

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

урок 12 из 107

[Верстка](#) [JavaScript](#) [PHP](#) [NodeJs](#) [Vue](#) [React](#) [Laravel](#) [WordPress](#) [AJAX](#) [Парсинг](#)[Бесплатные курсы по React для новичков. Начало 4-го ноября→](#) [Конкурс CSS картинок. Тема: Хэллоун. Призовой фонд: 100\\$. Подробности→](#)

Начальные значения свойств в конструкторе

Пусть у нас есть какой-то класс с двумя свойствами:

```
1  <?php
2  class Test
3  {
4      public $prop1;
5      public $prop2;
6  }
7  ?>
```

Давайте сделаем так, чтобы при создании объекта класса эти свойства имели какие-либо значения.

Как вы уже знаете, в момент создания объекта вызывается метод `__construct`. Давайте зададим начальные значения свойства в этом методе:

```
1  <?php
2  class Test
3  {
4      public $prop1;
5      public $prop2;
6
7      public function __construct()
8      {
9          $this->prop1 = 'value1'; // начальное зна
              чение свойства
10         $this->prop2 = 'value2'; // начальное зна
              чение свойства
11     }
12 }
13
14 $test = new Test;
15 echo $test->prop1; // выведет 'value1'
16 echo $test->prop2; // выведет 'value2'
17 ?>
```

Применение

Пусть у нас есть класс **Student** с двумя свойствами - **name** и **course** (курс студента).

Сделаем так, чтобы имя студента приходило параметром при создании объекта, а курс автоматически принимал значение **1**:

```
1  <?php
2  class Student
3  {
4      private $name;
5      private $course;
6
7      public function __construct($name)
8      {
9          $this->name = $name;
10         $this->course = 1; // курс изначально рав
              ен 1
11     }
12 }
```



```
13 | }  
13 | ?>
```

Сделаем геттеры для наших свойств:

```
1 | <?php  
2 | class Student  
3 | {  
4 |     private $name;  
5 |     private $course;  
6 |  
7 |     public function __construct($name)  
8 |     {  
9 |         $this->name = $name;  
10 |         $this->course = 1;  
11 |     }  
12 |  
13 |     // Геттер имени:  
14 |     public function getName()  
15 |     {  
16 |         return $this->name;  
17 |     }  
18 |  
19 |     // Геттер курса:  
20 |     public function getCourse()  
21 |     {  
22 |         return $this->course;  
23 |     }  
24 | }  
25 | ?>
```

Пусть имя созданного студента будет неизменяемым и доступным только для чтения, а вот для курса мы сделаем метод, который будет переводить нашего студента на следующий курс:

```
1 | <?php  
2 | class Student  
3 | {  
4 |     private $name;  
5 |     private $course;  
6 |  
7 |     public function __construct($name)  
8 |     {  
9 |         $this->name = $name;  
10 |         $this->course = 1;  
11 |     }  
12 |  
13 |     // Геттер имени:  
14 |     public function getName()  
15 |     {  
16 |         return $this->name;  
17 |     }  
18 |  
19 |     // Геттер курса:  
20 |     public function getCourse()  
21 |     {  
22 |         return $this->course;  
23 |     }  
24 |  
25 |     // Перевод студента на новый курс:  
26 |     public function transferToNextCourse()  
27 |     {  
28 |         $this->course++;  
29 |     }  
30 | }  
31 | ?>
```

Проверим работу нашего класса:

```
1 | <?php  
2 | $student = new Student('Коля'); // создаем обь
```

```

    ект класса
3 |
4 | echo $student->getCourse(); // выведет 1 -
    начальное значение
5 | $ student->transferToNextCourse(); // переведем сту
    дента на следующий курс
6 | e cho $student->getCourse(); // выведет 2
7 | ?>
```

Задача 12.1

Не подсматривая в мой код реализуйте такой же класс **Student**.

Задача 12.2

Модифицируйте метод **transferToNextCourse** так, чтобы при переводе на новый курс выполнялась проверка того, что новый курс не будет больше **5**.

