Статические методы

При работе с классами можно делать методы, которые для своего вызова не требуют создания объекта. Такие методы называются *статическими*.

Чтобы объявить метод статическим, нужно после модификатора доступа (то есть после public, private или protected) написать ключевое слово **static**, пример:

```php
<?php
  class Test
  {
    // Статический метод:
    public static function method()
    {
      return '!!!';
    }
  }
?>
```

Чтобы обратиться к статическому методу, нужно написать имя класса, потом два двоеточия и имя метода, объект класса при этом создавать не надо, вот так:

```php
<?php
  echo Test::method(); // выведет '!!!'
?>
```

Пример

Давайте рассмотрим статические методы на более практическом примере.

Пусть у нас дан вот такой математический класс **Math** (пока без статических методов):

```php
<?php
  class Math
  {
    // Находит сумму:
    public function getSum($a, $b)
    {
      return $a + $b;
    }

    // Находит произведение:
    public function getProduct($a, $b)
    {
      return $a * $b;
    }
  }
?>
```

Давайте воспользуемся нашим классом:

```php
<?php
  $math = new Math; // создаем объект класса
  echo $math->getSum(1, 2) + $math->getProduct(3,
    4); // используем методы
?>
```

Как мы видим, для того, чтобы использовать методы класса, мы должны создать объект этого класса.

Я думаю, вы уже заметили, что наш класс **Math** представляет собой просто набор методов и, фактически, нам нужен только один объект этого класса.В таком случае удобно объявить методы класса статическими и вообще не создавать объект этого класса, а сразу использовать его методы.

Итак, давайте объявим метод **getSum** и метод **getProduct** статическими, указав им ключевое слово **static**:

```php
<?php
  class Math
  {
    public static function getSum($a, $b)
    {
      return $a + $b;
    }

    public static function getProduct($a, $b)
    {
      return $a * $b;
    }
  }
?>
```

Воспользуемся методами нашего класса:

```php
<?php
  echo Math::getSum(1, 2) + Math::getProduct(3,
    4);
?>
```

Как вы видите, теперь объект создавать не нужно, и наш код стал немного короче.

Статические методы внутри класса

Если вы хотите использовать статические методы внутри класса, то к ним следует обращаться не через **$this->**, а с помощью **self::**.

Для примера добавим в наш класс **Math** метод **getDoubleSum**, который будет находить удвоенную сумму чисел.

Используем внутри нового метода **getDoubleSum** уже существующий метод **getSum**:

```php
<?php
  class Math
  {
    // Найдем удвоенную сумму:
    public static function getDoubleSum($a,
       $b)
     {
       return 2 * self::getSum($a, $b); // используем дру
         гой метод
     }

    public static function getSum($a, $b)
    {
      return $a + $b;
    }

    public static function getProduct($a, $b)
    {
      return $a * $b;
    }
  }

  // Воспользуемся новым методом:
  echo Math::getDoubleSum(1, 2); // выведет 6
?>
```

Практика

Пусть у нас дан вот такой класс **ArraySumHelper**, который мы рассматривали в одном из предыдущих уроков:

```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;
    }
  }
?>
```

Задача 29.1

Переделайте методы класса **ArraySumHelper** на статические.

Задача 29.2

Пусть дан массив с числами. Найдите с помощью класса **ArraySumHelper** сумму квадратов элементов этого массива.