Контроль типов при работе с объектами

Пусть у нас дан вот такой класс **Employee**:

```php
<?php
  class Employee
  {
    private $name;
    private $salary;

    public function __construct($name, $salary)
    {
      $this->name = $name;
      $this->salary = $salary;
    }

    public function getName()
    {
      return $this->name;
    }

    public function getSalary()
    {
      return $this->salary;
    }
  }
?>
```

Также пусть дан класс **EmployeesCollection** для хранения коллекции работников:

```php
<?php
  class EmployeesCollection
  {
    private $employees = []; // массив работников

    // Добавляет работника в набор
    public function add($employee) // параметром передается объект класса Employee
    {
      $this->employees[] = $employee; // добавим объект в набор
    }

    // Получает суммарную заплату работников:
    public function getTotalSalary()
    {
      $sum = 0;

      foreach ($this->employees as $employee) {
        $sum += $employee->getSalary();
      }

      return $sum;
    }
  }
?>
```

Рассмотрим внимательно метод **add** класса **EmployeesCollection**: в нем параметром передается объект класса **Employee**.

Однако программисту, читающему наш код, сходу тяжело будет понять, что параметром метода **add** должен служить именно объект и именно класса **Employee**.

Да, мы можем оставить комментарий в нашем коде, чтобы прояснить ситуацию, но это все равно не убережет программиста от ошибок, если он попытается передать, к примеру, объект какого-нибудь другого класса или вообще массив.

Было бы круто указать тип передаваемого параметра прямо в описании функции. Ранее в учебнике мы с вами уже разбирали подобную возможность для примитивов (то есть для чисел, строк и тп, НЕ объектов).

Можно также явно задать и тип параметра, в который будет передаваться объект - мы можем точно сказать, объект какого класса там ожидается.

Для этого перед именем переменной параметра следует написать имя ожидаемого класса, в нашем случае **Employee**.

Давайте переделаем наш метод **add**:

```php
<?php
  class EmployeeCollection
  {
    private $employees = [];

    // Явно укажем тип параметра:
    public function add(Employee $employee)
    {
      $this->employees[] = $employee;
    }

    public function getTotalSalary()
    {
      $sum = 0;

      foreach ($this->employees as $employee) {
        $sum += $employee->getSalary();
      }

      return $sum;
    }
  }
?>
```

Теперь, если попытаться передать объект другого класса в метод **add** - PHP выдаст нам ошибку.

Еще раз: подобное указание типов данных нужно не самому PHP для работы, а нам, как программистом для того, чтобы мы совершали меньше ошибок при разработке.

В дальнейшем на практике мы часто будем применять указания типов и, если сейчас не очень понятно, зачем это нужно, то на практике станет гораздо понятнее.

Задача 28.1

Сделайте класс **Post** (должность), в котором будут следующие свойства, доступные только для чтения: **name** (название должности) и **salary** (зарплата на этой должности).

Задача 28.2

Создайте несколько объектов класса **Post**: программист, менеджер водитель.

Задача 28.3

Сделайте класс **Employee** (работник), в котором будут следующие свойства: **name** (имя) и **surname** (фамилия). Пусть начальные значения этих свойств будут передаваться параметром в конструктор.

Задача 28.4

Сделайте геттеры и сеттеры для свойств **name** и **surname**.

Задача 28.5

Пусть теперь третьим параметром конструктора будет передаваться должность работника, представляющая собой объект класса **Post**. Укажите тип этого параметра в явном виде.

Задача 28.6

Сделайте так, чтобы должность работника (то есть переданный объект с должностью) записывалась в свойство **post**.

Задача 28.7

Создайте объект класса **Employee** с должностью программист. При его создании используйте один из объектов класса **Post**, созданный нами ранее.

Задача 28.8

Выведите на экран имя, фамилию, должность и зарплату созданного работника.

Задача 28.9

Реализуйте в классе **Employee** метод **changePost**, который будет изменять должность работника на другую. Метод должен принимать параметром объект класса **Post**. Укажите в методе тип принимаемого параметра в явном виде.