Разрешение конфликтов в трейтах

Так как один класс может использовать несколько трейтов, то нас может поджидать проблема, возникающая тогда, когда два трейта имеют одноименные методы.

В этом случае PHP выдаст фатальную ошибку. Чтобы поправить ситуацию, нужно будет разрешить конфликт имен явным образом. Как это делается - посмотрим на практике.

Пусть у нас есть два трейта с одинаковым методом **method**:

```php
<?php
  trait Trait1
  {
    private function method()
    {
      return 1;
    }
  }

  trait Trait2
  {
    private function method()
    {
      return 2;
    }
  }
?>
```

Пусть у нас также есть класс **Test**, использующий оба наших трейта. Если просто подключить оба трейта к нашему классу, то PHP выдаст ошибку, так как у трейтов есть совпадающий методы:

```php
<?php
  // Данный код выдаст ошибку!
  class Test
  {
    use Trait1, Trait2; // подключаем трейты
  }
?>
```

Давайте разрешим (в данном контексте это слово значит *разрулим*) конфликт имен наших трейтов. Для этого существует специальный оператор **insteadof** (переводится *вместо чего-то*).

Давайте с помощью этого оператора скажем следующее: *использовать метод **method** трейта **Trait1** вместо такого же метода трейта **Trait2***:

```php
<?php
  class Test
  {
    use Trait1, Trait2 {
      Trait1::method insteadof Trait2;
    }
  }

  new Test;
?>
```

Как вы видите, синтаксис тут следующий: вначале имя трейта, потом два двоеточия, потом имя метода, потом наш оператор **insteadof** и имя второго трейта.

Давайте проверим:

```php
<?php
  class Test
  {
    use Trait1, Trait2 {
      Trait1::method insteadof Trait2;
    }

    public function __construct()
    {
      echo $this->method(); // выведет 1, тк
        это метод первого трейта
     }

  }

  new Test;
?>
```

Итак, в нашем классе мы сказали, что если используется метод **method**, то следует брать его из первого трейта.

Можно и наоборот - взять метод второго трейта:

```php
<?php
  class Test
  {
    use Trait1, Trait2 {
      Trait2::method insteadof Trait1;
    }

    public function __construct()
    {
      echo $this->method(); // выведет 2, тк
        это метод второго трейта
     }

  }

  new Test;
?>
```

В любом случае, если мы указываем использовать метод одного трейта, то метод второго трейта оказывается недоступным.

Можно использовать и метод второго трейта, переименовав его через ключевое слово **as**, вот так:

```php
<?php
  class Test
  {
    use Trait1, Trait2 {
      Trait1::method insteadof Trait2; // берем мет
        од из первого трейта
      Trait2::method as method2; // метод второго тре
        йта будет доступен как method2
     }

    public function __construct()
    {
      echo $this->method() + $this->method2();
        // выведет 3
     }

  }

  new Test;
?>
```

При желании можно переименовать и метод первого трейта:

```php
<?php
  class Test
  {
    use Trait1, Trait2 {
      Trait1::method insteadof Trait2;
      Trait1::method as method1; // метод первого тре
        йта будет доступен как method1
      Trait2::method as method2; // метод второго тре
        йта будет доступен как method2
     }

    public function __construct()
    {
      echo $this->method1() + $this->method2();
        // выведет 3
     }

  }

  new Test;
?>
```

Использовать ключевое слово **as** без определения главного метода через **insteadof** нельзя, это выдаст ошибку:

```php
<?php
  // Данный класс выдаст ошибку:
  class Test
  {
    use Trait1, Trait2 {
      Trait1::method as method1;
      Trait2::method as method2;
    }

    public function __construct()
    {
      echo $this->method1() + $this->method2();
    }
  }

  new Test;
?>
```

Задача 47.1

Сделайте 3 трейта с названиями **Trait1**, **Trait2** и **Trait3**. Пусть в первом трейте будет метод **method**, возвращающий 1, во втором трейте - одноименный метод, возвращающий 2, а в третьем трейте - одноименный метод, возвращающий 3.

Задача 47.2

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