Работа с геттерами и сеттерами

Давайте рассмотрим следующий класс:

```php
<?php
  class User
  {
    public $name;
    public $age;

    // Метод для изменения возраста юзера:
    public function setAge($age)
    {
      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($age)) {
        $this->age = $age;
      }
    }

    // Метод для проверки возраста:
    private function isAgeCorrect($age)
    {
      return $age >= 18 and $age <= 60;
    }
  }
?>
```

Как вы видите, у нас есть публичные свойства **$name** и **$age**, публичный метод **setAge** для изменения возраста и приватный метод проверки возраста **isAgeCorrect**.

Очевидно, что предполагается, что возраст всегда будет меняться через метод **setAge**, так как в нем выполняется проверка возраста на корректность.

Однако, ничего не мешает сделать так:

```php
<?php
  $user = new User;

  // Вместо вызова setAge установим некорректный воз
    раст напрямую:
  $ user->age = 100500; // и у нас получится!
?>
```

Упс... Получается, что мы надеемся, что везде и всегда возраст будет меняться через **setAge**, но случайно другой программист или мы сами можем напрямую обратиться к свойству и записать в него все, что угодно.

Это ошибкоопасное место и с этим нужно что-то сделать.

Для решения проблемы можно объявить возраст приватным:

```php
<?php
  class User
  {
    public $name;
    private $age; // объявим возраст приватным

    // Метод для изменения возраста юзера:
    public function setAge($age)
    {
      // Проверим возраст на корректность:
```

```
      if ($this->isAgeCorrect($age)) {
        $this->age = $age;
      }
    }

    // Метод для проверки возраста:
    private function isAgeCorrect($age)
    {
      return $age >= 18 and $age <= 60;
    }
  }
?>
```

Теперь установить возраст напрямую (корректный или нет - не важно) не получится:

```
<?php
  $user = new User;

  // Теперь установить возраст напрямую не получится:
  $user->age = 100500; // увидим ошибку PHP!
?>
```

Отлично, мы получили то, что хотели. Но теперь есть другая проблема: мы не можем прочитать возраст снаружи, так как он приватный:

```
<?php
  $user = new User;

  // Правильным образом установим возраст:
  $user->setAge(50);

  // Попытка прочитать новый возраст при
    ведет к ошибке PHP:
  e cho $user->age; // а нам бы этого не хотелось...
?>
```

Для решения проблемы сделаем еще один метод **getAge**, с помощью которого мы будем прочитывать значения свойства **$age**:

```
<?php
  class User
  {
    public $name;
    private $age; // объявим возраст приватным

    // Метод для чтения возраста юзера:
    public function getAge()
    {
      return $this->age;
    }

    // Метод для изменения возраста юзера:
    public function setAge($age)
    {
      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($age)) {
        $this->age = $age;
      }
    }

    // Метод для проверки возраста:
    private function isAgeCorrect($age)
    {
      return $age >= 18 and $age <= 60;
    }
  }
?>
```

Теперь мы свободно можем и менять, и получать возраст:

```php
<?php
  $user = new User;

  // Установим возраст:
  $user->setAge(50);

  // Прочитаем новый возраст:
  echo $user->getAge(); // выведет 50
?>
```

Еще раз: почему мы не пишем и не читаем из свойства напрямую - потому-что нам при записи в свойства нужна какая-то проверка, поэтому мы и городим весь этот огород.

Такой подход, который мы сейчас сделали, - стандартный. В терминах этого подхода метод **getAge** называется **геттером** (*англ* getter), а метод **setAge** - **сеттером** (*англ* setter).

Этот подход удобно использовать тогда, когда нам нужна какая-то проверка в сеттере.

Очень часто бывает так, что даже если нам не нужны никакие проверки - все равно свойство объявляется приватным, а для доступа к нему используются геттеры и сеттеры.

Почему? Потому что, возможно, нам захочется добавить проверку в дальнейшем и, если все изменения свойства в коде делаются через сеттер, нам не придется вносить правки в код снаружи класса - мы просто внесем одну правку в сам сеттер.

Задача 8.1

Сделайте класс **Employee**, в котором будут следующие **private** свойства: **name** (имя), **age** (возраст) и **salary** (зарплата).

Задача 8.2

Сделайте геттеры и сеттеры для всех свойств класса **Employee**.

Задача 8.3

Дополните класс **Employee** приватным методом **isAgeCorrect**, который будет проверять возраст на корректность (от 1 до 100 лет). Этот метод должен использоваться в сеттере **setAge** перед установкой нового возраста (если возраст не корректный - он не должен меняться).

Задача 8.4

Пусть зарплата наших работников хранится в долларах. Сделайте так, чтобы геттер **getSalary** добавлял в конец числа с зарплатой значок доллара. Говоря другими словами в свойстве **salary** зарплата будет хранится просто числом, но геттер будет возвращать ее с долларом на конце.