Профессиональная разработка вэб-приложений

Герасименко Сергей Валерьевич

Функция «include_once»

Функция **include_once(“путь к файлу”)** подключает и выполняет указанный файл во время выполнения скрипта. Данная функция проверяет был ли ранее подключен файл и если да, то функция не выполняется.

```php
//файл inc.php который подключаем
$var=1;
```

```php
//файл test.php который вызывает include
for($i=0;$i<2;$i++){
include_once("inc.php");
echo "Переменная - ".$var."<BR>";
$var++;//Увеличиваем переменную на 1
}

//Выведет Переменная - 1 Переменная - 2
```

Функция «include»

Функция **include (“путь к файлу”)** подключает и выполняет указанный файл во время выполнения скрипта без осуществления проверки на возможное его подключение ранее. То есть если в файле были инициализированы переменные, то они переопределяются:

```php
//файл inc.php который подключаем
$var=1;
```

PHP - Код

```php
//файл test.php который вызывает include
for($i=0;$i<2;$i++){
include ("inc.php");
echo "Переменная - ".$var."<BR>";
$var++;//Увеличиваем переменную на 1
}

//Выведет Переменная - 1 Переменная - 1
```

Функция «require» и «require_once»

Функция **require()** - аналогичная include(), но если вызываемого файла нет (например файла inc.php или мы укажем неверный путь) то require() остановит выполнение скрипта, а при include() выполнение продолжится. **require_once()** - функция аналогичная include_once(), но с замечаниями как и для require().

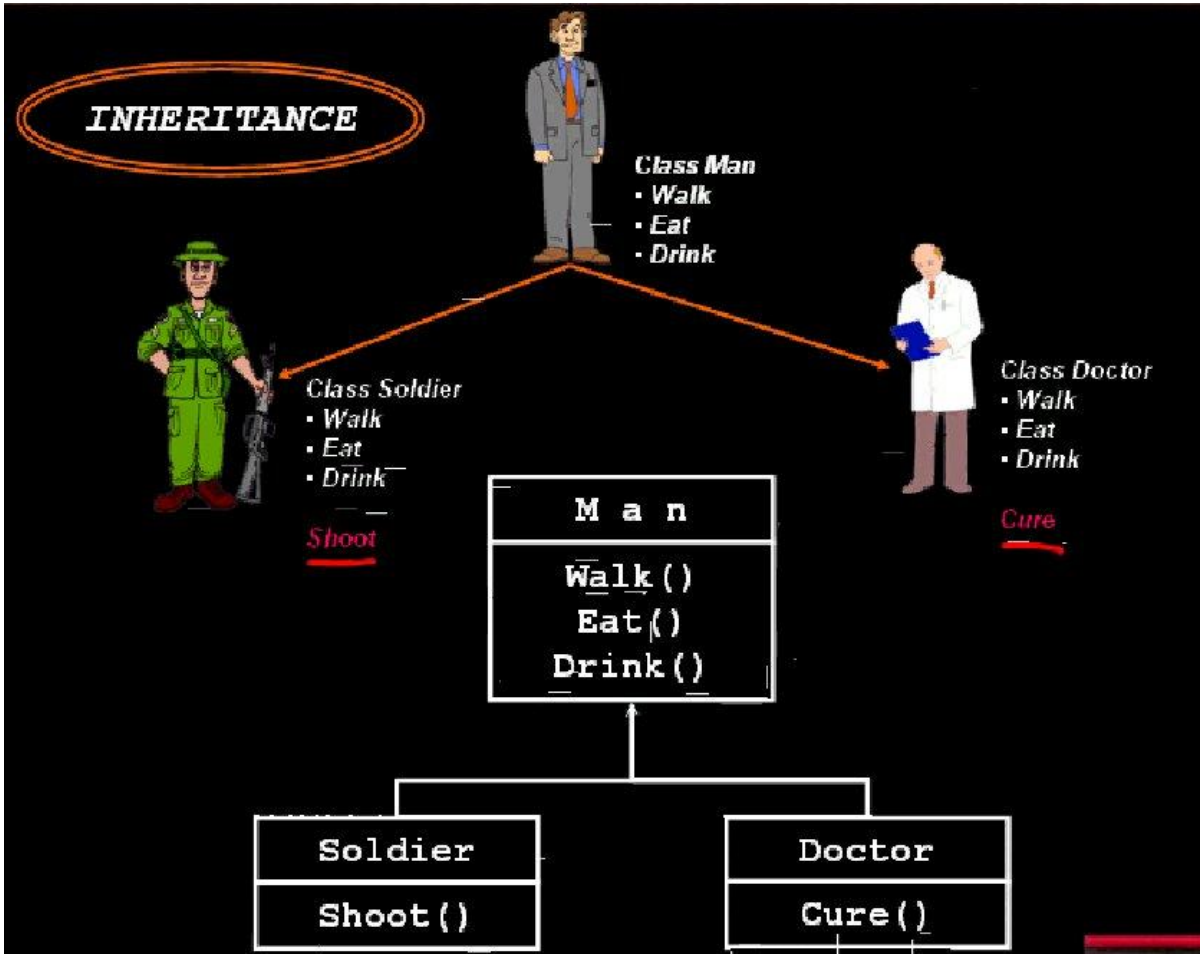
Объектно-ориентированное программирование в PHP

- **Объе́ктно-ориенти́рованное** программи́рование (в дальнейшем ООП) — парадигма программирования, в которой основными концепциями являются понятия объектов и классов.

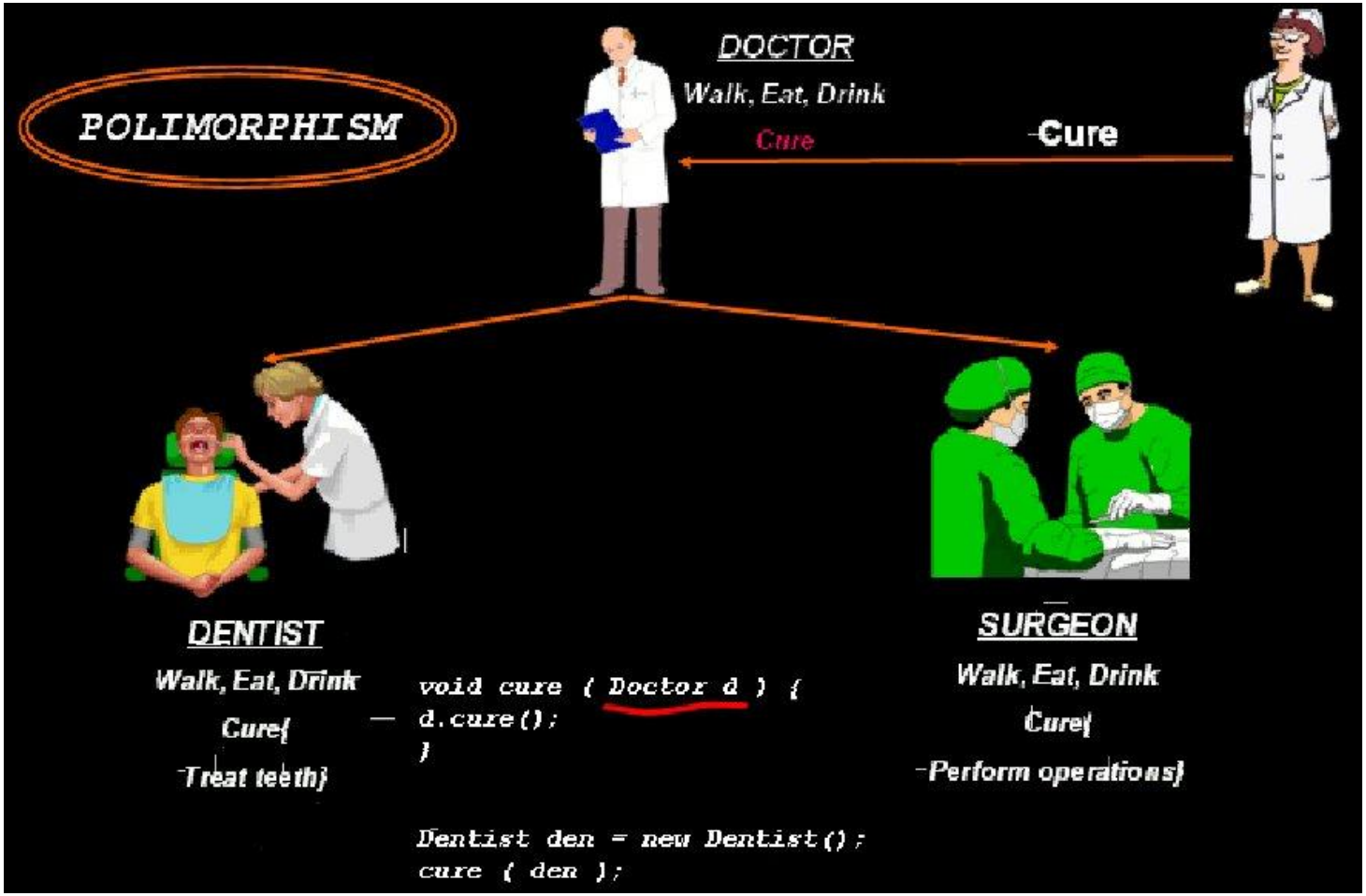
Основы ООП


Object – Oriented Programming
INHERITANCE
POLIMORPHISM
INCAPSULATION

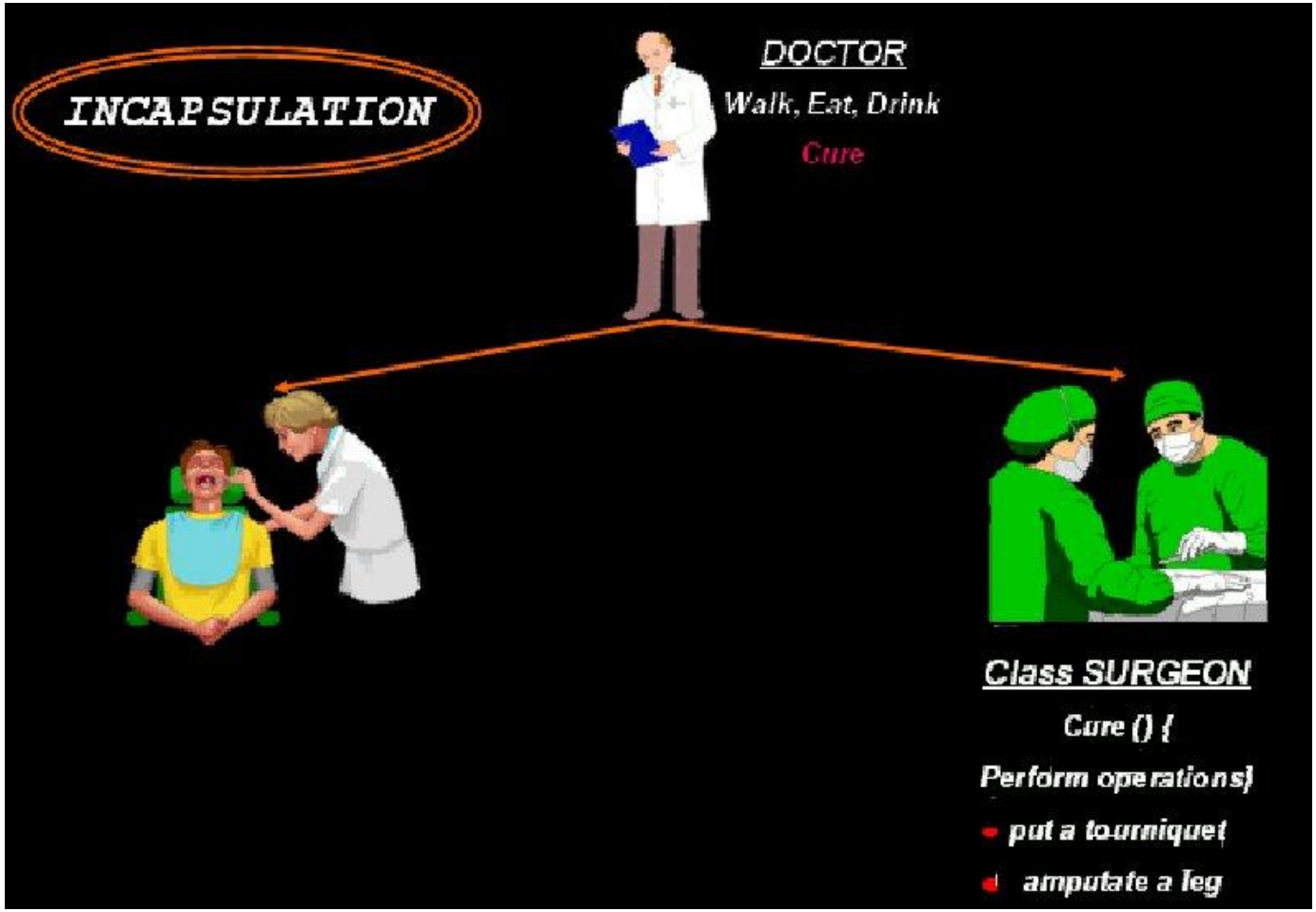
Принцип наследования



Полиморфизм



Инкапсуляция



Шаблон класса

Объект – это структурированная переменная, содержащая всю информацию о некотором физическом предмете или реализуемом в программе понятии, класс – это описание таких объектов и действий, которые можно с ними выполнять.

```
class Имя_класса{
  var $имя_свойства;
    /*список свойств*/
  function имя_метода( ){
    /* определение метода */
  }
  /*список методов*/
}
```

Пример ООП в PHP

Для обращения к полям класса и методам класса необходимо использовать ключевое слово **this**.

```php
<?
class Articles { // Создаем класс Статей
  var $title;
  var $author;
  var $description;
// метод, который присваивает значения атрибутам класса
  function make_article($t, $a, $d){
    $this->title = $t;
    $this->author = $a;
    $this->description = $d;
  }
//метод для отображения экземпляров класса
  function show_article(){
    $art = $this->title . "<br>" .
           $this->description .
      "<br>Автор: " . $this->author;
    echo $art;
  }
}
?>
```

Создание объекта класса. Вызов методов класса

Для использования класса к нему необходимо обратиться через создание объекта данного класса. Общая форма создания объекта:
$obj = new MyClass();
Используя вновь созданный объект $obj мы получаем возможность вызывать методы класса и определять поля класса. Пример:

```php
<?php
$art = new Articles;
    // создаем переменную (объект) $art
echo ($art ->title);
    // выводим свойство (название) переменной (объекта) $art
$another_art = new Articles;
    // создаем объект $another_art
$another_art->show_article();
    // вызываем метод для отображения объекта в браузер
?>
```

Конструкторы в php

Очень часто при создании экземпляра объекта на основе класса требуется выполнить какие-то базовые настройки, например установка свойств объекта. Именно для этих целей в ООП и существует метод конструктор. В версиях до **PHP 5** имя метода конструктора совпадало с именем класса к которому он относится, а начиная с версии **PHP 5** имя метода конструктора необходимо называть **__construct()** (т.е. 2 подчеркивания перед словом construct()).

```php
class MyClass {

  public function __construct() {
    echo "Я только что был создан!";
  }
}

$myObject = new MyClass();  // выведет «Я только что был создан!»
```

Параметризованные конструкторы

```
class Demo {

    var $x;

    function __construct($i){
      $x=$i;
    }
}

Demo d = new Demo(10);
Demo d2 = new Demo(20);
echo  d2->x;
    }
}
```

Наследование в PHP

- Наследование - это не просто создание точной копии класса, а расширение уже существующего класса, чтобы потомок мог выполнять какие-нибудь новые, характерные только ему функции.

Пример наследования в PHP

```php
<?php
class Parent {
  function parent_funct() { echo "<h1>Это родительская функция</h1>"; }
  function test () { echo "<h1>Это родительский класс</h1>"; }
}

class Child extends Parent {
  function child_funct() { echo "<h2>Это дочерняя функция</h2>"; }
  function test () { echo "<h2>Это дочерний класс</h2>"; }
}

$object = new Parent;
$object = new Child;

$object->parent_funct(); // Выводит 'Это родительская функция'
$object->child_funct(); // Выводит 'Это дочерняя функция'
$object->test(); // Выводит 'Это дочерний класс'
?>
```

Практика

- Разработать программу, которая возводит все элементы массива в определенную степень. В подклассе увеличить все элементы массива в два раза

Статические поля класса

В PHP используются *статические методы* которые задаются при помощи ключевого слова **static**. Для доступа к статическим полям не требуется создавать экземпляры соответствующего класса.

Пример использования статических полей класса и статических методов

```
Class Man{
  private $username;
  public static $numMan = 0;

  public function __construct($username){
        $this->username=$username;
         self::numMan++;
  }
}
echo Man::$numMan.”<br>”;
$m = new Man(“Sergey”);
echo Man::$numMan.”<br>”;
```