Модификатор доступа protected

Итак, как вы уже знаете, приватные свойства и методы не наследуются.

Если вы хотите, чтобы метод или свойство появились у потомка, вы должны объявить их как **public**.

Проблема, однако, в том, что публичные методы будут также доступны и извне класса, а мы бы этого не хотели.

Другими словами, мы хотели бы, чтобы некоторые методы или свойства родителя наследовались потомками, но при этом для всего остального мира вели себя как приватные.

Для решения этой проблемы существует специальный модификатор **protected**, который и реализует описанное.

Давайте изучим его работу на реальном примере.

Пусть у нас дан вот такой класс **User** с приватными свойствами **name** и **age**:

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

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

    public function setName($name)
    {
      $this->name = $name;
    }

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

    public function setAge($age)
    {
      $this->age = $age;
    }
  }
?>
```

Пусть от класса **User** наследует класс **Student**:

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

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

    public function setCourse($course)
    {
      $this->course = $course;
    }

```

```
    }
?>
```

Пока все отлично и все работает.

Пусть теперь мы решили в классе **Student** сделать метод **addOneYear**, который будет добавлять 1 год к свойству **age**. Давайте реализуем указанный метод:

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

    // Реализуем этот метод:
    public function addOneYear()
    {
      $this->age++;
    }

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

    public function setCourse($course)
    {
      $this->course = $course;
    }
  }
?>
```

Проблема в том, что если оставить свойство **age** приватным, то мы не сможем обратится к нему в классе-потомке - это выдаст ошибку при попытке вызова метода **addOneYear**:

```php
<?php
  $student = new Student();

  $student->setAge(25);
  $student->addOneYear(); //!! выдаст ошибку
?>
```

Для исправления ошибки поправим класс **User** - объявим свойство **age** как **protected**, а не как **private**:

```php
<?php
  class User
  {
    private $name;
    protected $age; // объявим свойство как protected
    ...
  }
?>
```

Теперь метод **addOneYear** потомка сможет менять свойство **age**, но оно по-прежнему не будет доступно извне наших классов.

Давайте соберем все вместе и запустим:

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

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

    public function setName($name)
    {
```

```
    $this->name = $name;
  }

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

  public function setAge($age)
  {
    $this->age = $age;
  }
}

class Student extends User
{
  private $course;

  // Метод добавления 1 года к возрасту:
  public function addOneYear()
  {
    $this->age++;
  }

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

  public function setCourse($course)
  {
    $this->course = $course;
  }
}
?>
```

Проверим работу класса **Student**:

```
<?php
  $student = new Student();

  $student->setName('Коля'); // установим имя
  $student->setCourse(3); // установим курс
  $student->setAge(25); // установим возраст в 25

  $student->addOneYear(); // увеличим возраст на
    единицу
  e cho $student->getAge(); //!! выведет 26
?>
```

Попытка обратится к свойству **age** снаружи класса выдаст ошибку, как нам и нужно:

```
<?php
  $student = new Student();
  $student->age = 30; // выдаст ошибку
?>
```

Задача 20.1

Скопируйте мой код класса **User**. Самостоятельно не подсматривая в мой код реализуйте описанный класс **Student** с методами **getCourse**, **setCourse** и **addOneYear**.