Работа с трейтами

Как вы уже знаете, в PHP нельзя наследовать от нескольких классов сразу, только от одного.

Ранее мы уже проходили решение этой проблемы: вместо наследования использовать объекты одних классов внутри других.

В PHP есть и другой способ. Он заключается в использовании *трейтов*.

Трейт представляет собой набор свойств и методов, которые можно включить в другой класс. При этом свойства и методы трейта будут восприниматься классом будто свои.

Синтаксис трейта такой же как и у класса, за исключением того, что имя трейта нужно объявлять с помощью ключевого слова **trait**.

Экземпляр трейта нельзя создать - трейты предназначены только для подключения к другим классам.

Само подключение осуществляется с помощью команды **use**, после которой через пробел указывается имя подключаемого трейта. Данная команда пишется в начале класса.

Давайте посмотрим применение трейтов на практическом примере.

Пусть у нас дан вот такой трейт **Helper**, содержащий приватные свойства **name** и **age**, а также их геттеры:

```php
<?php
  trait Helper
  {
    private $name;
    private $age;

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

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

Пусть у нас также есть вот такой класс **User**, в конструкторе которого задаются свойства **name** и **age**:

```php
<?php
  class User
  {
    public function __construct($name, $age)
    {
      $this->name = $name;
      $this->age = $age;
    }
  }
?>
```

Давайте теперь добавим геттеры для свойств нашего класса **User**. Только не будем их записывать в самом классе, а просто подключим трейт **Helper**, в котором эти методы уже реализованы:

```php
<?php
  class User
  {
    use Helper; // подключаем трейт
```

```
5
6       public function __construct($name, $age)
7       {
8         $this->name = $name;
9         $this->age = $age;
10      }
11    }
12  ?>
```

После подключения трейта в нашем классе появятся методы этого трейта. При этом обращаться мы к ним будем будто к методам самого класса:

```
1  <?php
2    $user = new User('Коля', 30);
3    echo $user->getName(); // выведет 'Коля'
4    echo $user->getAge(); // выведет 30
5  ?>
```

Свойства трейта также будут доступны в нашем классе.

Для того, чтобы продемонстрировать преимущества трейтов, давайте сделаем еще один класс **City** (город).

У города также будет имя и возраст, однако, логично, что город и юзер не могут наследовать от одного родителя, так представляют собой немного разные сущности, пусть и имеющие похожие методы.

Поэтому воспользуемся созданным нами трейтом **Helper** и в классе **City**:

```
1  <?php
2    class City
3    {
4      use Helper;
5
6      public function __construct($name, $age)
7      {
8        $this->name = $name;
9        $this->age = $age;
10     }
11   }
12 ?>
```

Проверим работу нашего класса:

```
1  <?php
2    $city = new City('Минск', 1000);
3    echo $city->getName(); // выведет 'Мин
       ск'
4    e cho $city->getAge(); // выведет 1000
5  ?>
```

Задача 46.1

Реализуйте класс **Country** (страна) со свойствами **name** (название), **age** (возраст), **population** (количество населения) и геттерами для них. Пусть наш класс для сокращения своего кода использует уже созданный нами трейт **Helper**.

Несколько трейтов

В классе можно использовать не один, а несколько трейтов. В этом и проявляется их преимущество перед наследованием. Нужные для использования в классе трейты можно указать через запятую после ключевого слова **use**.

Задача 46.2

Сделайте 3 трейта с названиями **Trait1**, **Trait2** и **Trait3**. Пусть в первом трейте будет метод **method1**, возвращающий 1, во втором трейте - метод **method2**, возвращающий 2, а в третьем трейте - метод **method3**, возвращающий 3. Пусть все эти методы будут приватными.

Задача 46.3

Сделайте класс **Test**, использующий все три созданных нами трейта. Сделайте в этом классе публичный метод **getSum**, возвращающий сумму результатов методов подключенных трейтов.