Классы и объекты

Как создать класс в PHP?

Класс – это набор данных и методов, имеющих общую, целостную, хорошо определенную сферу ответственности.

```
class имя_класса {
    описание_класса
}
```

Данные (свойства) – необязательный компонент класса: класс может включать только методы, предоставляющие целостный набор услуг.

Главное условие эффективного программирования

«максимизация части программы, которую можно игнорировать при работе над конкретными фрагментами кода».

Главное средство достижения этой цели – *классы*.

Как создать объект в PHP?

Объекты – создаются на основе заранее определенных классов при помощи оператора *new*.

```php
class ShopProduct {
    private $_price = 0;

    public function getPrice() {
        return $this->_price;
    }
}
$product = new ShopProduct();
```

Псевдо-переменная *$this* доступна в контексте объекта и является ссылкой на вызываемый объект.

В контексте класса можно создать новый объект через new self() и new parent().

Объекты всегда передаются в функцию или присваиваются переменной по ссылке. Копию объекта можно создать с помощью оператора *clone*.

Как объявить свойство класса?

Переменные, объявленные внутри класса, называются *свойствами* класса. Часто используют и другие термины: аттрибуты, поля, данные, свойства-члены.

```
class ShopProduct {
  public  $title;       # наименование товара
  private $_price = 0; # цена товара со значением по-умолчанию 0

  public function getPrice() {
    return $this->_price; # получаем свойство цена внутри класса
  }
}

$product = new ShopProduct();

$product->title = 'Баян'; # устанавливаем значение свойства
echo $product->title;     # получаем значение свойства
```

К свойствам объекта можно обращаться с помощью оператора '->' указав имя объектной переменной и имя свойства.

Как объявить метод класса?

Методы – это специальные функции, которые объявляются внутри класса.

Методы позволяют объектам выполнять задачи.

```
class ShopProduct {
    private $_price = 0;

    public function getPrice() {
        return $this->_price;
    }

    public function setPrice($price) {
        $this->_price = $price;
    }
}

$product = new ShopProduct();

$product->setPrice(1000);
echo $product->getPrice();
```

Метод-конструктор

```
void __construct ( [ mixed args [ , ... ]] )
```

Вызывается автоматически при создании экземпляра объекта.

Конструктор выполняет инициализацию объекта.

```php
class ShopProduct {
    public $title;
    private $_price = 0;

    public function __construct($title, $price=0) {
        $this->title = $title;
        $this->_price = $price;
    }
}

$product = new ShopProduct('Расческа для усов', 1000);
$product = new ShopProduct('Уточка для ванной', 3500);
```

Конструктор относится к *магическим методам* PHP, которые будут рассмотрены в следующем уроке.

Как осуществлять контроль типа?

В PHP 5 добавили возможность уточнять тип данных классов.

Эта возможность доступна для аргументов функций и методов.

```
class ProductCollection {
    public function addProduct(ShopProduct $product) {
        array_push($this->_products, $product);
    }
}

$collection = new ShopCollection();

$product = new ShopProduct();
$collection->addProduct($product);

$collection->addProduct(new ShopProduct());
```

Уточнение можно использовать для классов, интерфейсов, массивов (array) и функций обратного вызова (callable).

Но нельзя использовать для принудительного определения аргументов элементарных типов и трейтов.

Задача №1

Задачка-разминка.

Создать класс ShopProduct – товар в магазине.

Какие свойства, по вашему мнению, имеет товар? Добавить их в класс.

Наследование

Наследование – это механизм, позволяющий описать новый класс на основе уже существующего (родительского, базового) класса.

Полученный в результате наследования класс называют *дочерним* или *потомком*.

```
class BookProduct extends ShopProduct {
    public $pages = 0;
}

class CDProduct extends ShopProduct {
    public $duration = 0;
}
```

Дочерний класс наследует все свойства и методы своего *родителя*.

Дочерний класс расширяет функциональность родительского класса.

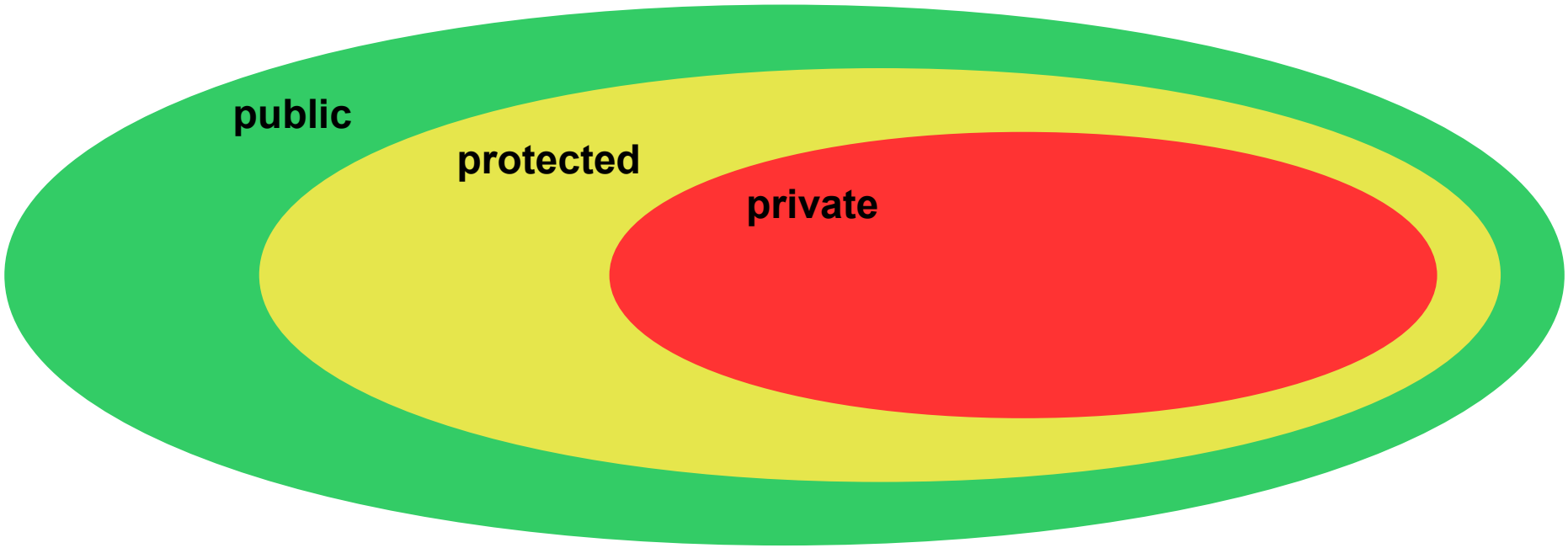
PHP не поддерживает множественное наследование.

Модификаторы доступа

PUBLIC – Открытый тип доступа.
Разрешает доступ из любого контекста.

PROTECTED – Защищенный тип доступа.
Разрешает доступ только внутри класса и его потомков, но запрещает доступ извне.

PRIVATE – Закрытый тип доступа.
Разрешает доступ только в пределах данного класса.



Задача №2

В классе ShopProduct описать три свойства с разными модификаторами доступа: public, protected и private.

Создать новый объект и вывести значения этих свойств на экран (клиентский код)а.

В класс ShopProduct добавить метод, который будет выводить значения этих трех свойств на экран.

Создать дочерний от ShopProduct класс и добавить метод, который будет выводить значения этих трех свойств на экран.

Оператор разрешения области видимости

"Двойное двоеточие" – это лексема, позволяющая обращаться к статическим свойствам, константам и перегруженным свойствам или методам класса.

parent:: – ссылка на переопределенный родительский метод.

self:: – ссылка на текущий класс.

static:: – последнее статическое связывание. Ссылка на вызываемый класс в контексте статического наследования.

Как объявить статическое свойство или метод?

Как объявить константу класса?

Как объявить статическое свойство или метод?

Доступ к свойствам и методам класса можно получить без создания экземпляра объекта.

Такие свойства и методы являются *статическими* и должны быть объявлены с помощью ключевого слова *static*.

```
class Singleton {
    static private $_instance;

    private function __construct() {}

    static public function getInstance() {
        if (!self::$_instance) {
            self::$_instance = new self();
        }
        return self::$_instance;
    }
}

$obj = Singleton::getInstance();
```

Как объявить константу класса?

Аналогичны обычным константам, только объявляются внутри класса.

```
class MyClass {
    const CONSTANT = 'Значение константы';

    public function showConstant() {
        echo self::CONSTANT;
    }
}

echo MyClass::CONSTANT;
```

Конец