

主讲教师：传智.神龙教主

传智播客 PHP学院 <http://php.itcast.cn> 发布

## 面向对象编程（OOP）

OOP：Object Oriented Programming 面向对象编程

面向过程：以程序执行顺序为主线来开发

面向对象：将一个项目看成有多个对象组成。通过多个对象组成项目

多学一招：OOAD

OOAD就是OOA和OOD

OOA：Object-Oriented Analysis 面向对象分析

OOD：Object-Oriented Design 面向对象设计

### 对象

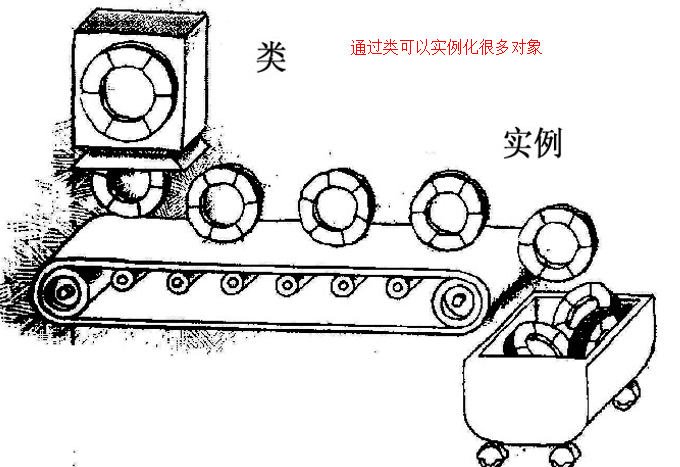
对象是有属性和动作（方法）组成的

### 类

类是相同属性和方法的集合（抽象）

脚下留心：在程序中我们需要的是属性（变量）和方法（函数）来保存数据和执行逻辑，要调用属性和方法必须要有对象，要有对象必须先有类。

开发顺序：类——对象——属性和方法



脚下留心：类分为父类和子类

类的本质是一个复杂的数据类型，对象的本质就是复杂变量

## 在PHP中实现类

语法：

Class 类名{

//属性

//方法

}

命令规则：

1. 类名用帕斯卡命名法
2. 以字母、下划线开头，后面跟的是字母、数字、下划线
3. 不能用PHP关键字做变量名
4. 类名不区分大小写

### 属性（也叫成员变量）

属性其实就是变量

属性=访问修饰符+变量

//类

class Student {

public $name='李白'; //访问修饰符 属性

public $sex='男';

}

//实例化对象

$stu1=new Student(); //实例化一个对象，并且将对象付给$stu1变量

$stu2=new Student();

//给变量赋值

$stu1->name='tom'; //stu1的name和stu2的name不是同一空间

$stu2->name='berry';

//调用属性

echo $stu1->name,'<br>';

echo $stu2->name,'<br>';

echo '<br>---------增加属性---------------<br>';

$stu1->add='北京';

var\_dump($stu1);

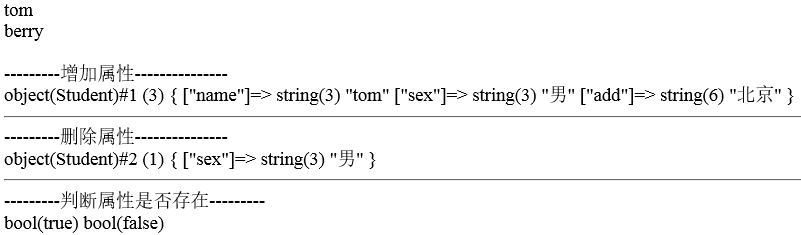
echo '<hr>---------删除属性---------------<br>';

unset($stu2->name);

var\_dump($stu2);

echo '<hr>---------判断属性是否存在---------<br>';

var\_dump(isset($stu1->name),isset($stu2->name));



### 方法

方法就是函数

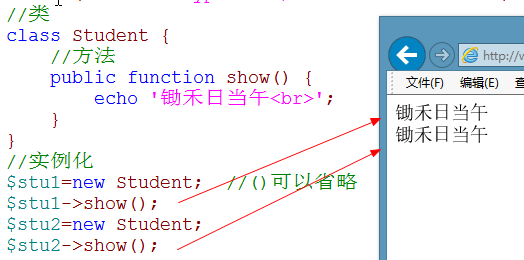
方法=访问修饰符+函数

语法:

访问修饰符 function 方法名(参数){

//方法体

}



### new

new用来实例化对象



### 调用属性和方法 ->

在OOP中，要调用属性和方法必须要有对象



## 类、对象、属性、方法在内存中的分布

class Student {

public $name;

public function show() {

echo '锄禾日当午<br>';

}

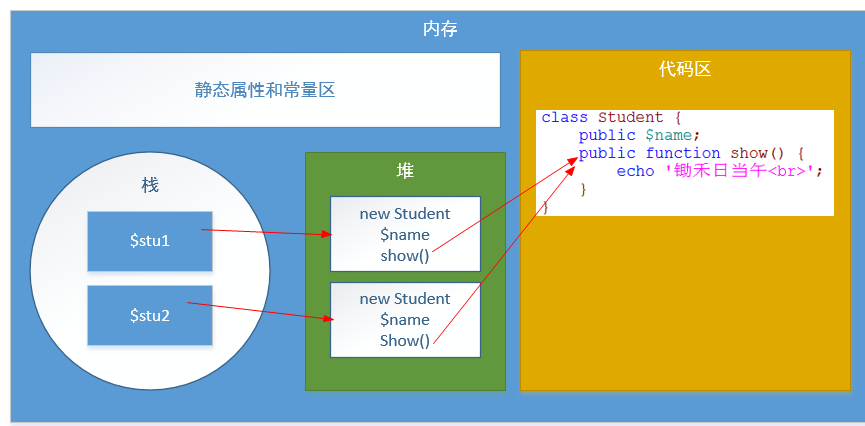
}

$stu1=new Student;

$stu2=new Student;

$stu1->show();

$stu2->show();



## 命名法

### 帕斯卡命名法（大驼峰）

首字母大写，后面的每个单词的首字母大写

比如：UserName SheIsAGirl

### 骆驼命名法（小驼峰）

1. 首字母小写，后面的每个单词的首字母大写
2. 每个单词的首字母都小写，单词之间用下划线连接

比如：userName user\_name \_user\_name

### 匈牙利命名法

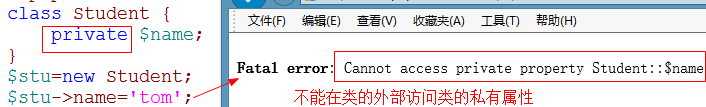
构成：变量类型+变量名字

比如：iAge fnShow sStu aStu

|  |  |  |  |
| --- | --- | --- | --- |
| 前缀 | 含义 | 前缀 | 含义 |
| iAge | 整形的年龄 | fnShow | 函数show |
| sStu | 学生字符串 | aStu | 学生数组 |

## 访问修饰符

|  |  |
| --- | --- |
| 修饰符 | 描述 |
| public(公有的) | 可以在类的内部和外部访问 |
| private(私有的) | 只能在类的内部访问 |
| protected(受保护的) | 在类的内部和子类中访问（在整个继承链上访问） |



多学一招：在OOP中，属性一般都是私有的，通过公有的方法对私有的属性进行赋值和取值。（目的为了过滤数据）

class Student {

private $name; //属性也叫成员变量

private $sex;

//赋值

public function setInfo($name,$sex) {

if($name==''){

echo '姓名不能为空';

exit;

}

if($sex!='男' && $sex!='女'){

echo '性别不正确';

exit;

}

//赋值

$this->name=$name; //$this表示调用当前方法的对象

$this->sex=$sex;

}

//取值

public function getInfo() {

echo '姓名：'.$this->name,'<br>';

echo '性别：'.$this->sex,'<hr>';

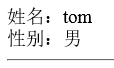
}

}

$stu=new Student;

$stu->setInfo('tom','男');

$stu->getInfo();

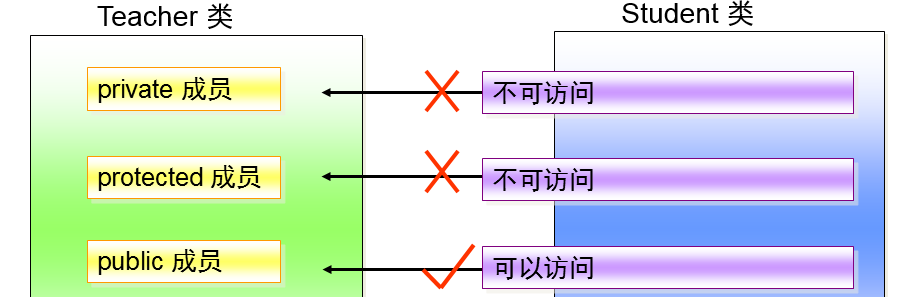


脚下留心：$this表示调用当前方法的对象

## 封装

访问修饰符的作用是用来做封装的。

封装的目的是用来有选择性提供数据和访问



## 构造方法

构造方法也叫构造函数。在实例化对象的时候自动执行。用来完成初始化工作。

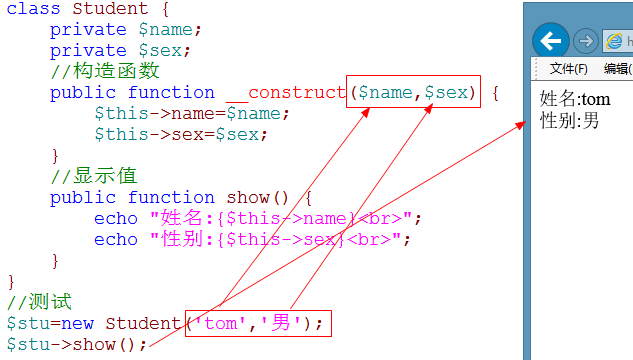
语法：

[public] function \_\_construct(参数){ //注意是两个\_

}

public可以省略，默认构造函数访问修饰符就是public的。

例题：



## 析构方法（析构函数）

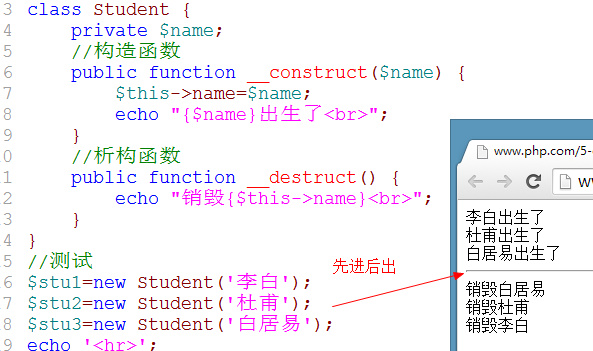
当对象销毁的时候自动调用；析构函数不能带有参数。

语法：

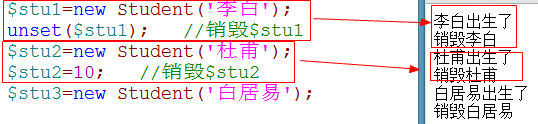
public function \_\_destruct(){

}

例题



//手动销毁对象

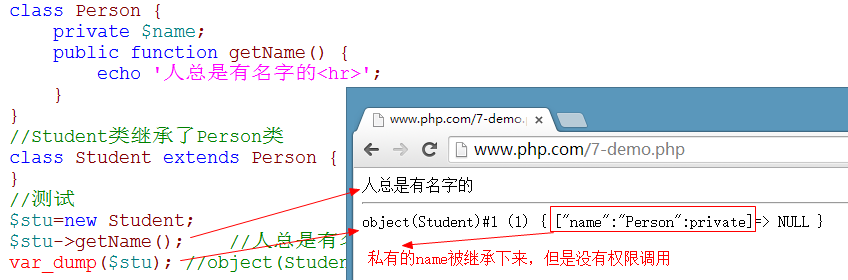


### 用析构函数模拟实现垃圾回收



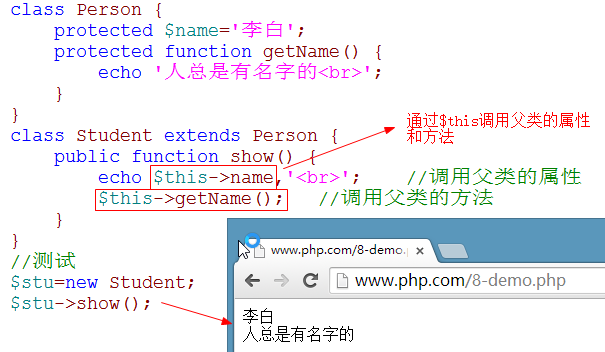
## 继承（extends）

1. 具有层次性
2. 子类继承了父类的属性和方法，实现了代码的可重用性
3. 父类和子类是相对的



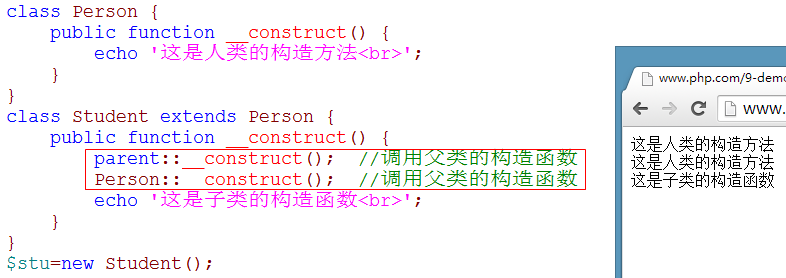
脚下留心：子类继承了父类的所有属性和方法，但是不一定有权限调用

### 在子类中调用父类的属性和方法



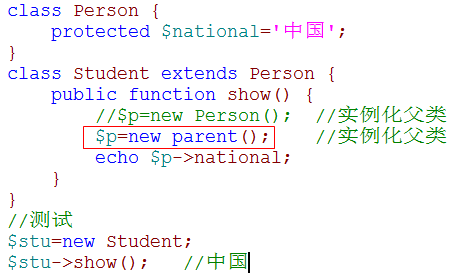
### 调用父类的构造方法

1. 如果子类没有构造函数，就调用父类的构造函数
2. 如果子类有构造函数就调用自己的构造函数
3. 通过parent来调用父类的构造函数



### parent

表示父类的名字



### 子类给父类的构造函数传递参数

class Person {

protected $name;

protected $sex;

public function \_\_construct($name,$sex) {

$this->name=$name;

$this->sex=$sex;

}

}

class Student extends Person {

private $score;

public function \_\_construct($name,$sex,$score) {

parent::\_\_construct($name,$sex); //调用父类的构造函数

$this->score=$score;

}

//显示值

public function show() {

echo "姓名：{$this->name}<br>";

echo "性别：{$this->sex}<br>";

echo "成绩：{$this->score}";

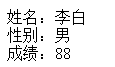
}

}

//测试

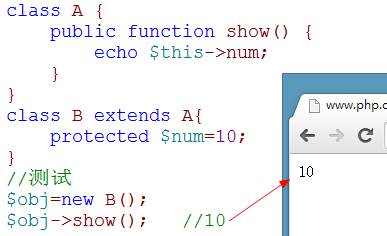
$stu=new Student('李白','男','88');

$stu->show();



### protected

protected修饰的成员可以在整个继承链上访问



脚下留心：PHP不允许多重继承，也就是还说C不能同时继承A和B，可以将C继承B，B再继承A。

## $this

$this表示当前对象的引用

class Goods {

public function \_\_construct() {

var\_dump($this);

}

}

class Book extends Goods {

}

//测试

new Goods(); //object(Goods)#1 (0) { }

echo '<hr>';

new Book(); //object(Book)#1 (0) { }

判断题：$this表示$this所在类的对象（错误）