Определение принадлежности объекта к классу

Сейчас мы с вами изучим оператор **instanceof**. Данный оператор используется для определения того, является ли текущий объект экземпляром указанного класса.

Давайте посмотрим на примере. Пусть у нас даны какие-то два класса:

```php
<?php
  // Первый класс:
  class Class1
  {

  }

  // Второй класс:
  class Class2
  {

  }
?>
```

Создадим объект первого класса:

```php
<?php
  $obj = new Class1;
?>
```

Проверим принадлежность объекта из переменной **$obj** первому классу и второму:

```php
<?php
  // Выведет true, тк объект принадлежит классу Cla
    ss1:
  v ar_dump($obj instanceof Class1);

  // Выведет false, тк объект НЕ принадлежит кла
    ccy Class2:
  v ar_dump($obj instanceof Class2);
?>
```

Задача 27.1

Сделайте класс **Employee** с публичными свойствами **name** (имя) и **salary** (зарплата).

Задача 27.2

Сделайте класс **Student** с публичными свойствами **name** (имя) и **scholarship** (стипендия).

Задача 27.3

Создайте по 3 объекта каждого класса и в произвольном порядке запишите их в массив **$arr**.

Задача 27.4

Переберите циклом массив **$arr** и выведите на экран столбец имен всех работников.

Задача 27.5

Аналогичным образом выведите на экран столбец имен всех студентов.

Задача 27.6

Переберите циклом массив **$arr** и с его помощью найдите сумму зарплат работников и сумму стипендий студентов. После цикла выведите эти два числа на экран.

Оператор instanceof и наследование

Пусть теперь у нас есть родительский класс и дочерний:

```php
<?php
  // Родительский класс:
  class ParentClass
  {

  }

  // Дочерний класс:
  class ChildClass extends ParentClass
  {

  }
?>
```

Создадим объект дочернего класса:

```php
<?php
  $obj = new ChildClass;
?>
```

Проверим теперь с помощью **instanceof**, принадлежит ли наш объект классу **ParentClass** и классу **ChildClass**:

```php
<?php
  var_dump($obj instanceof ChildClass); // выв
    едет true
  v ar_dump($obj instanceof ParentClass); //
    тоже выведет true
?>
```

Как вы видите из примера - оператор **instanceof** не делает различия при проверки между родительскими и дочерними классами.

Не путайтесь - если объект будет действительно родительского класса то, конечно же, проверка на принадлежность к дочернему классу вернет **false**:

```
<?php
  $obj = new ParentClass; // объект родительского кла
    сса

  var_dump($obj instanceof ParentClass); //
    выведет true
  v ar_dump($obj instanceof ChildClass); // выв
    едет false
?>
```

Задача 27.7

Сделайте класс **User** с публичным свойствами **name** (имя) и **surname** (фамилия).

Задача 27.8

Сделайте класс **Employee**, который будет наследовать от класса **User** и добавлять **salary** (зарплата).

Задача 27.9

Сделайте класс **City** с публичными свойствами **name** (название города) и **population** (количество населения).

Задача 27.10

Создайте 3 объекта класса **User**, 3 объекта класса **Employee**, 3 объекта класса **City**, и в произвольном порядке запишите их в массив **$arr**.

Задача 27.11

Переберите циклом массив **$arr** и выведите на экран столбец свойств **name** тех объектов, которые принадлежат классу **User** или потомку этого класса.

Задача 27.12

Переберите циклом массив **$arr** и выведите на экран столбец свойств **name** тех объектов, которые НЕ принадлежат классу **User** или потомку этого класса.

Задача 27.13

Переберите циклом массив **$arr** и выведите на экран столбец свойств **name** тех объектов, которые принадлежат именно классу **User**, то есть не классу **City** и не классу **Employee**.

Применение

Давайте рассмотрим применение оператора **instanceof** на достаточно сложном примере.

Пусть у нас есть вот такой класс для работников:

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

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

    // Геттер имени:
    public function getName()
    {
      return $this->name;
    }

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

Пусть также есть такой класс для студентов:

```php
<?php
  class Student
  {
    private $name; // имя
    private $scholarship; // стипендия

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

    // Геттер имени:
    public function getName()
    {
      return $this->name;
    }

    // Геттер стипендии:
    public function getScholarship()
    {
      return $this->scholarship;
    }
  }
?>
```

Как вы видите, и работник, и студент имеют имя и какой-то доход: у работника это зарплата, а у студента - стипендия.

Пусть теперь мы хотим сделать класс **UsersCollection**, предназначенный для хранения работников и студентов.

Работников мы будем хранить в свойстве **employees**, а студентов - в свойстве **students**:

```php
<?php
  class UsersCollection
  {
    private $employees = []; // массив работников
    private $students = []; // массив студентов
  }
?>
```

Давайте теперь реализуем единый метод **add** для добавления и работников, и студентов.

Этот метод параметром будет принимать объект и, если это работник - добавлять его в массив работников, а если студент - в массив студентов.

Пример того, как мы будем пользоваться методом **add** после его реализации:

```php
<?php
  $usersCollection = new UsersCollection;

  $usersCollection->add(new Employee('Коля', 2
    00)); // попадет к работникам
  $ usersCollection->add(new Student('Вася', 100
    )); // попадет к студентам
?>
```

Итак, давайте реализуем описанный метод **add**. Здесь нам и поможет изученный нами оператор **instanceof**:

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

    // Добавление в массивы:
    public function add($user)
    {
      // Если передан объект класса Employee:
      if ($user instanceof Employee) {
        $this->employees[] = $user; // добавляем к р
          аботникам
      }

      // Если передан объект класса Student:
      if ($user instanceof Student) {
        $this->students[] = $user; // добавляем к с
          тудентам
      }

    }
  }
?>
```

Давайте также реализуем методы для нахождения суммарной зарплаты и суммарной стипендии:

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

    // Добавление в массивы:
    public function add($user)
    {
      if ($user instanceof Employee) {
```

```
      $this->employees[] = $user;
    }

    if ($user instanceof Student) {
      $this->students[] = $user;
    }
  }

  // Получаем суммарную зарплату:
  public function getTotalSalary()
  {
    $sum = 0;

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

    return $sum;
  }

  // Получаем суммарную стипендию:
  public function getTotalScholarship()
  {
    $sum = 0;

    foreach ($this->students as $student) {
      $sum += $student->getScholarship();
    }

    return $sum;
  }
}
?>
```

Реализуем также метод, который будет находить общую сумму платежей и работникам, и студентам:

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

    // Добавление в массивы:
    public function add($user)
    {
      if ($user instanceof Employee) {
        $this->employees[] = $user;
      }

      if ($user instanceof Student) {
        $this->students[] = $user;
      }
    }

    // Получаем суммарную зарплату:
    public function getTotalSalary()
    {
      $sum = 0;

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

      return $sum;
    }

    // Получаем суммарную стипендию:
    public function getTotalScholarship()
    {
      $sum = 0;

```

```
        foreach ($this->students as $student) {
          $sum += $student->getScholarship();
        }

        return $sum;
      }

      // Получаем общую сумму платежей и работникам, и с
        тудентам:
       public function getTotalPayment()
      {
        return $this->getTotalScholarship() + $this->getTotalSalary();
      }
    }
?>
```

Проверим работу нашего класса:

```
<?php
  $usersCollection = new UsersCollection;

  $usersCollection->add(new Student('Петя', 100
    ));
  $ usersCollection->add(new Student('Ваня', 200
    ));

  $usersCollection->add(new Employee('Коля', 3
    00));
  $ usersCollection->add(new Employee('Вася', 4
    00));

  // Получим полную сумму стипендий:
  echo $usersCollection->getTotalScholarship();
    // выведет 300

  // Получим полную сумму зарплат:
  echo $usersCollection->getTotalSalary();
    // выведет 700

  // Получим полную сумму платежей:
  echo $usersCollection->getTotalPayment();
    // выведет 1000
?>
```

Задача 27.14

Скопируйте мой код классов **Employee** и **Student** и самостоятельно не подсматривая в мой код реализуйте такой же класс **UsersCollection**.