Перезапись конструктора родителя в потомке

Пусть у нас есть вот такой класс **User**, у которого свойства **name** и **age** задаются в конструкторе и в дальнейшем доступны только для чтения (то есть приватные и имеют только геттеры, но не сеттеры):

```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;
    }
  }
?>
```

От этого класса наследует класс **Student**:

```php
<?php
  class Student extends User
  {
    private $course;

    public function getCourse()
    {
      return $this->course;
    }
  }
?>
```

Класс-потомок не имеет своего конструктора - это значит что при создании объекта класса сработает конструктор родителя:

```php
<?php
  $student = new Student('Коля', 19); // сра
    ботает конструктор родителя

  echo $student->getName(); // выведет 'Коля'
  echo $student->getAge(); // выведет 19
?>
```

Все замечательно, но есть проблема: мы бы хотели при создании объекта класса **Student** третьим параметром передавать еще и курс, вот так:

```php
<?php
  $student = new Student('Коля', 19, 2); //
    это пока не работает
?>
```

Самое простое, что можно сделать, это переопределить конструктор родителя своим конструктором, забрав из родителя его код:

```php
<?php
  class Student extends User
  {
    private $course;

    // Конструктор объекта:
    public function __construct($name, $age, $co
      urse)
     {
      // Дублируем код конструктора родителя:
      $this->name = $name;
      $this->age = $age;

      // Наш код:
      $this->course = $course;
    }

    public function getCourse()
    {
      return $this->course;
    }
  }
?>
```

При этом мы в классе потомке обращаемся к приватным свойствам родителя **name** и **age**, что, конечно же, не будет работать так, как нам нужно.

Переделаем их на **protected**:

```php
<?php
  class User
  {
    protected $name; // объявим свойство защ
      ищенным
     protected $age;  // объявим свойство защ
      ищенным

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

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

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

Теперь при создании студента третьим параметром мы можем передать и курс:

```php
<?php
  $student = new Student('Коля', 19, 2); //
    теперь это работает

  echo $student->getName(); // выведет 'Коля'
  echo $student->getAge(); // выведет 19
  echo $student->getCourse(); // выведет 2
?>
```

Задача 22.1

Не подсматривая в мой код реализуйте такой же класс **Student**, наследующий от **User**.

Используем конструктор родителя

Понятно, что дублирование кода родителя в классе потомке - это не очень хорошо.

Давайте вместо дублирования кода в конструкторе потомка вызовем конструктор родителя.

Для полной ясности распишу все по шагам.

Вот так выглядит конструктор класса **User**, он принимает два параметра **$name** и **$age** и записывает их в соответствующие свойства:

```php
<?php
  // Конструктор объекта класса User:
  public function __construct($name, $age)
  {
    $this->name = $name;
    $this->age = $age;
  }
?>
```

Вот конструктор класса **Student**, который мы хотим переписать:

```php
<?php
  // Конструктор объекта класса Student:
  public function __construct($name, $age, $co
    urse)
  {
    // Этот код хотим заменить на вызов конструктора род
      ителя:
     $this->name = $name;
    $this->age = $age;

    // Наш код:
    $this->course = $course;
  }
?>
```

Как вызвать конструктор родителя внутри потомка? Вы это уже знаете - с помощью **parent**. То есть вот так: *parent::__construct*.

При этом конструктор родителя первым параметром ожидает имя, а вторым - возраст, и мы должны ему их передать, вот так: *parent::__construct(**$name**, **$age**)*.

Давайте сделаем это:

```php
<?php
  // Конструктор объекта класса Student:
  public function __construct($name, $age, $co
    urse)
  {
    // Вызовем конструктор родителя, передав ему
```

```
        два параметра:
      parent::__construct($name, $age);

      // Запишем свойство course:
      $this->course = $course;
    }
?>
```

Напишем полный код класса **Student**:

```
<?php
  class Student extends User
  {
    private $course;

    // Конструктор объекта:
    public function __construct($name, $age, $co
      urse)
     {
      parent::__construct($name, $age); //
        вызываем конструктор родителя
      $this->course = $course;
    }

    public function getCourse()
    {
      return $this->course;
    }
  }
?>
```

Проверим, что все работает:

```
<?php
  $student = new Student('Коля', 19, 2);

  echo $student->getName(); // выведет 'Коля'
  echo $student->getAge(); // выведет 19
  echo $student->getCourse(); // выведет 2
?>
```

Так как класс **Student** теперь не обращается напрямую к свойствам **name** и **age** родителя, можно их опять сделать приватными:

```
<?php
  class User
  {
    private $name; // объявим свойство приватным
    private $age;  // объявим свойство приватным

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

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

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

Задача 22.2

Сделайте класс **User**, в котором будут следующие свойства только для чтения: **name** (имя), **surname** (фамилия), Начальные значения этих свойств должны устанавливаться в конструкторе. Сделайте также геттеры этих свойств.

Задача 22.3

Сделайте так, чтобы третьим параметром в конструктор передавалась дата рождения работника в формате *год-месяц-день* Запишите ее в свойство **birthday**. Сделайте геттер для этого свойства.

Задача 22.4

Сделайте приватный метод **calculateAge**, который параметром будет принимать дату рождения, а возвращать возраст с учетом того, был ли уже день рождения в этом году, или нет.

Задача 22.5

Сделайте так, чтобы метод **calculateAge** вызывался в конструкторе объекта, рассчитывал возраст пользователя и записывал его в приватное свойство **age**. Сделайте геттер для этого свойства.

Задача 22.6

Сделайте класс **Employee**, который будет наследовать от класса **User**. Пусть новый класс имеет свойство **salary**, в котором будет хранится зарплата работника. Зарплата должна передаваться четвертым параметром в конструктор объекта. Сделайте также геттер для этого свойства.