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→

Цепочки методов

Пусть у нас дан класс **Arr**, который хранит в себе массив чисел и может вычислять сумму этих чисел с помощью метода **getSum**. Числа могут добавляться по одному с помощью метода **add**:

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

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

    // Находим сумму чисел:
    public function getSum()
    {
      return array_sum($this->numbers);
    }
  }
?>
```

Пример использования класса **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 элементов массива:
  echo $arr->getSum(); // выведет 6
?>
```

Пусть теперь мы хотим сделать так, чтобы методы вызывались не отдельно, а цепочкой, вот так:

```php
<?php
  $arr = new Arr;
  echo $arr->add(1)->add(2)->add(3)->get
    Sum(); // так пока не работает, это наша цель
?>
```

Для того, чтобы можно было написать такую цепочку, нужно, чтобы все методы, которые участвуют в цепочке возвращали **$this**.

Как это будет работать: пусть результатом **$arr->add(1)** будет **$this**. Этот **$this** представляет собой ссылку на наш объект, то есть фактически то же самое, что хранится в переменной **$arr**.

И так будет работать каждый метод цепочки - его результатом будет тот же объект и фактически у следующего метода цепочки слева перед -> будет написан сам объект.

То есть такая цепочка:

```php
<?php
  echo $arr->add(1)->add(2)->add(3);
```

```
?>
```

Фактически является такой:

```
<?php
  $arr->add(1); $arr->add(2); $arr->add(3);
?>
```

На самом деле возвращать **$this** должны не все методы цепочки, а все методы, после которых можно написать еще один метод. В нашем случае метод **add** должен возвращать **$this**, а метод **getSum** - нет, так мы предполагаем, что этот метод всегда будет последним в цепочке и будет возвращать результат, который и выводится на экран через **echo**.

Итак, давайте поправим наш класс **Arr**:

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

    // Добавляем число в массив:
    public function add($number)
    {
      $this->numbers[] = $number;
      return $this; // вернем ссылку сами на себя
    }

    // Находим сумму чисел:
    public function getSum()
    {
      return array_sum($this->numbers);
    }
  }
?>
```

Проверим, что все работает:

```
<?php
  $arr = new Arr;
  echo $arr->add(1)->add(2)->add(3)->get
    Sum(); // выведет 6
?>
```

Можно упростить еще больше:

```
<?php
  echo (new Arr)->add(1)->add(2)->add(3)->get
    Sum(); // выведет 6
?>
```

> В цепочке, конечно же, может участвовать не один метод, как у нас сейчас, а любое количество методов в разных комбинациях, главное, чтобы все они своим результатом возвращали **$this**.

Задача 17.1

Не подсматривая в мой код самостоятельно реализуйте такой же класс **Arr**, методы которого будут вызываться в виде цепочки.

Задача 17.2

Добавьте в класс Arr еще один метод **append**, который параметром будет принимать массив чисел и добавлять эти числа в конец массива, хранящегося в объекте.

Предполагается, что методы **add** и **append** можно будет использовать в любом порядке:

```
<?php
  echo (new Arr)->add(1)->append([2, 3, 4])
    ->add(5)->getSum();
?>
```

Задача 17.3

Сделайте класс **User**, у которого будут приватные свойства **surname** (фамилия), **name** (имя) и **patronymic** (отчество).

Эти свойства должны задаваться с помощью соответствующих сеттеров.

Сделайте так, чтобы эти сеттеры вызывались цепочкой в любом порядке, а самым последним методом в цепочке можно было вызвать метод **getFullName**, который вернет ФИО юзера (первую букву фамилии, имени и отчества).

Пример:

```
<?php
  echo (new User)->setName('Николай')->set
    Patronymic('Иванович')
     ->setSurname('Петров')->getFullName(); //
      выведет 'ПНИ'
?>
```