }

    // Сумма квадратов элементов массива:
    public function getSum2($arr)
    {

    }

    // Сумма кубов элементов массива:
    public function getSum3($arr)
    {

    }

    // Сумма четвертых степеней элементов массива:
    public function getSum4($arr)
    {

    }
  }
?>
```

Давайте посмотрим, как мы будем пользоваться нашим классом:

```php
<?php
  $arraySumHelper = new ArraySumHelper; // создаем объект

  $arr = [1, 2, 3];
  echo $arraySumHelper->getSum1($arr); //
    найдем сумму элементов
  e cho $arraySumHelper->getSum2($arr); //
    найдем сумму квадратов элементов
  e cho $arraySumHelper->getSum3($arr); //
    найдем сумму кубов элементов
  e cho $arraySumHelper->getSum4($arr); //
    найдем сумму 4-тых степеней элементов
?>
```

Вот еще пример - найдем сумму квадратов элементов массива и сумму кубов и сложим результат вместе:

```php
<?php
  $arraySumHelper = new ArraySumHelper;

  $arr = [1, 2, 3];
  echo $arraySumHelper->getSum2($arr) + $ar
    raySumHelper->getSum3($arr);
?>
```

То есть фактически мы получаем набор функций, просто сгруппированных в одном классе.

Однако, в отличие от обычного набора функций, мы можем пользоваться преимуществами ООП - например, делать вспомогательные методы приватными, чтобы они не были доступны извне класса.

Давайте приступим к написанию кода нашего класса.

Если обдумать реализацию методов, то становится очевидно, что они будут фактически одинаковыми, отличие будет только в степени, в которую будут возводится элементы нашего массива. Вот код, иллюстрирующий это:

```php
<?php
  class ArraySumHelper
  {
    public function getSum1($arr)
    {
      $sum = 0;

      foreach ($arr as $elem) {
        $sum += $elem; // первая степень элемента - э
          то сам элемент
      }

      return $sum;
    }

    public function getSum2($arr)
    {
      $sum = 0;

      foreach ($arr as $elem) {
        $sum += pow($elem, 2); // возведем во
          вторую степень
      }

      return $sum;
    }

    public function getSum3($arr)
    {
      $sum = 0;

      foreach ($arr as $elem) {
        $sum += pow($elem, 3); // возведем в т
          ретью степень
      }

      return $sum;
    }

    public function getSum4($arr)
    {
      $sum = 0;

      foreach ($arr as $elem) {
        $sum += pow($elem, 4); // возведем в ч
          етвертую степень
      }

      return $sum;
    }

```

```
    }
?>
```

Вместо того, чтобы реализовывать каждый метод заново, давайте лучше сделаем вспомогательный приватный метод **getSum**, который параметрами будет принимать массив и степень и возвращать сумму степеней элементов массива:

```php
<?php
  private function getSum($arr, $power) {
    $sum = 0;

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

    return $sum;
  }
?>
```

Итак, давайте поменяем методы нашего класса с использованием нового метода **getSum**:

```php
<?php
  class ArraySumHelper
  {
    public function getSum1($arr)
    {
      return $this->getSum($arr, 1);
    }

    public function getSum2($arr)
    {
      return $this->getSum($arr, 2);
    }

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

    public function getSum4($arr)
    {
      return $this->getSum($arr, 4);
    }

    private function getSum($arr, $power) {
      $sum = 0;

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

      return $sum;
    }
  }
?>
```

Наш класс **ArraySumHelper** больше учебный, чем реальный, но тут вам важно понять принцип - то, что часто некоторый класс может использоваться просто как набор методов и при этом создается только один объект класса. В дальнейшем мы будем разбирать более жизненные (но и более сложные) примеры.

Задача 18.1

Напишите реализацию методов класса **ArrayAvgHelper**, заготовки методов которого расположены ниже:

```php
<?php
  class ArraySumHelper
  {
    /*
```

```php
    Находит сумму первых
    степеней элементов массива:
  */
  public function getAvg1($arr)
  {

  }

  /*
    Находит сумму вторых степеней
    элементов массива и извлекает
    из нее квадратный корень:
  */
  public function getAvg2($arr)
  {

  }

  /*
    Находит сумму третьих степеней
    элементов массива и извлекает
    из нее кубический корень:
  */
  public function getAvg3($arr)
  {

  }

  /*
    Находит сумму четвертых степеней
    элементов массива и извлекает
    из нее корень четвертой степени:
  */
  public function getAvg4($arr)
  {

  }

  /*
    Вспомогательный метод, который параметром
    принимает массив и степень и возвращает
    сумму степеней элементов массива:
  */
  private function getSum($arr, $power)
  {

  }

  /*
    Вспомогательный метод, который параметром
    принимает целое число и степень и возвращает
    корень заданной степени из числа:
  */
  private function calcSqrt($num, $power)
  {

  }
}
?>
```

Математическая подсказка: корень первой степени - это само число. То есть **calcSqrt(число, 1)** должно просто вернуть само число.

А корень любой степени можно найти с помощью функции **pow**, просто параметром передав ей дробь. Например, **pow(число, 1/3)** - так найдем корень третьей степени.