Сравнение объектов

Сейчас мы с вами научимся сравнивать объекты с помощью операторов == и ===.

Вы должны уже знать, что для примитивов (то есть не объектов) оператор == сравнивает данные по значению без учета типа, а оператор === - учитывая тип:

```php
<?php
  var_dump(3 == 3);    // выведет true
  var_dump(3 == '3');  // выведет true

  var_dump(3 === 3);   // выведет true
  var_dump(3 === '3'); // выведет false
?>
```

Давайте теперь посмотрим, как работает сравнение объектов.

При использовании оператора == для сравнения двух объектов выполняется сравнение свойств объектов: два объекта равны, если они имеют одинаковые свойства и их значения (значения свойств сравниваются через ==) и являются экземплярами одного и того же класса.

При сравнении через ===, переменные, содержащие объекты, считаются равными только тогда, когда они ссылаются на один и тот же экземпляр одного и того же класса.

Давайте посмотрим на примере. Пусть у нас дан вот такой класс **User**:

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

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

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

    public function getAge()
    {
      return $this->age;
    }
  }
?>
```

Создадим два объекта нашего класса с одинаковыми значениями свойств и сравним созданные объекты:

```php
<?php
  $user1 = new User('Коля', 30);
  $user2 = new User('Коля', 30);

  var_dump($user1 == $user2); // выведет true
?>
```

Пусть теперь значения свойств одинаковые, но у них разный тип:

```php
<?php
  $user1 = new User('Коля', 30); // возраст - ч
    исло
  $ user2 = new User('Коля', '30'); // возраст - с
    трока

  var_dump($user1 == $user2); // выведет true
?>
```

Пусть значения свойств разные:

```php
<?php
  $user1 = new User('Коля', 30);
  $user2 = new User('Коля', 25);

  var_dump($user1 == $user2); // выведет false
?>
```

Давайте теперь сравним два наших объекта через ===:

```php
<?php
  $user1 = new User('Коля', 30);
  $user2 = new User('Коля', 30);

  var_dump($user1 === $user2); // выведет false
?>
```

Чтобы две переменных с объектами действительно были равными при сравнении через ===, они должны указывать на один и тот же объект.

Давайте сделаем, чтобы это было так, и сравним переменные:

```php
<?php
  $user1 = new User('Коля', 30);
  $user2 = $user1; // передача объекта по ссылке

  var_dump($user1 === $user1); // выведет true
?>
```

Задача 26.1

Сделайте функцию **compare**, которая параметром будет принимать два объекта и возвращать **true**, если они имеют одинаковые свойства и значения являются экземплярами одного и того же класса, и **false**, если это не так.

Задача 26.2

Сделайте функцию **compare**, которая параметром будет принимать два объекта и возвращать **true**, если переданные переменные ссылаются на один и тот же объект, и **false**, если на разные.

Задача 26.3

Сделайте функцию **compare**, которая параметром будет принимать два объекта и сравнивать их.

Функция должна возвращать **1**, если переданные переменные ссылаются на один и тот же объект.

Функция должна возвращать **0**, если объекты разные, но одного и того же класса и с теми же свойствами и их значениями.

Функция должна возвращать **-1** в противном случае.

Применение

Давайте рассмотрим применение изученной теории.

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;
    }
  }
?>
```

Пусть также дан такой класс **EmployeesCollection** для хранения коллекции работников:

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

    // Добавляем нового работника:
    public function add($newEmployee)
    {
      $this->employees[] = $newEmployee;
    }

    // Получаем всех работников в виде массива:
    public function get()
    {
      return $this->employees;
    }
  }
?>
```

Давайте сделаем так, чтобы работник, который уже есть в нашем наборе, не добавлялся еще раз.

Для этого сделаем вспомогательный метод **exists**, который будет принимать объект с новым работником и проверять его наличие в массиве **$this->employees**:

```php
<?php
  private function exists($newEmployee)
  {
    foreach ($this->employees as $employee) {
      if ($employee == $newEmployee) { // сравниваем чер
        ез ==
```

```
        return true;
      }
    }

    return false;
  }
?>
```

Давайте применим новый метод в нашем классе:

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

    public function add($newEmployee)
    {
      // Если такого работника нет - то добавляем:
      if (!$this->exists($newEmployee)) {
        $this->employees[] = $newEmployee;
      }
    }

    public function get()
    {
      return $this->employees;
    }

    private function exists($newEmployee)
    {
      foreach ($this->employees as $employee) {
        if ($employee == $newEmployee) {
          return true;
        }
      }

      return false;
    }
  }
?>
```

Давайте проверим работу нашего класса:

```
<?php
  $employeesCollection = new EmployeesCollection;

  $employeesCollection->add(new Employee('Ko
    ля', 100));
  $ employeesCollection->add(new Employee('Ko
    ля', 100)); // второго такого же не добавит

  var_dump($employeesCollection->get()); // убедимся в э
    том
?>
```

В общем-то, мы получили устраивающий нас код. Но давайте попробуем поиграться с ним: поменяем при сравнении двойное равно на тройное:

```
<?php
  private function exists($newEmployee)
  {
    foreach ($this->products as $product) {
      if ($product === $newEmployee) { //!! сравниваем чер
        ез ===
        return true;
      }
    }
```

```
9
10      return false;
11    }
12  ?>
```

Теперь при попытке добавить нового работника с такими же значениями свойств он будет добавляться:

```
1  <?php
2    $employeesCollection = new EmployeesCollection;
3
4    $employeesCollection->add(new Employee('Ko
       ля', 100));
5    $ employeesCollection->add(new Employee('Ko
       ля', 100)); // добавит
6
7    var_dump($employeesCollection->get());
8  ?>
```

Но если попытаться добавить тот же объект, то добавления не произойдет:

```
1  <?php
2    $employeesCollection = new EmployeesCollection;
3
4    $employee = new Employee('Коля', 100);
5
6    $employeesCollection->add($employee);
7    $employeesCollection->add($employee); //
       не добавит, тк тот же объект
8
9    var_dump($employeesCollection->get());
10 ?>
```

Задача 26.4

Скопируйте мой код класса **Employee**, затем самостоятельно реализуйте описанный класс **EmployeesCollection**, проверьте его работу.

Функция in_array

На самом деле код функции **exists** можно заменить стандартной PHP функцией **in_array**.

Нужно только знать, как она выполняет сравнение - по двойному равно или по тройному.

Из документации следует, что функция **in_array** имеет третий необязательный параметр. Если он не установлен или равен **false** - функция сравнивает по двойному равно, а равен **true** - то по тройному.

Давайте упростим код класса при условии сравнения объектов через двойное равно:

```
1  <?php
2    class EmployeesCollection
3    {
4      private $employees = [];
5
6      public function add($newEmployee)
7      {
8        if (!in_array($newEmployee, $this->employees, fal
           se)) {
9          $this->employees[] = $newEmployee;
10       }
11     }
12
13     public function get()
14     {
15       return $this->employees;
```

```
    }
  }
?>
```

А теперь при условии сравнения на тройное равно:

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

    public function add($newEmployee)
    {
      // Эквивалентно методу exists с ===
      if (!in_array($newEmployee, $this->employees, tru
        e)) {
        $this->employees[] = $newEmployee;
      }
    }

    public function get()
    {
      return $this->employees;
    }
  }
?>
```

Задача 26.5

Упростите ваш класс **EmployeesCollection** с использованием функции **in_array**, проверьте его работу.