面向对象第2天

[学习目标 1](#_Toc522031966)

[静态属性和静态方法 2](#_Toc522031967)

[1、概述 2](#_Toc522031968)

[2、举例说明 2](#_Toc522031969)

[4、self关键字使用 2](#_Toc522031970)

[综合案例 3](#_Toc522031971)

[OOP中内存的分配情况 5](#_Toc522031972)

[1、为什么使用var\_dump打印对象时，只能看到成员属性呢？ 5](#_Toc522031973)

[2、OOP中内存的分配情况 6](#_Toc522031974)

[3、什么是值传递？ 6](#_Toc522031975)

[4、什么是引用传递？ 7](#_Toc522031976)

[5、其它类型变量使用引用传递 8](#_Toc522031977)

[类的封装性 9](#_Toc522031978)

[1、什么是类的封装性 9](#_Toc522031979)

[2、访问权限修饰符介绍 9](#_Toc522031980)

[3、实例：封装一个简单的数据库工具类 9](#_Toc522031981)

[类的继承性 10](#_Toc522031982)

[1、什么是类的继承性？ 10](#_Toc522031983)

[2、继承的语法格式 11](#_Toc522031984)

[3、单继承和多继承 11](#_Toc522031985)

[4、实例：单继承的简单演示 12](#_Toc522031986)

[5、parent关键字 12](#_Toc522031987)

[类的多态 13](#_Toc522031988)

[1、什么是类的多态？ 13](#_Toc522031989)

[2、方法重写的要求 14](#_Toc522031990)

[3、实例：方法重写的要求 14](#_Toc522031991)

[4、实例：商品子类重写商品基础类中的方法 15](#_Toc522031992)

# 学习目标

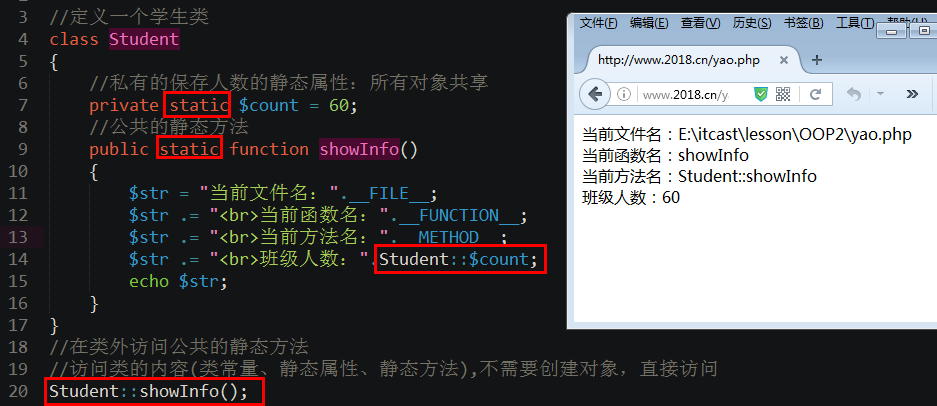
* OOP中内存的分配情况
* 类的封装性
* 类的继承性
* 类的多态性
* 最终类和最终方法

# 静态属性和静态方法

## 1、概述

* static关键字修饰的属性，就是静态属性；
* static关键字修饰的方法，就是静态方法；
* 静态属性：就是类的属性，与类相关，与对象无关；
* 静态方法：就是类的方法，与类相关，与对象无关；
* 类的东西(类常量、静态属性、静态方法)，通过"**类名::**"来调用；
* 静态属性或静态方法，在内存中只有一份，被所有对象去共享；
* 静态属性或静态方法的好处：就是为了节省内存。例如：创建了100个对象，而静态属性只有一份。
* 静态属性和类常量的区别：类常量在一次HTTP请求过程值永远不变；但是静态属性可以改变。
* 静态属性和静态方法，都可以加权限控制符，而类常量没有权限。

## 2、举例说明

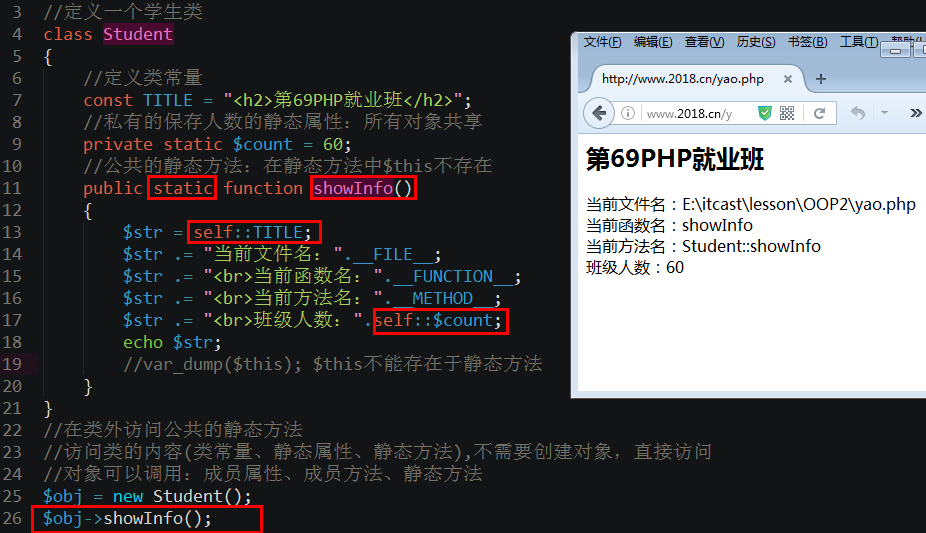


## 4、self关键字使用

* $this是指向当前对象的指针，而self是指向当前类的指针；
* $this关键字用来调用对象的属性和方法；
* self用来调用类常量、静态属性、静态方法；
* $this关键字只能在成员方法中使用；
* self关键字可以在成员方法和静态方法中使用；

**提示：对象可以调用：成员属性、成员方法、静态方法**

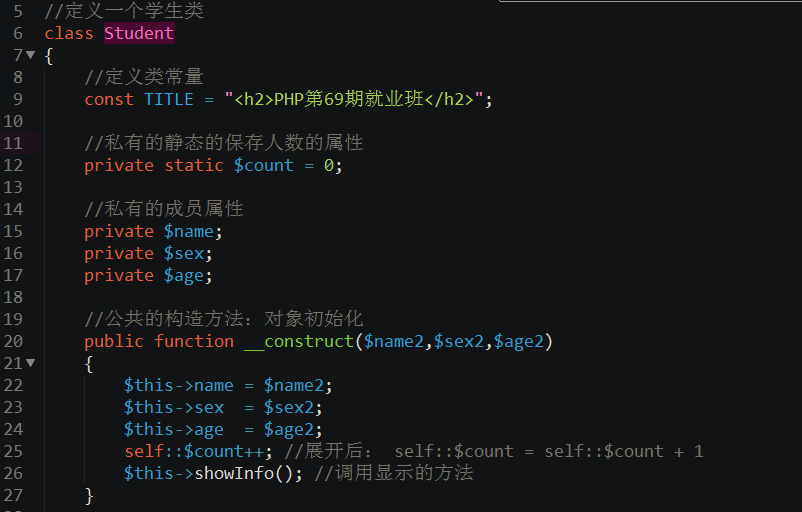
**提示：self可以调用：类常量、静态属性、静态方法、成员方法**

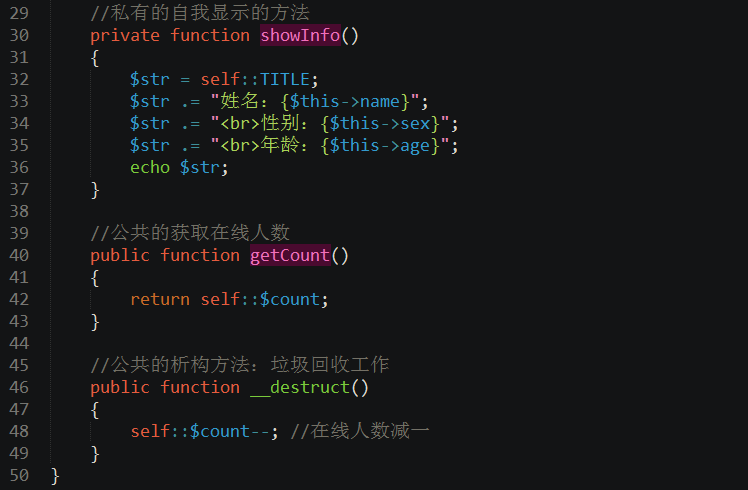


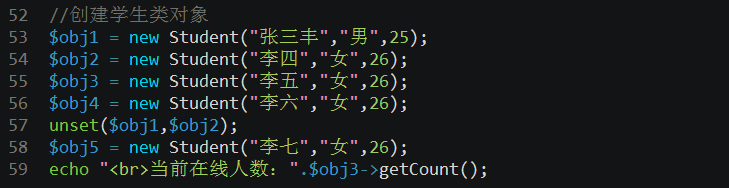
# 综合案例

**设计一个“学生类”，要求如下：**

定义一个“学生类”，并由此类实例化两个“学生对象”。该类包括姓名，性别，年龄等基本信息，并至少包括一个静态属性（表示总学生数）和一个常量，以及包括构造方法和析构方法。该对象还可以调用一个方法来进行“自我介绍”（显示其中的所有属性）。构造方法可以自动初始化一个学生的基本信息，并显示“ｘｘ加入传智，当前有xx个学生”。

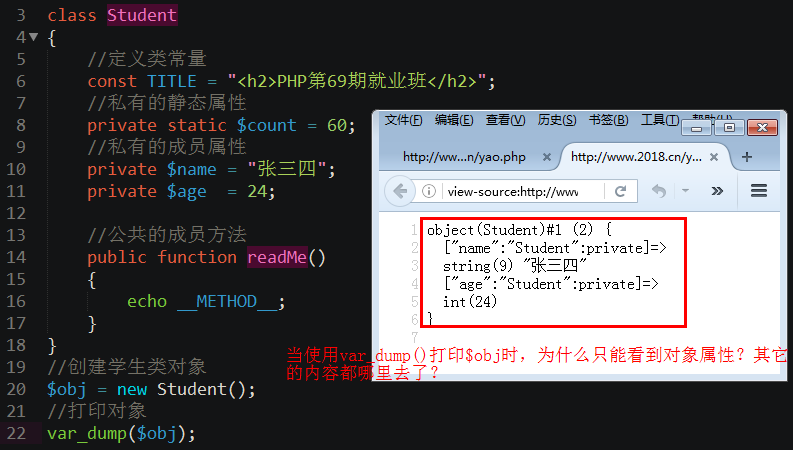




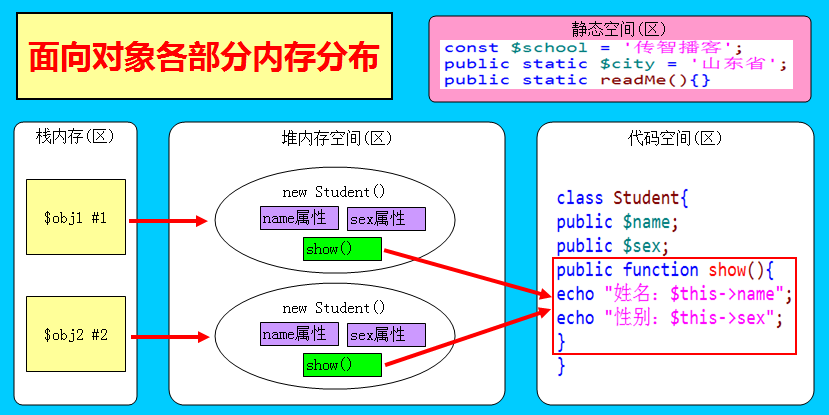


# OOP中内存的分配情况

## 1、为什么使用var\_dump打印对象时，只能看到成员属性呢？

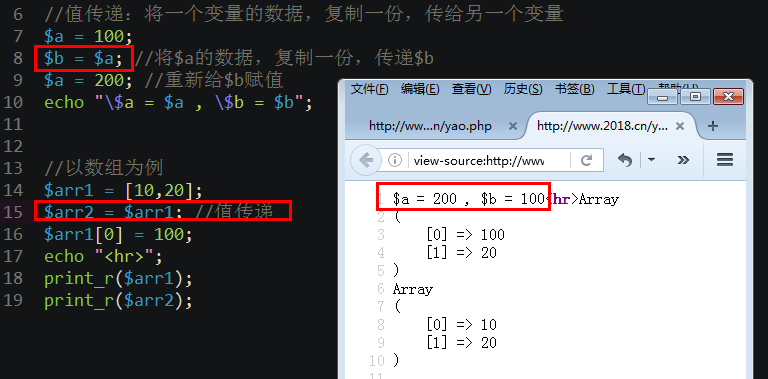


## 2、OOP中内存的分配情况

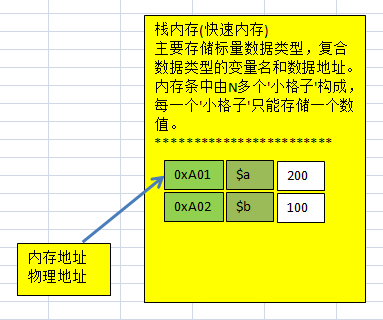


## 3、什么是值传递？

* 所谓“值传递”，就是将一个变量的“数据”或“值”，复制一份，传递给另一个变量；
* 这两个变量之间没有任何关系，修改其中一个变量的值，另一个变量的值不受影响；
* 默认情况下，PHP值传递的数据类型有：字符串型、整型、浮点型、布尔型、数组型、NULL。

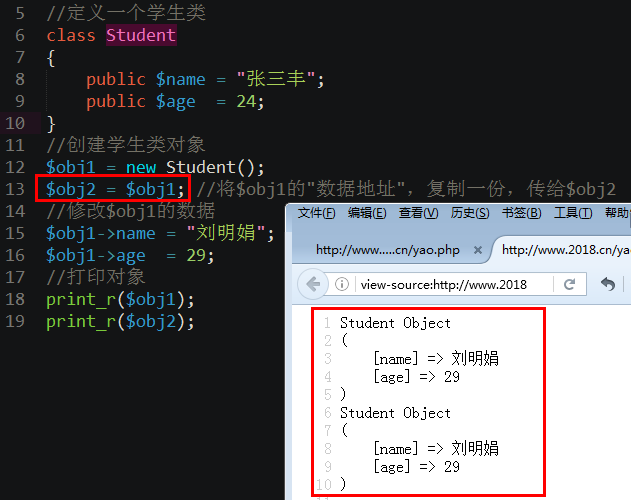
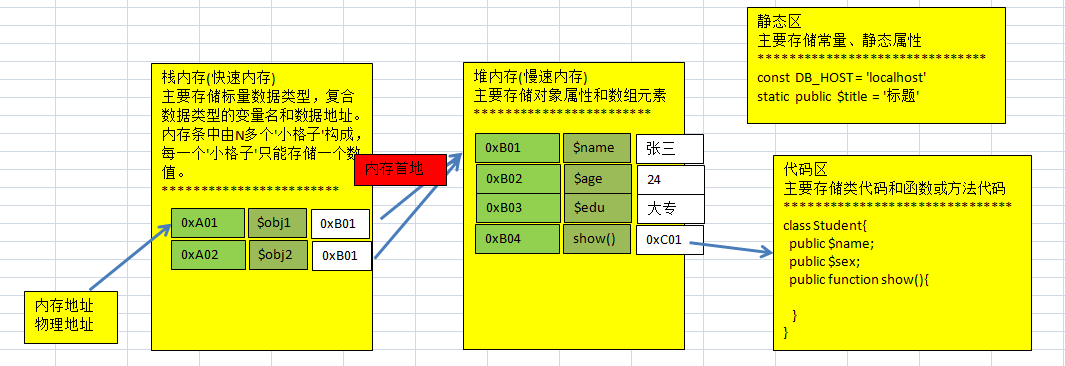


**值传递在内存中如何表现？**



## 4、什么是引用传递？

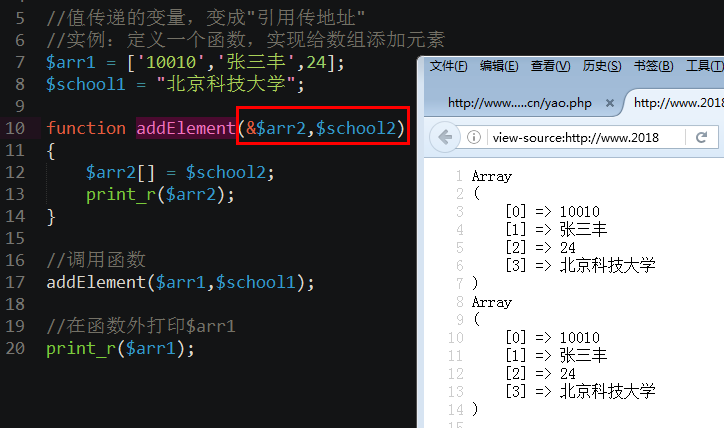
* 所谓“引用传递”，就是将一个变量的“数据地址”，复制一份，传递给另一个变量；
* 这两个变量指向“同一个地址”，修改其中一个变量的值，另一个变量的值也会受影响；
* 默认情况下，PHP引用传递的数据类型有：对象和资源。

对于海量数据，使用“引用传地址”，要比“传值”速度快。

## 5、其它类型变量使用引用传递

如果其它类型变量使用引用传递，只需要在引用的变量前加“&”符号即可。



**提示：$obj2 = $obj1 和$obj2 = &$obj1效果是一样的。**

# 类的封装性

## 1、什么是类的封装性

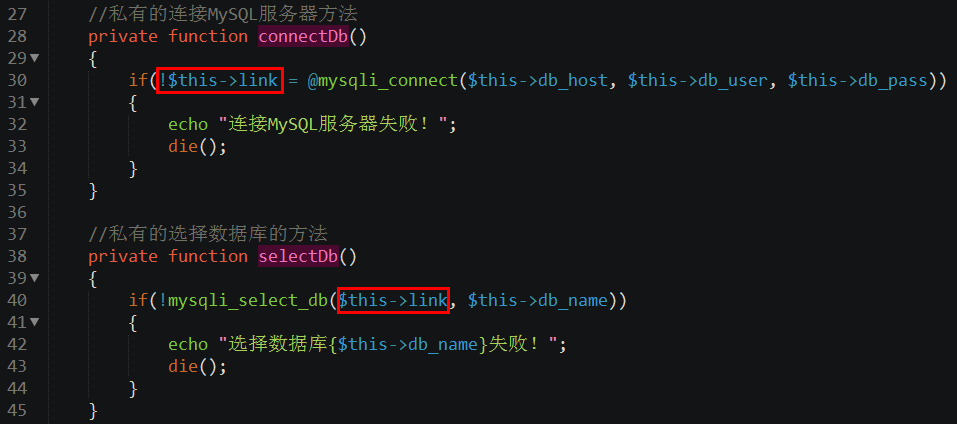
* 类的三大特性：封装性、继承性、多态性；
* 封装性：将敏感的数据保护起来，不被外界访问；还可以理解为，将一个功能的方方面面，封装成一个整体，即类；
* 类的封装性，是通过访问权限修饰符来实现的；
* 提示：在项目中，属性基本都是私有的。通过公有的方法，对私有的属性进行赋值和取值。

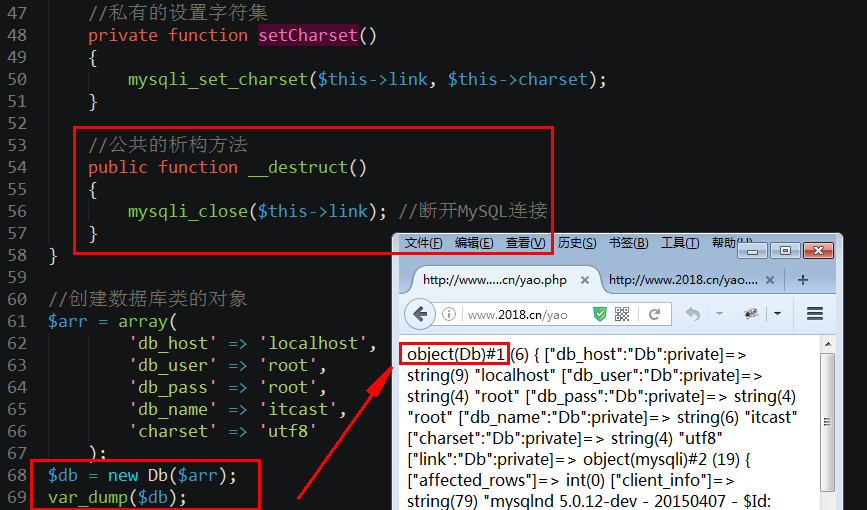
## 2、访问权限修饰符介绍

* public(公共的权限)：在任何地方都可以被访问，主要是类内、类外、子类中都可以被访问。
* private(私有的权限)：只能在本类中被访问，类外和子类中无权访问。
* protected(受保护的权限)：只能在本类中和子类中被访问，在类外不能访问。

## 3、实例：封装一个简单的数据库工具类





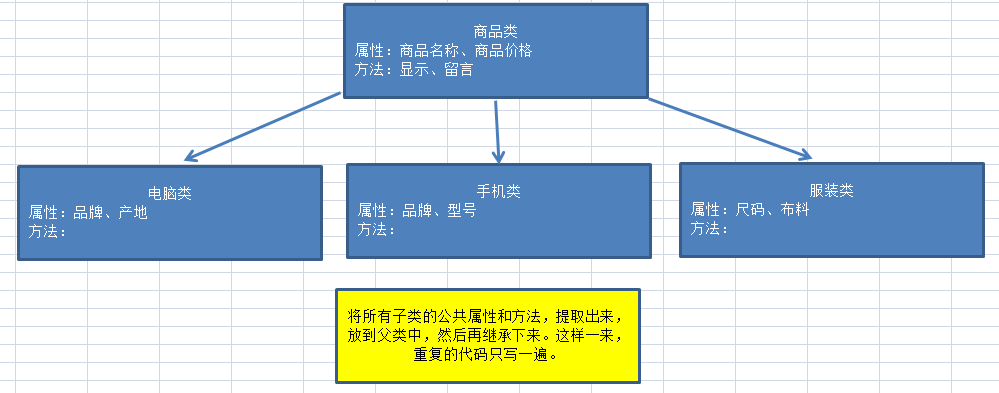


# 类的继承性

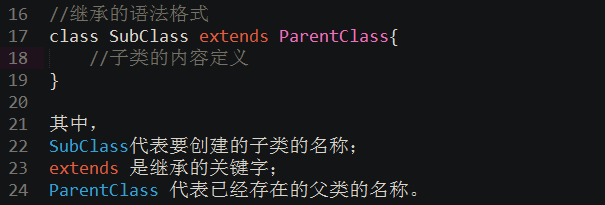
## 1、什么是类的继承性？

* CSS样式继承：父标签定义的样式，可以在子标签中直接使用。相同的样式只写一遍，减少代码量。
* 如果一个B类拥有了A类的所有特征信息，我们就说B类继承了A类。
* A类可以称为：基础类(最顶层的类)、父类、上层类。
* B类可以称为：最终类(最终的类)、子类、下层类。
* 提示：如果多个子类拥有相同的属性和方法，可以将相同的属性和方法提取出来，放到“父类”中，然后，再创建子类并继承父类；这样一样，重复的代码只写一遍，减少代码量，提高工作效率。
* 为什么要使用类的继承？是为了实现功能的升级和扩展。
* 功能的升级：原来有的功能，在子类进行完善。
* 功能的扩展：原来没有的功能，在子类增加新功能。



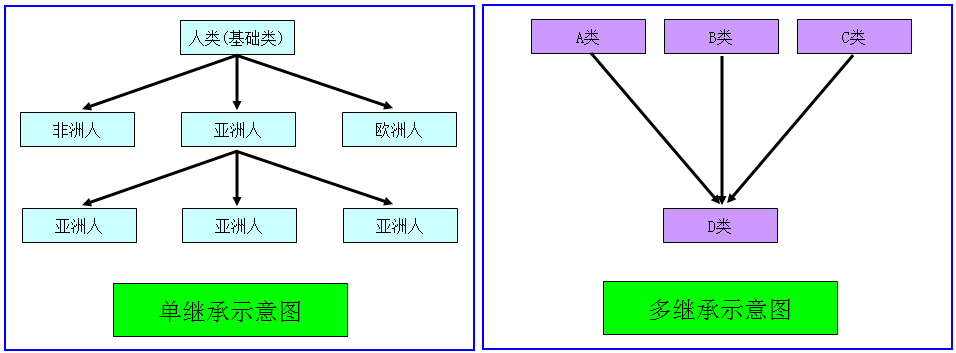


## 2、继承的语法格式

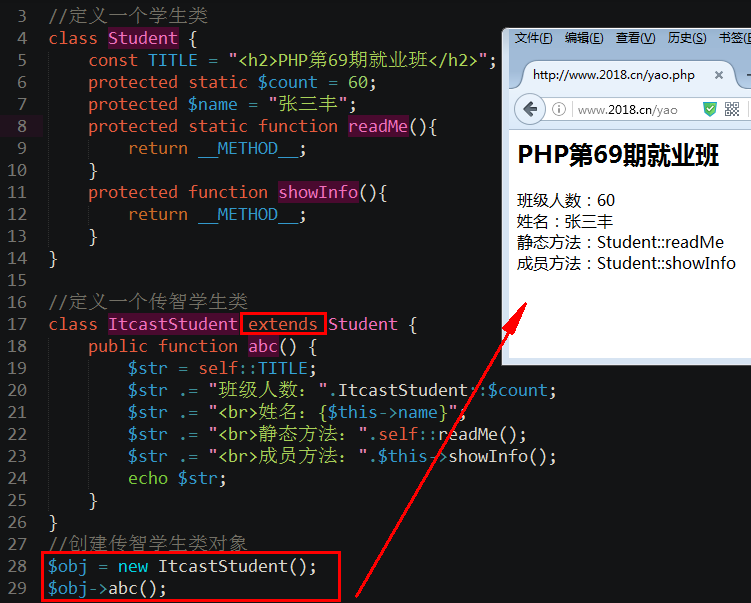


## 3、单继承和多继承

* 单继承：只能从一个父类来继承功能。如：Java、PHP、C#等。
* 多继承：同时从多个父类来继承功能。如：C++

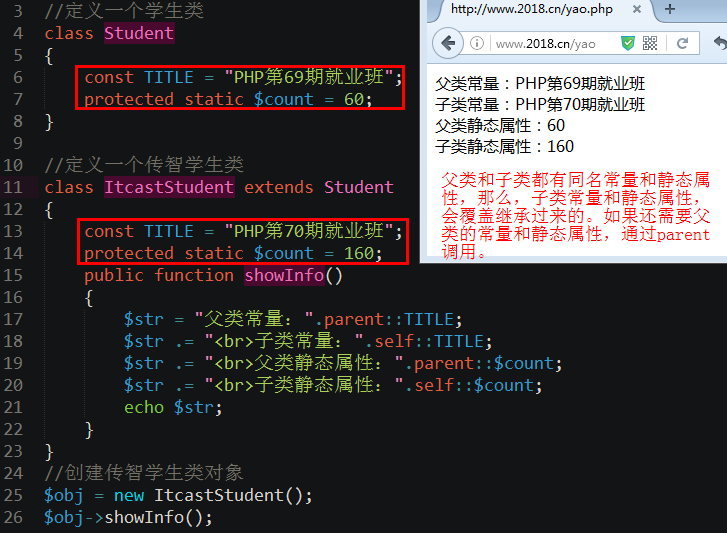


## 4、实例：单继承的简单演示



## 5、parent关键字

* self代表当前类，parent代表父类。
* self和parent可以存在于所有方法(成员方法和静态方法)中。
* self用来调用本类的内容，包括：类常量、静态属性、静态方法、成员方法；
* parent用来调用父类的内容，包括：类常量、静态属性、静态方法、成员方法；
* self和parent都使用范围解析符"::"来调用其它内容。
* 语法：**parent::类常量 | 静态属性 | 静态方法 | 成员方法**



# 类的多态

## 1、什么是类的多态？

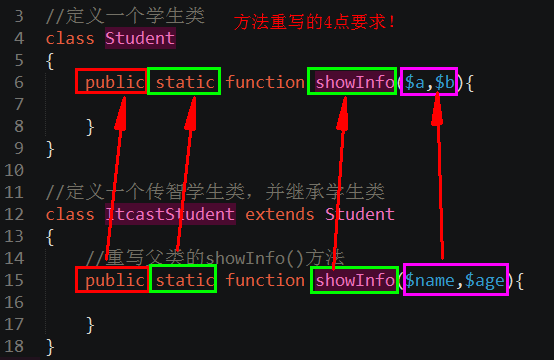
* 类的多态，就是类的多种形态。
* 类的多态主要指方法重载或方法重写。
* 方法重载：在同一个类中定义两个同名方法，PHP不支持。
* 方法重写：在子类中定义一个与父类同名的方法，就是“方法重写”。
* 为什么要重写方法？主要是实现功能的升级。父类中有的方法，子类中再用同样的名字再定义一次，一般来说，子类中方法的功能比父类更完善、更详尽。



## 2、方法重写的要求

* 子类中重写的方法，要与父类中的方法同名；
* 子类中重写的方法形参个数，要与父类中的同名方法形参个数一致；
* 子类中重写的方法类型，要与父类中同名方法类型一致；
* 子类中重写的方法的访问权限，不能低于父类中同名方法的访问权限；
  + 父类方法权限为public，子类同名方法权限只能是public；
  + 父类方法权限为protected，子类同名方法权限可以是protected和public；
  + 父类方法权限为private，子类无法继承，也无法重写。
* 注意：对于重写构造方法，就比较特殊，就没有形参个数的要求。

## 3、实例：方法重写的要求



## 4、实例：商品子类重写商品基础类中的方法



