* 正则表达式
  + 要求：
    - 1，能看懂所有的正则
    - 2，能写出简单的正则
* 错误
  + 1.错误级别
  + 2，错误配置
* 时间
  + 1.时区配置
  + 2，常用函数
  + 万年历
* 文件系统
  + 目录
  + 文件
  + 1.递归遍历目录
  + 2，递归统计目录大小
  + 3，扩展-留言板
* 上传&下载
  + 单文件上传
  + 多文件上传
  + 上传的要求：
    - 1.表单提交方式必须是post
    - 2.必须有属性：enctype=”multipart/form-data”
  + 下载
    - 默认下载
    - 强制下载

# 面向对象编程

## 3个目标|优势

* 重用性
* 扩展性
* 灵活性

## 3个特点

* 封装
* 继承
* 多态

## 类

* 什么是类？
* 对某一类具有相同特性或者行为功能的抽象就是类
* 特性：眼，鼻，耳朵，嘴巴，手，，，
* 行为：吃，喝，拉，撒，睡，，，
* 类是不能直接用的，类只是一个模子，模具

### 类的声明

* 格式：

**class 类名**

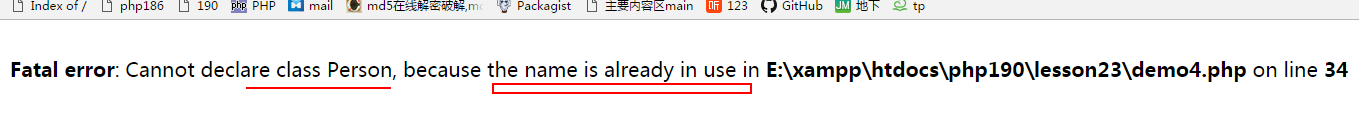
**{**

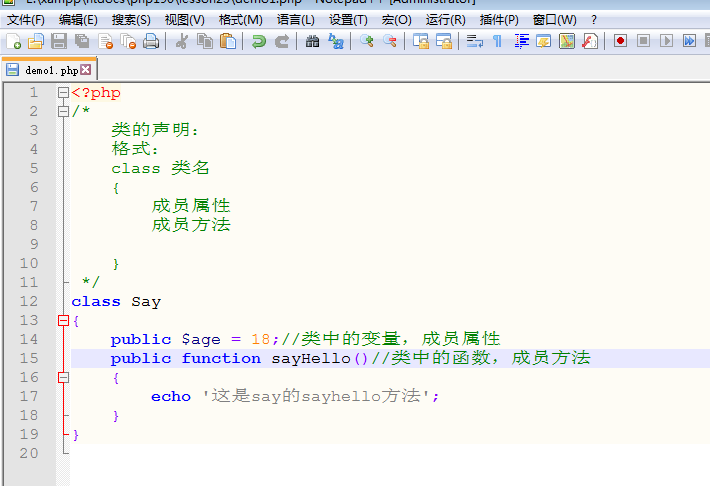
**成员属性//变量**

**成员方法//函数**

**}**

* **命名规则：**
  + **1.可以使用字母，数字，下划线**
  + **2，数字不能开头**
  + **3，下划线开头慎用**
  + **4.命名要有意义**
  + **5.命名法：大驼峰：GetElementById**
  + **6.系统保留关键字，词，不让用**
  + **7.类名不能重复**





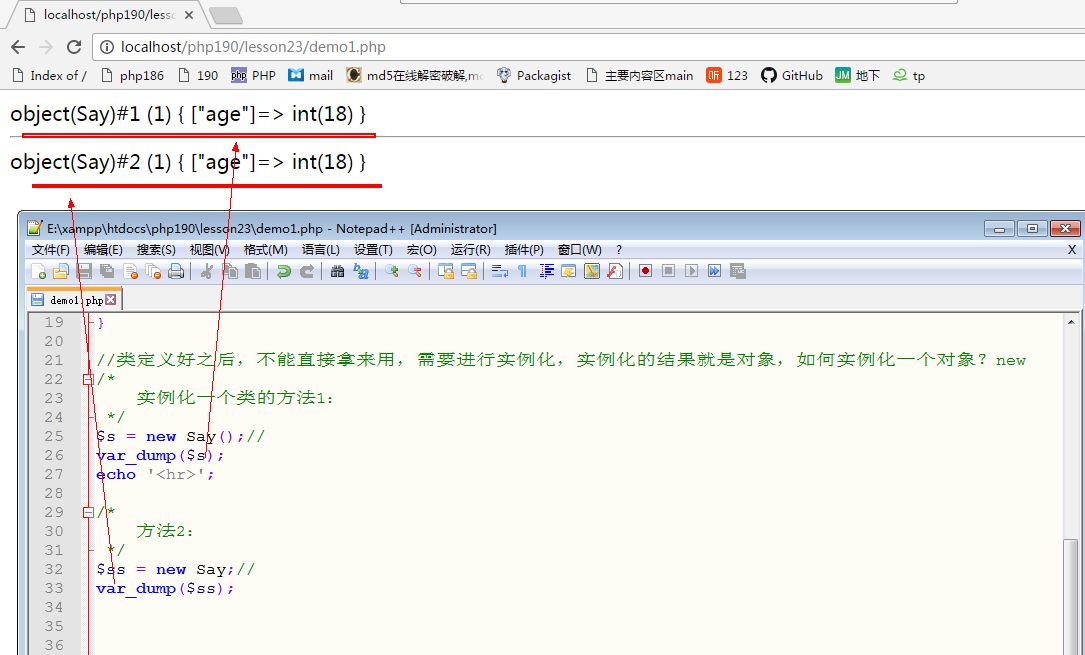
* **成员属性：类中定义的变量**
* **成员方法：类中定义的函数**
* **注意：所有的成员属性和成员方法必须有修饰**

## 对象

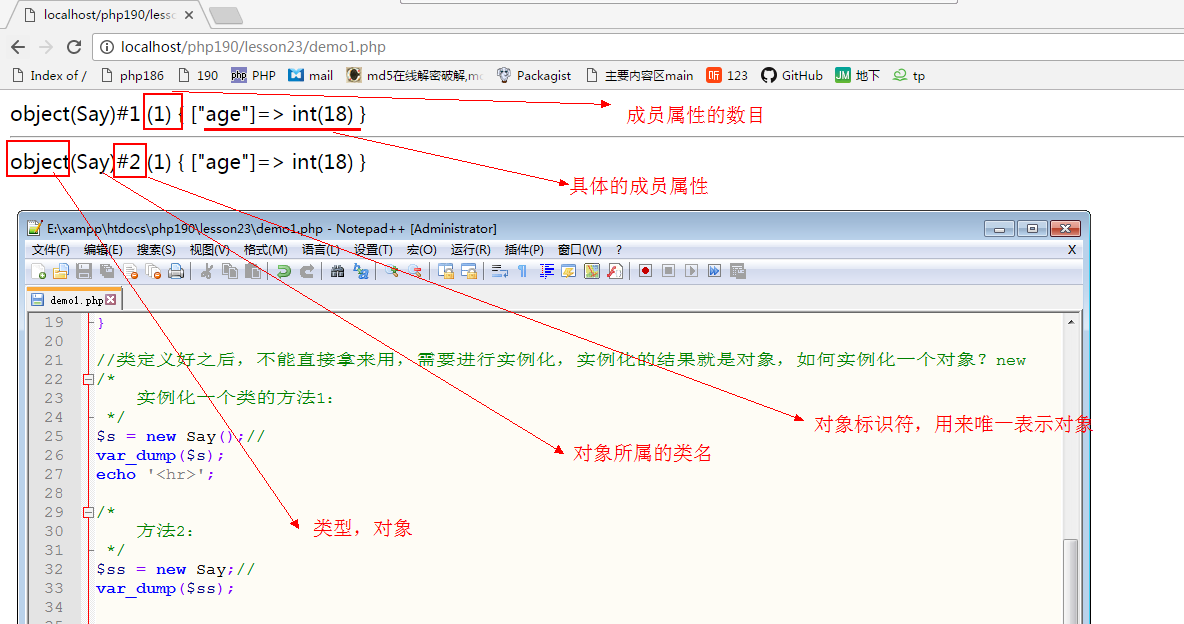
* 什么是对象？
  + 类的实例化结果，就是对象
  + 铁水注入模子，冷却后的就是锅，这个锅就是对象

### 类的实例化-就是搞一个对象出来

* 使用关键词new
  + 格式：$对象= new 类名();--推荐的



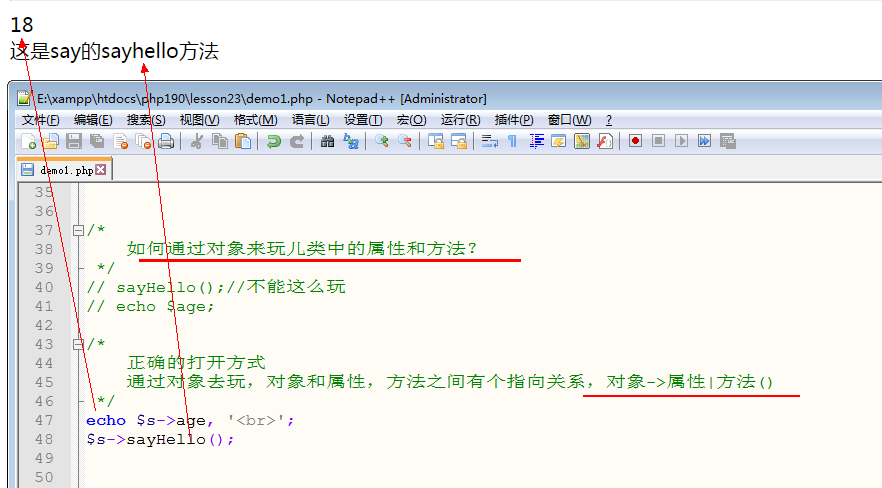
### 具体分析一个对象



* **注意:**
  + **1.对象标识符用来标识具体的对象，如果标识符一致，表名是一个对象**
  + **2.成员方法是打印不出来的，能打印的只能是属性**

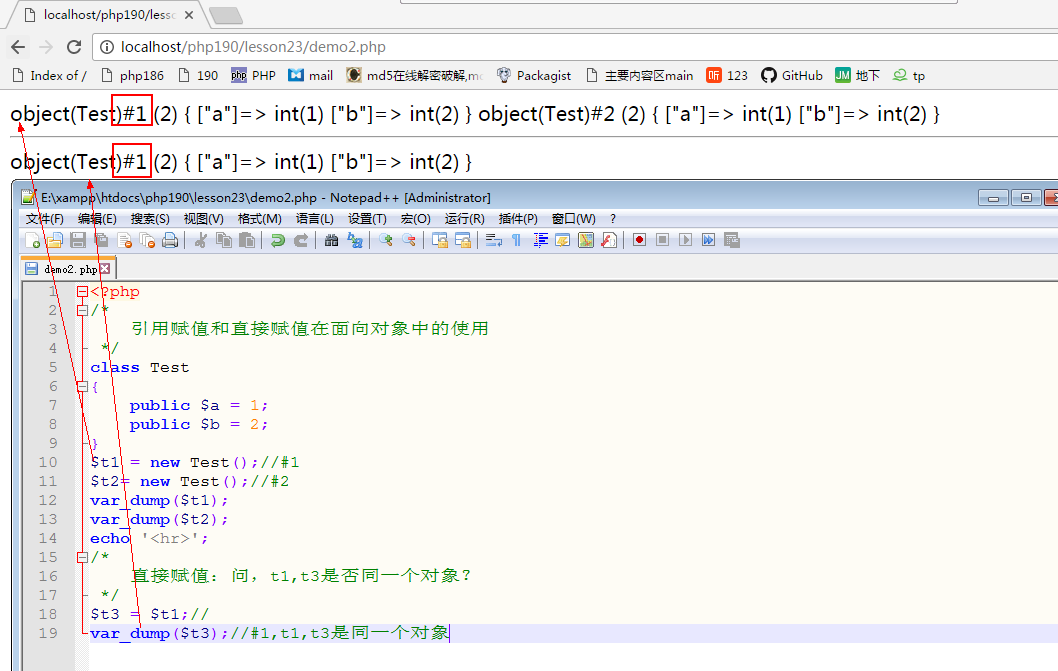
### 如何通过对象玩儿类中的属性和方法

* 格式：
  + **对象->属性|方法()**
  + **注意：属性前面不需要$**
  + **方法后面一定要有()**



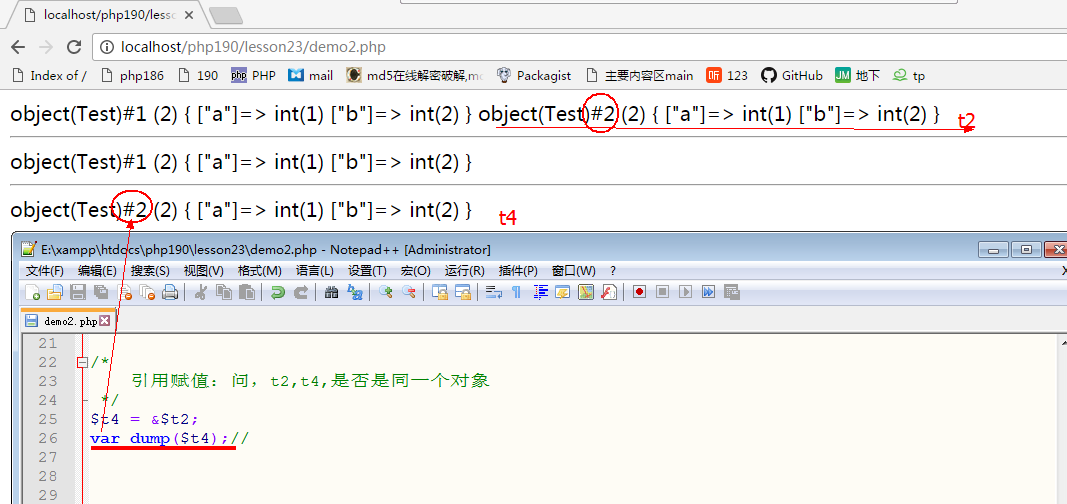
### 对象的复制|引用赋值和直接赋值对对象的影响

* 示例1：直接赋值



* **结论：直接赋值的结果，他们是同一个对象**

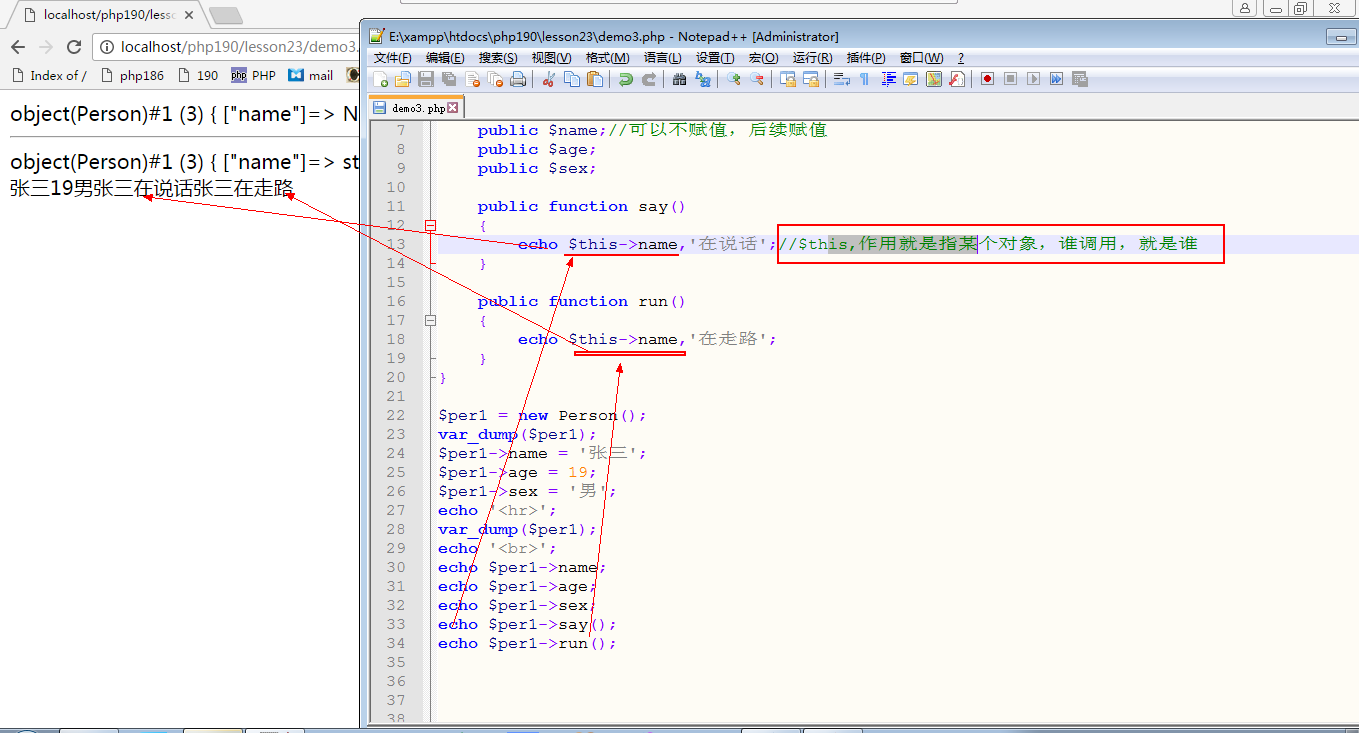
**示例2：**



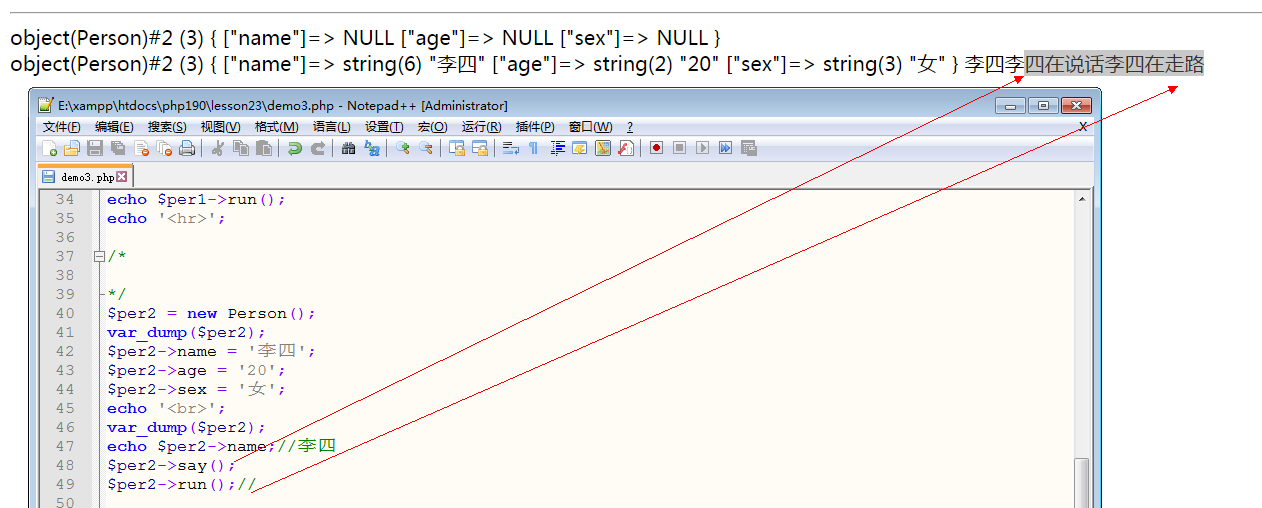
* **结论：引用赋值的结果，同样也是同一个对象**
  + **引用赋值也好，直接赋值也罢，结果都是同一个对象，看什么呢？看标识符也就是#XXX**
  + **引用赋值和直接赋值都是直接copy对象标识符，所以结果都是指向同一个对象**
  + **那除了new之外，还有没其他办法复制不一样的对象呢？**
    - **可以的，后面再说。对象的克隆**

### $this

* 作用：
  + **在类内部调用属性的时候，$this就表示当前的对象，谁调用，就是哪个对象**



* 示例：



## 构造函数&析构函数

### 构造函数：

* 特点：对象实例化的时候自动调用，并且是第一个调用的函数
* 作用：可用来做初始化的赋值
* 格式：

**Public function \_\_construct($形参1，$形参2，$形参N)**

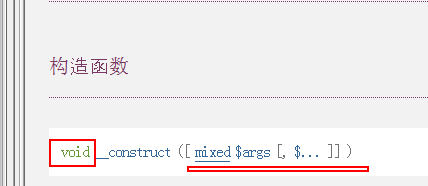
**｛**

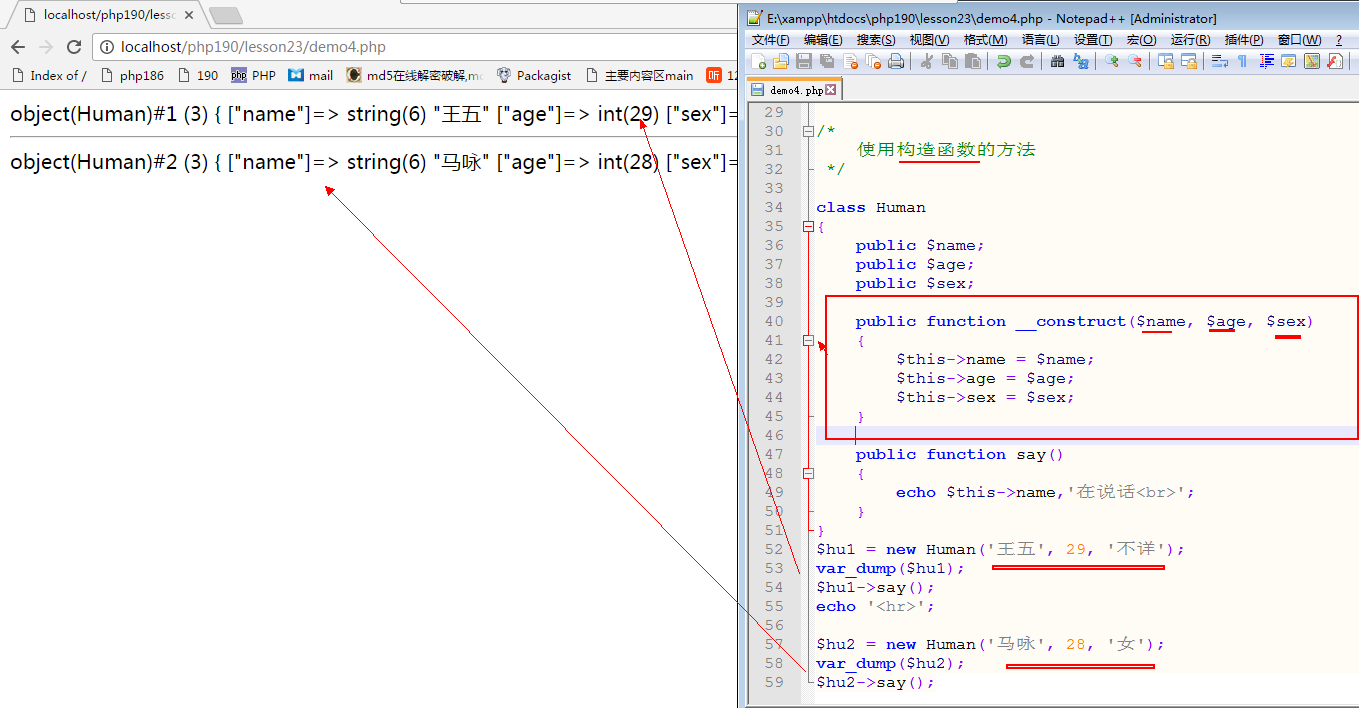
**$this->形参1 = $形参1；**

**$this->形参2 = $形参2；**

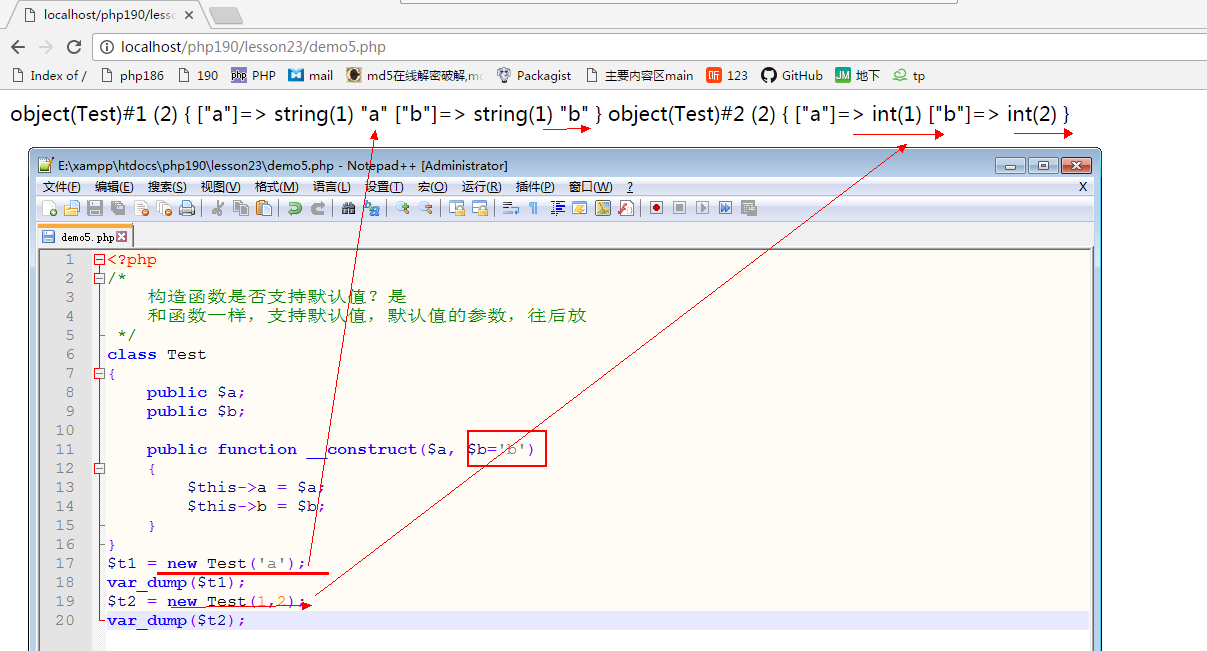
**$this->形参N = $形参N；**

**｝**



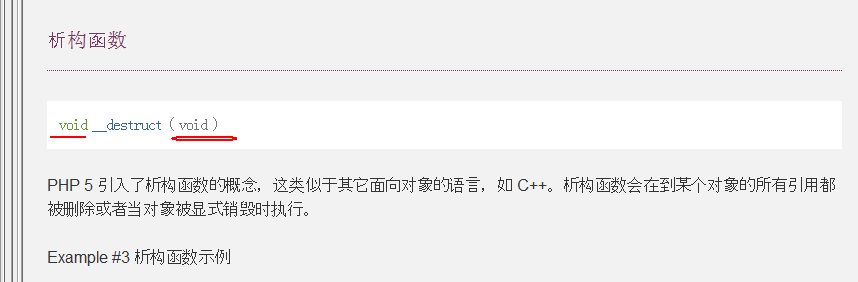


* **如果想用默认值怎么办？比如说，咱班都是男的，那么sex是否能给默认值？**
  + **可以，和函数使用默认值一样，使用默认值的参数往后放**

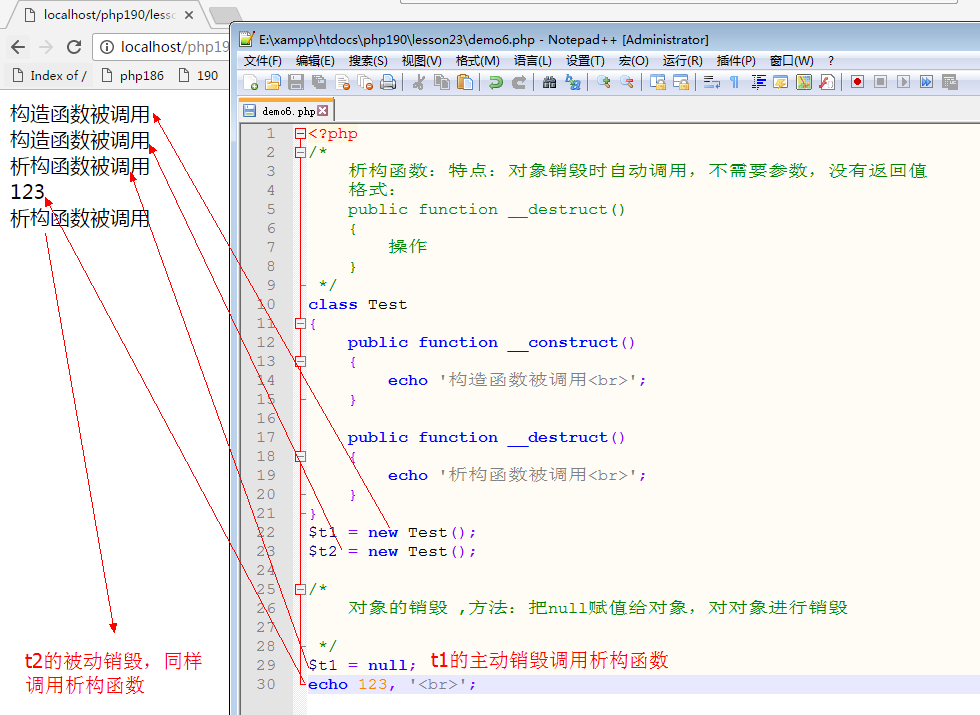


### 析构函数

* **特点：在对象被销毁的时候，自动调用**
* 作用：可以用来在对象销毁的时候做一些操作，比如说：数据库连接的关闭，文件句柄的关闭，目录句柄的关闭等
* 注意：析构函数不需要参数

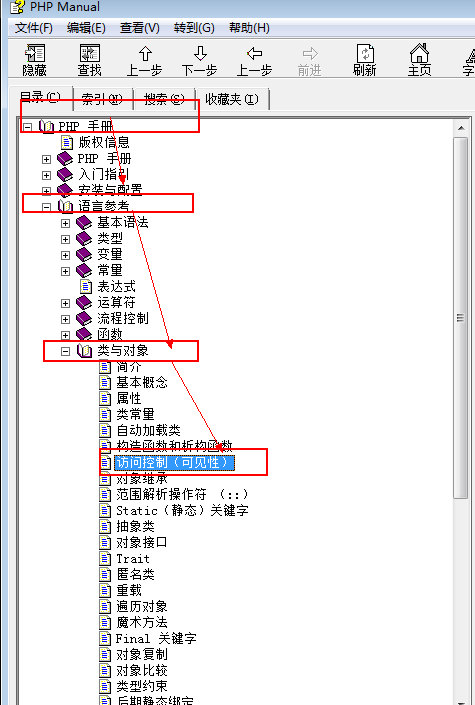


* 示例：



* **对象的销毁**
  + **对象的销毁，其实是断开对象名和标识符之间的联系**
  + **主动销毁：$对象 = null**
  + **被动销毁：代码执行到尾部，自动销毁**
  + **不论什么格式的销毁，都会自动调用析构函数**

# 封装



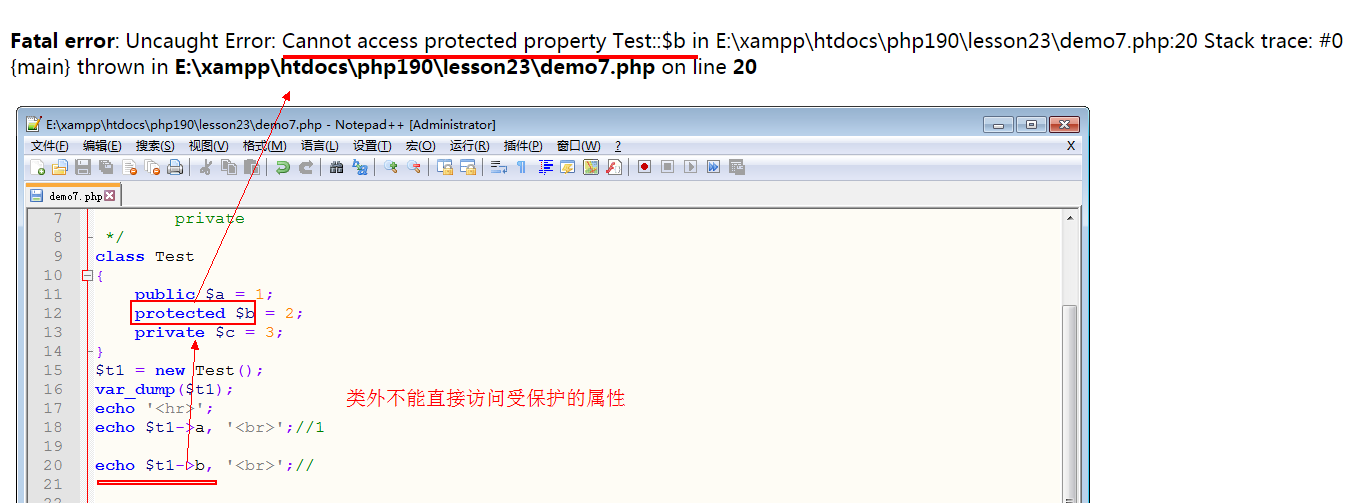
* 目的：隐藏内部具体的细节，进行访问的控制
* 例子：比如说：电脑的封装，人的封装
* 如何实现？
  + 通过修饰符
  + 注意：**这些修饰符，都是只能修饰成员属性和成员方法，不能修饰类**
    - **Public：公共**
    - **Protected：受保护的**
    - **Private：私有的**

## Public:

* 公共的
* 特点：
  + 类内，类外，其他类中都能用
  + 没有限制

## Protected

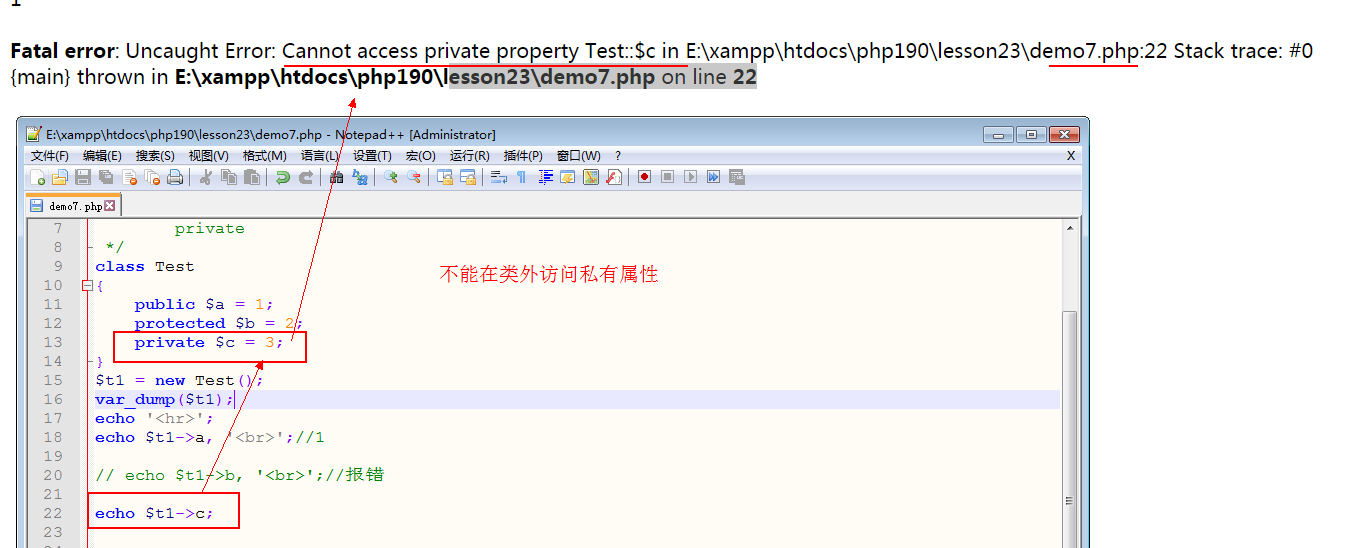
* 受保护的
* 特点：
  + 只能类内，子类中好使，类外，其他类中不好使

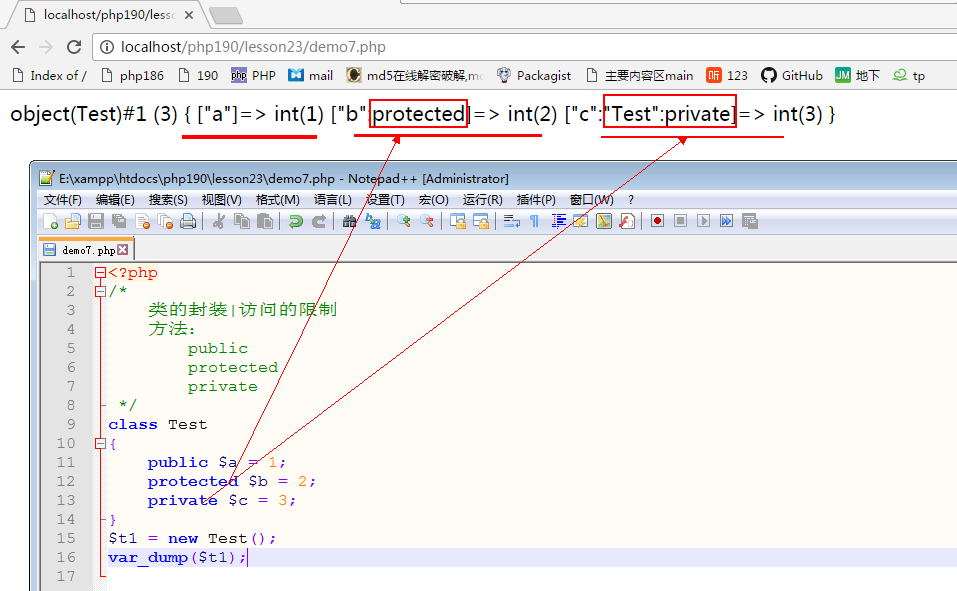


* + 有部分限制

## Private

* 私有的
  + 特点：
  + 只能类内使用，类外，子类中，其他类中都不好使
  + 完全限制

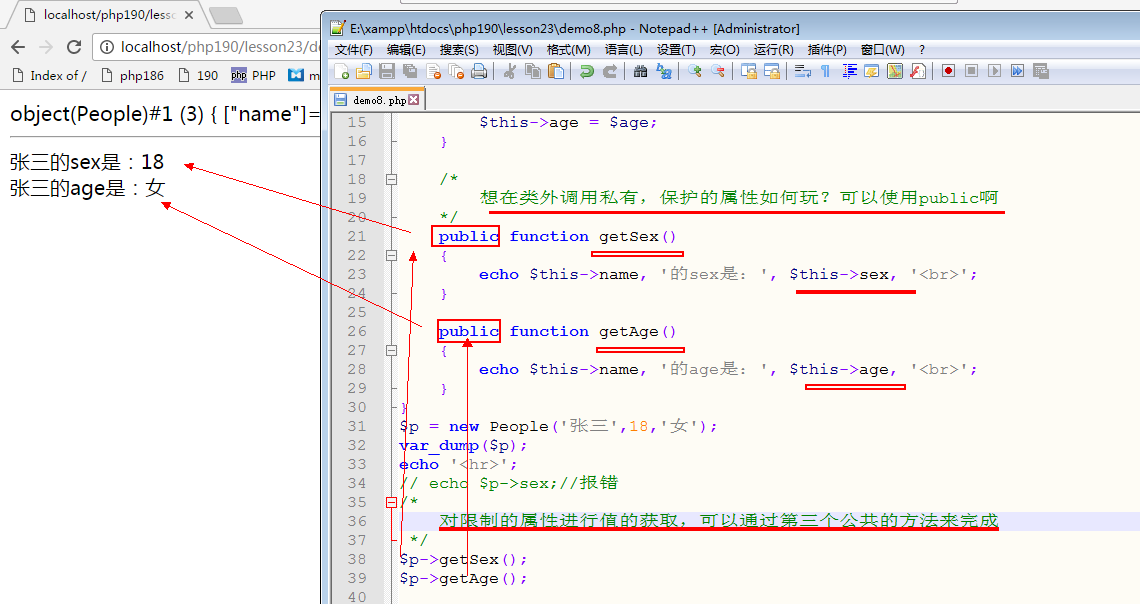




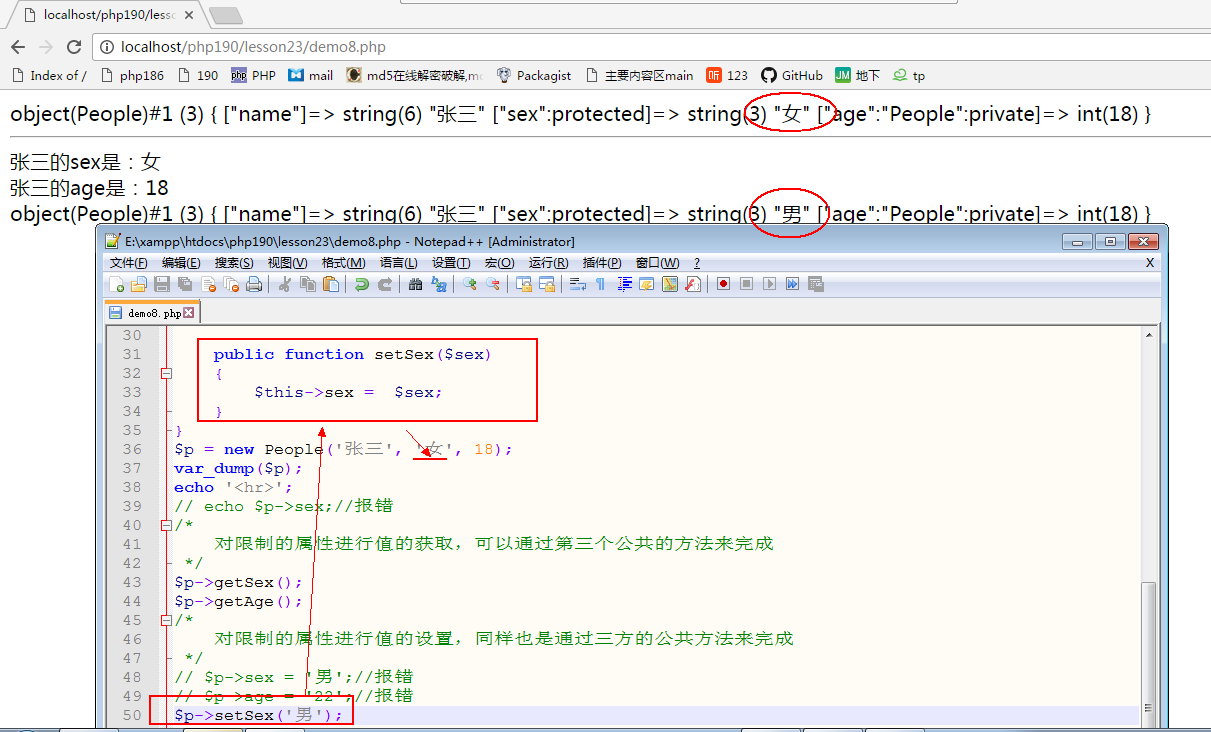
### 如何调用受限制的属性和方法？

**对限制了的属性和方法可以使用第三方的公共的方法来进行调用**

**示例1：值的获取**



**示例2：值的设置**



* **带来的问题就是：如果有N个受限制的属性，是不是对于这个N个值的设置，访问，销毁，检测都得用三方的公共方法，也就意味着，需要额外的4N个方法**