Модификаторы доступа public и private

Давайте теперь разберем, что делает ключевое слово **public**, которое мы писали перед всеми свойствами и методами.

Ключевое слово **public** указывает на то, что данные свойства и методы доступны извне (вне кода класса). В противоположность public есть ключевое слово **private**, которое указывает на то, что свойства и методы недоступны извне.

Зачем это надо? К примеру, у нас есть класс, реализующий некоторый функционал. Есть набор методов, но часть этих методов является вспомогательными.

Будет лучше, чтобы эти вспомогательные методы нельзя было использовать **вне** нашего класса.

В этом случае мы легко сможем поредактировать код этих вспомогательных методов и будем уверенными в том, что их снаружи никто не использует и ничего страшного не случится.

Такой подход называется *инкапсуляцией* - все лишнее не должно быть доступно извне, в этом случае жизнь программиста станет проще.

То же самое касается и свойств. Некоторые свойства выполняют чисто вспомогательную функцию и не должны быть доступны вне класса, иначе мы их можем случайно поменять снаружи и сломать работу нашего кода.

Методы и свойства, которые мы не хотим сделать недоступными извне, называются *приватными* и объявляются с помощью ключевого слова **private**.

Давайте попробуем - объявим свойства **$name** и **$age** приватными и попытаемся обратиться к ним снаружи - мы сразу увидим ошибку PHP:

```php
<?php
  class User
  {
    private $name;
    private $age;
  }

  $user = new User;

  // Выдаст ошибку, так как свойство name - pri
    vate:
  $ user->name = 'Коля';
?>
```

Применим на практике

Давайте посмотрим на класс **User**, который мы сделали в предыдущем уроке:

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

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

    // Метод для изменения возраста юзера:
    public function setAge($age)
```

```
    {
      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($age)) {
        $this->age = $age;
      }
    }

    // Метод для добавления к возрасту:
    public function addAge($years)
    {
      $newAge = $this->age + $years; // вычислим нов
        ый возраст

      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($newAge)) {
        $this->age = $newAge; // обновим, если нов
          ый возраст прошел проверку
      }

    }
  }
?>
```

Как мы знаем, метод **isAgeCorrect** является вспомогательным и мы не планируем использовать его снаружи класса.

Логично поэтому сделать его приватным, чтобы другой программист, который будет потом работать над нашим проектом (или мы сами через некоторое время) случайно не использовал этот метод снаружи класса.

Сделаем это:

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

    // Объявим приватным:
    private function isAgeCorrect($age)
    {
      return $age >= 18 and $age <= 60;
    }

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

    // Метод для добавления к возрасту:
    public function addAge($years)
    {
      $newAge = $this->age + $years; // вычислим нов
        ый возраст

      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($newAge)) {
        $this->age = $newAge; // обновим, если нов
          ый возраст прошел проверку
      }

    }
  }
?>
```

Обычно все приватные методы размещают в конце класса, давайте перенесем наш метод в самый низ:

```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 добавления к возрасту:
    public function addAge($years)
    {
      $newAge = $this->age + $years; // вычислим нов
        ый возраст

      // Проверим возраст на корректность:
      if ($this->isAgeCorrect($newAge)) {
        $this->age = $newAge; // обновим, если нов
          ый возраст прошел проверку
      }
    }

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

Существует **специальное правило**: если вы делаете новые метод и не знаете, сделать его публичным или приватным, - делайте приватным. В дальнейшем, если он понадобится снаружи, - вы поменяете его на публичный.

Еще раз резюмируем: слова **public** и **private** не нужны для реализации логики программы, а нужны для того, чтобы уберечь программистов от ошибок.

Задача 6.1

Не подсматривая в мой код внесите такие же правки в класс **User**.

Задача 6.2

Попробуйте вызвать метод **isAgeCorrect** снаружи класса. Убедитесь, что это будет вызывать ошибку.

Задача 6.3

Сделайте класс **Student** со свойствами **$name** и **$course** (курс студента, от 1-го до 5-го).

Задача 6.4

В классе **Student** сделайте **public** метод **transferToNextCourse**, который будет переводить студента на следующий курс.

Задача 6.5

Выполните в методе **transferToNextCourse** проверку на то, что следующий курс не больше 5.

Задача 6.6

Вынесите проверку курса в отдельный **private** метод **isCourseCorrect**.