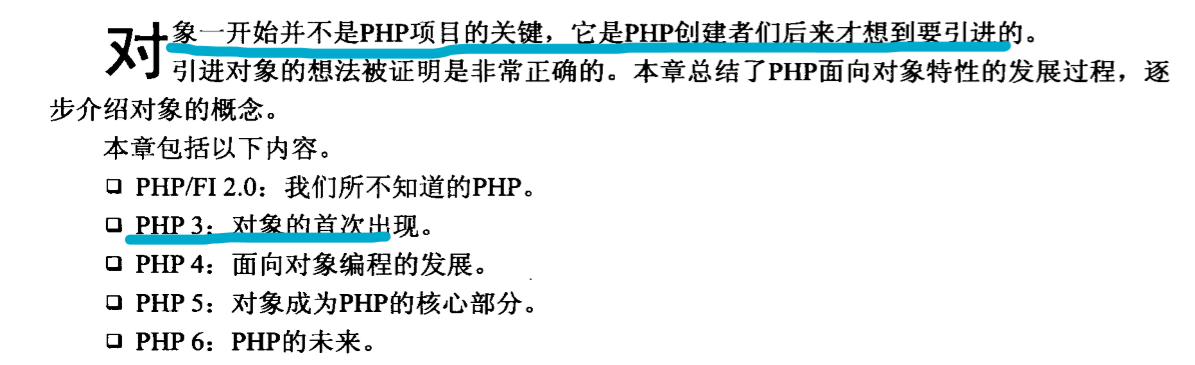
## 第二章.php与对象



### Php/FI

Php起源于有Raamus lerDorf用perl开发的两个工具 php FI

Php :personal homepage tool 意为个人主页工具，

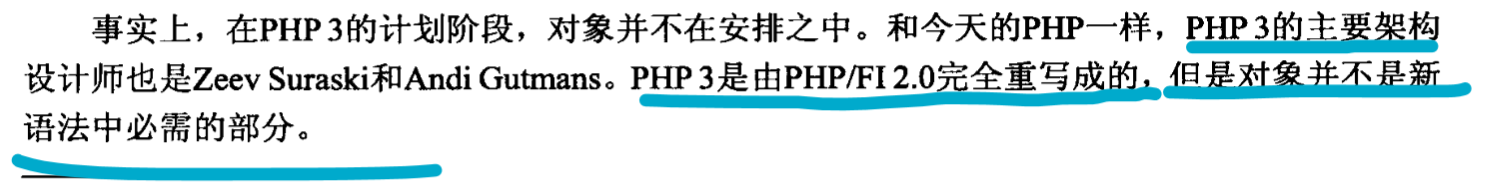
FI：Form Interpreter 表单解释器,



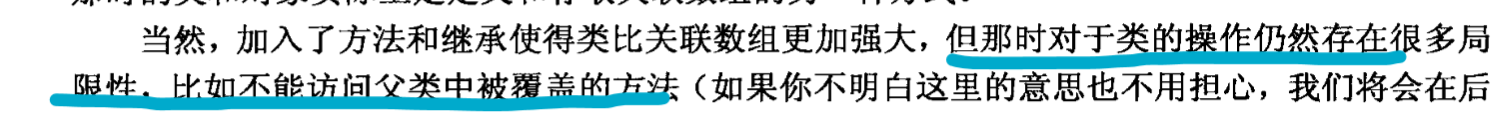
#### 功能与局限性



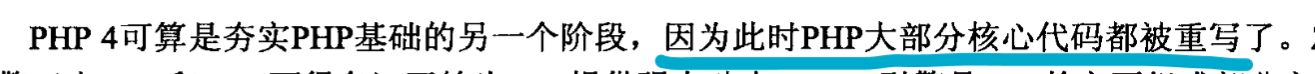
### Php3



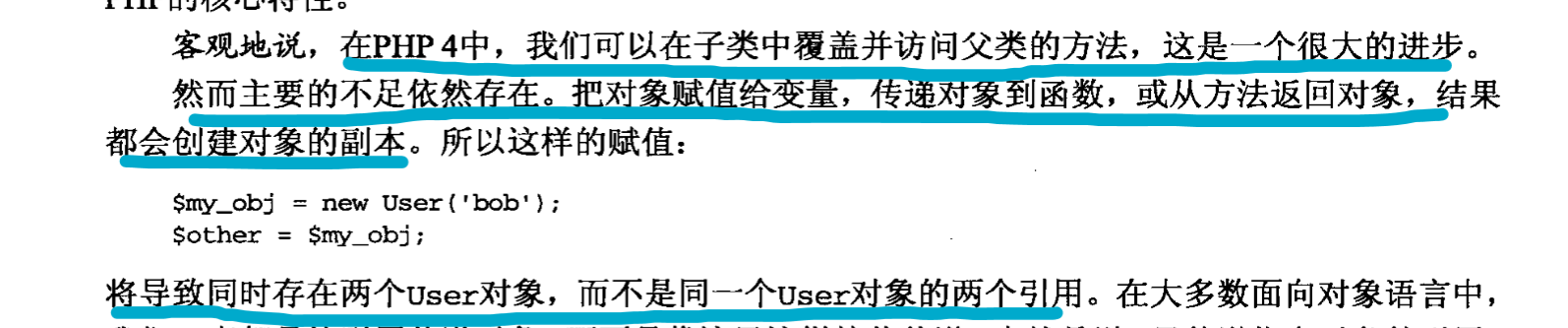
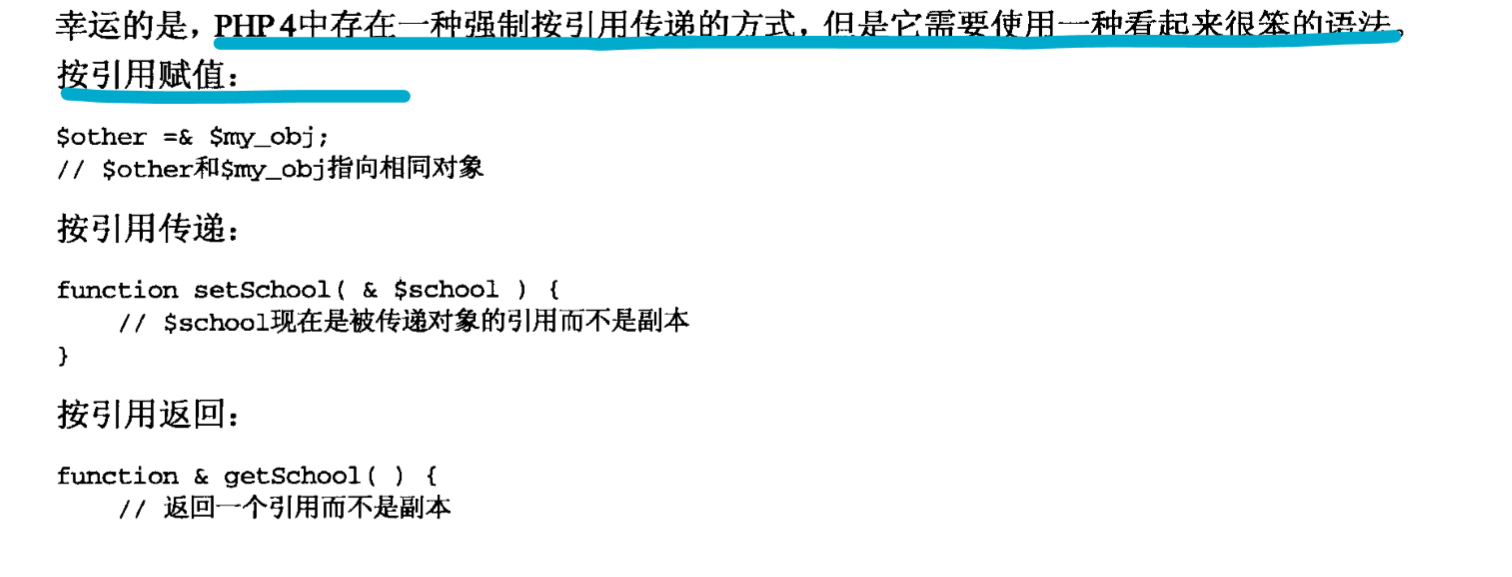
#### 功能和局限性

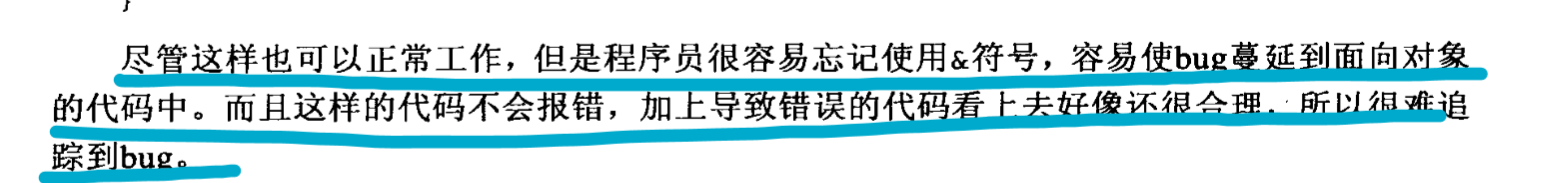


### Php4



#### 功能与局限性



### Php5



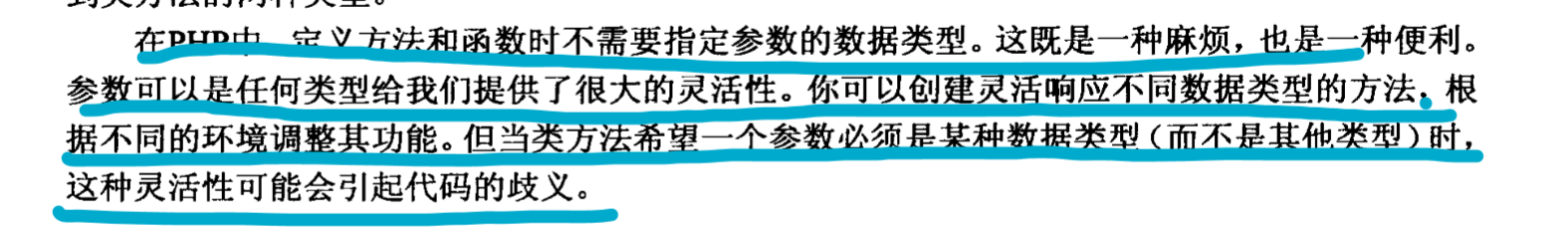
### PHP6

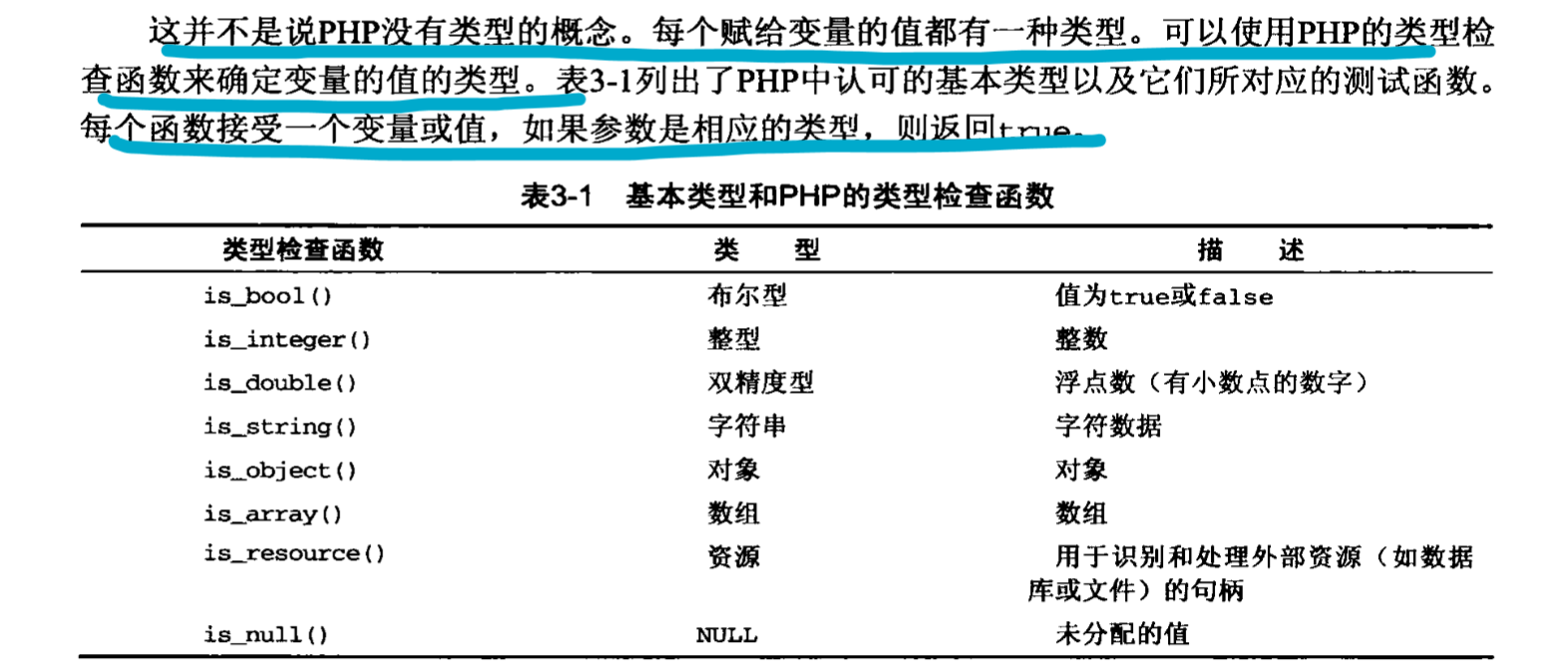
### Php7

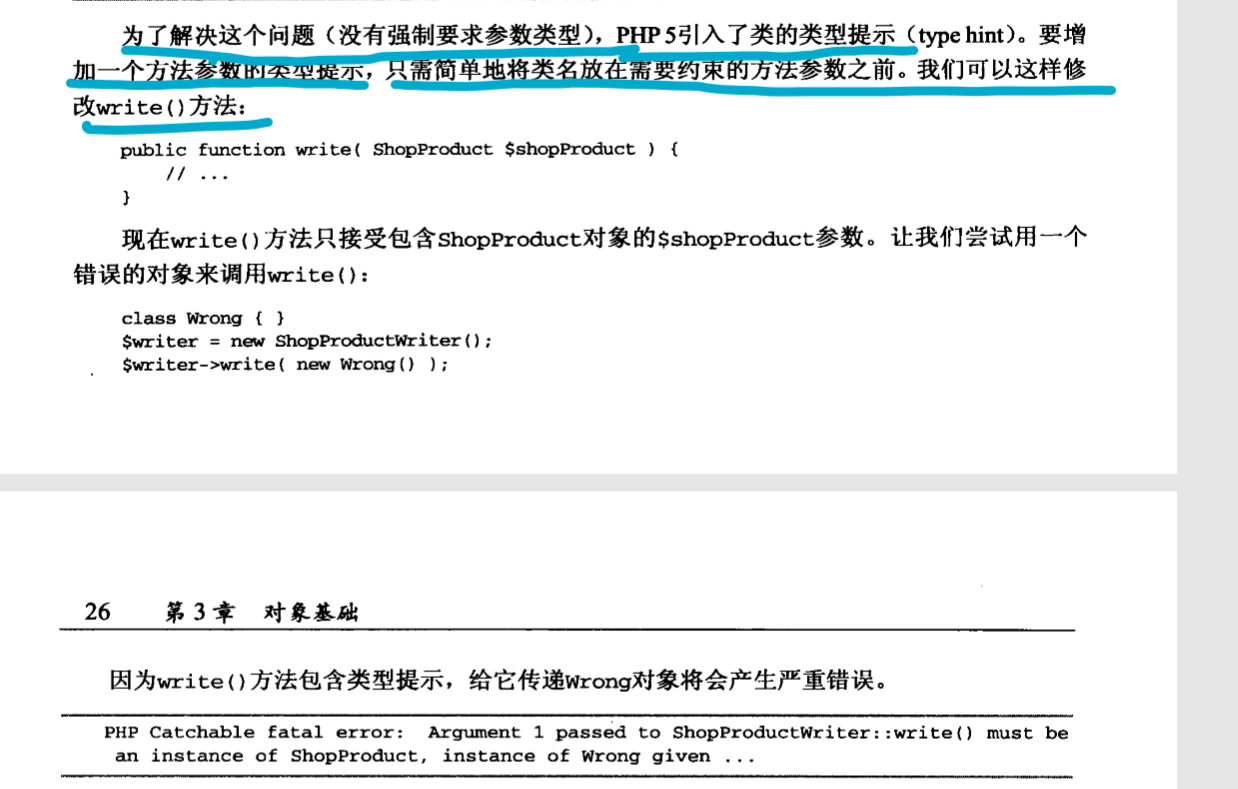
## 第三章．对象基础

其他的略

### 关于php的弱类型



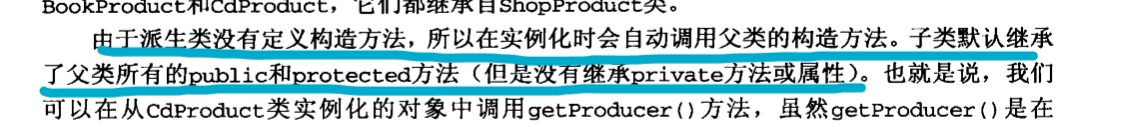




### 继承

#### 什么是继承

子类继承父类后拥有父类的所有public 和protected属性和方法，但是父类中的private属性和方法只能拥有无法访问即子类是无法直接输出父类的private属性



#### 子类继承父类的执行顺序

当实例化一个子类对象时，执行顺序为

父类静态变量、父类静态代码块（从上到下的顺序加载）

子类静态变量、子类静态代码块（从上到下的顺序加载）

父类的非静态变量、父类的非静态代码块（从上到下的顺序加载）

父类的构造方法

子类的非静态变量、子类的非静态代码块（从上到下的顺序加载）

子类的构造方法

#### Public proected private类的访问

public ,private ,static

public变量可以通过$test->new test(); $test->name=”123”;修改变量 也可直接调用其公有方法。private变量和方法则不可以。

Static 变量和方法 类内可以通过self::$count=0;self::test();调用—static方法里只能使用static变量

类外通过Myclass:: $count=0;Myclass::test()；调用

访问父类静态属性使用parent，如：parent::$name

Static private 变量和方法类外不能Myclass:: $count=0;调用

#### [PHP中self和this的用法区别](https://www.cnblogs.com/kenshinobiy/p/9557294.html)

一个类的所有实例都共享相同的静态变量，在一个类的方法中只能通过self::访问静态变量和静态方法，访问其他变量和方法用this。

class exampleClass

{

    public static $foo;

    public $bar;

   public function regularFunction() { echo $this->bar; }

    public static function staticFunction() { echo self::$foo; }

    public static function anotherStatFn() { self::staticFunction(); }

public function regularFnUsingStaticVar() {

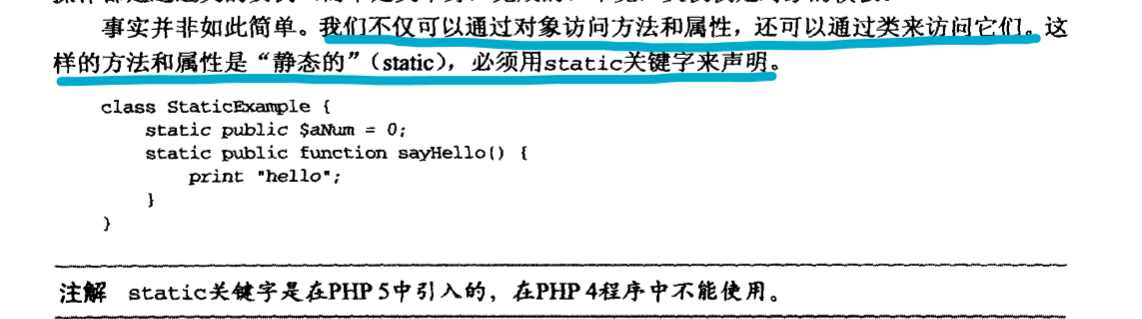
echo self::$foo; }

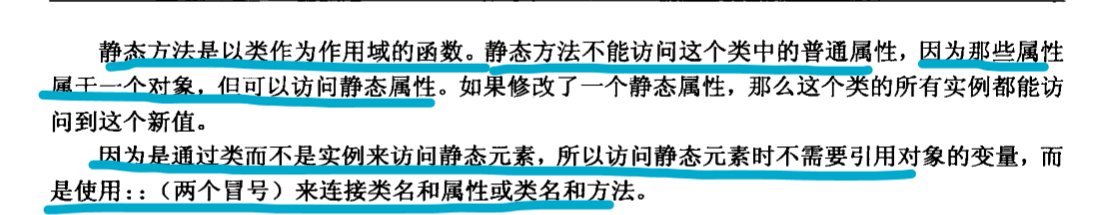
    // 注: PHP 5.3 使用 $this::$bar 代替 self::$bar 是允许的

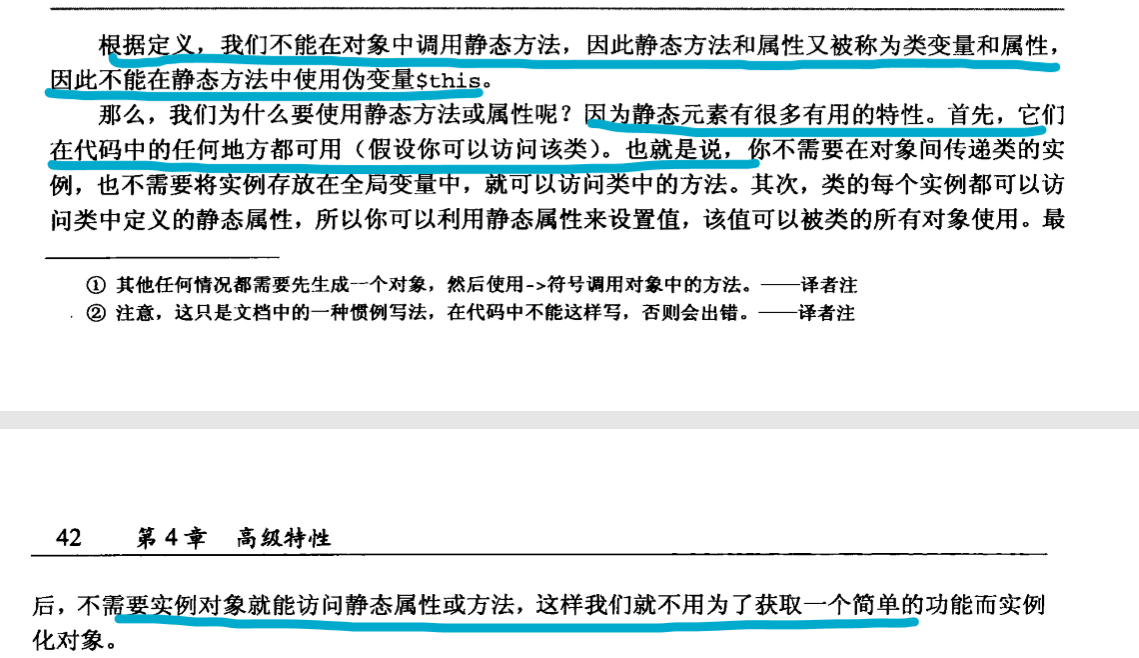
}

## 第四章．高级特性

### 4.1静态方法与属性







#### 内存区域的划分与分配

c/c++

**栈区**：程序运行时由编译器自动分配，存放函数的参数值，局部变量等，程序结束时由编译器自动释放

**堆区**：在内存开辟另一块存储区域。一般由程序员分配释放， 若程序员不释放，程序结束时可能由OS回收 。注意它与数据结构中的堆是两回事，分配方式倒是类似于链表，呵呵。用malloc, calloc, realloc等分配内存的函数分配得到的就是在堆上

**全局区(静态区)**：编译器编译时即分配内存。全局变量和静态变量的存储是放在一块的。对于C语言初始化的全局变量和静态变量在一块区域， 未初始化的全局变量和未初始化的静态变量在相邻的另一块区域。而C++则没有这个区别 - 程序结束后由系统释放

**文字常量区**：常量字符串就是放在这里的。 程序结束后由系统释放

**程序代码区**：存放函数体的二进制代码。

#### 静态变量与全局变量的区别

class test{

static private $a=2;

function demo(){

self::$a++;

echo self::$a;

echo "<br>";

}

}

$test=new test();

$test->demo();//3

$test->demo();//4

$test2=new test();

$test2->demo();//5

$test2->demo();//6

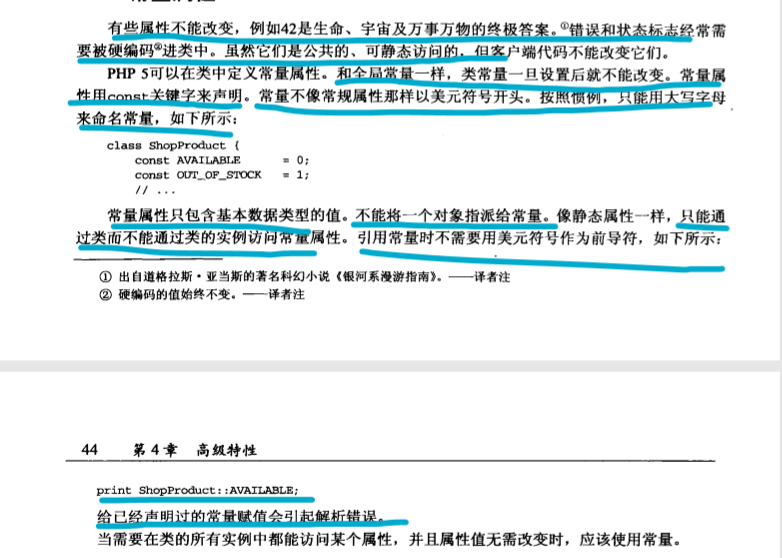
$test3=new test();

$test3->demo();//7

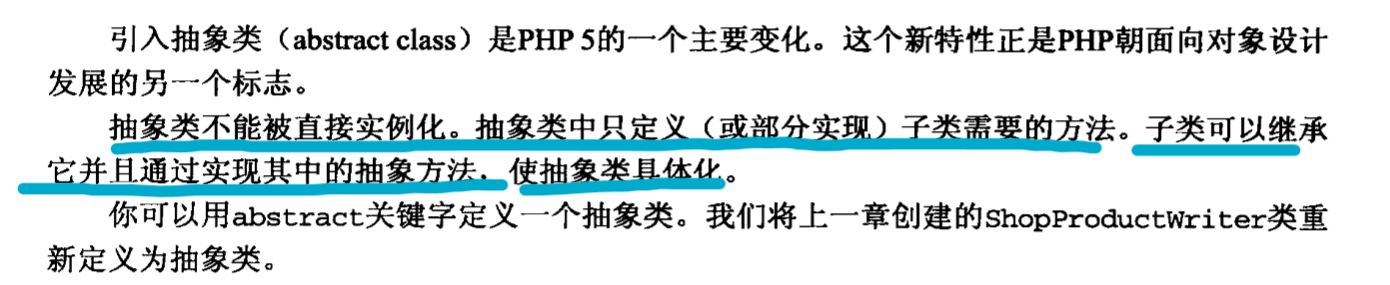
$test3->demo();//8

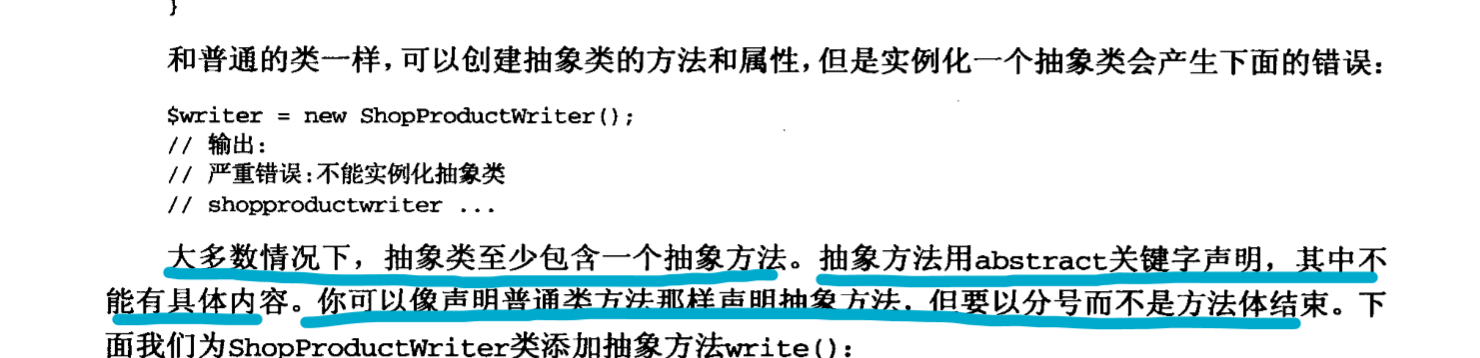
**静态全局变量只在定义它的文件内有效**；**静态局部变量只在定义它的函数内有效，只是程序仅分配一次内存，函数返回后，该变量不会消失**

### 4.2常量属性

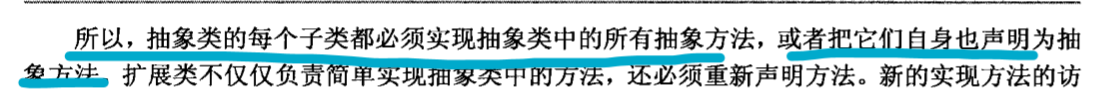


### 4.3抽象类





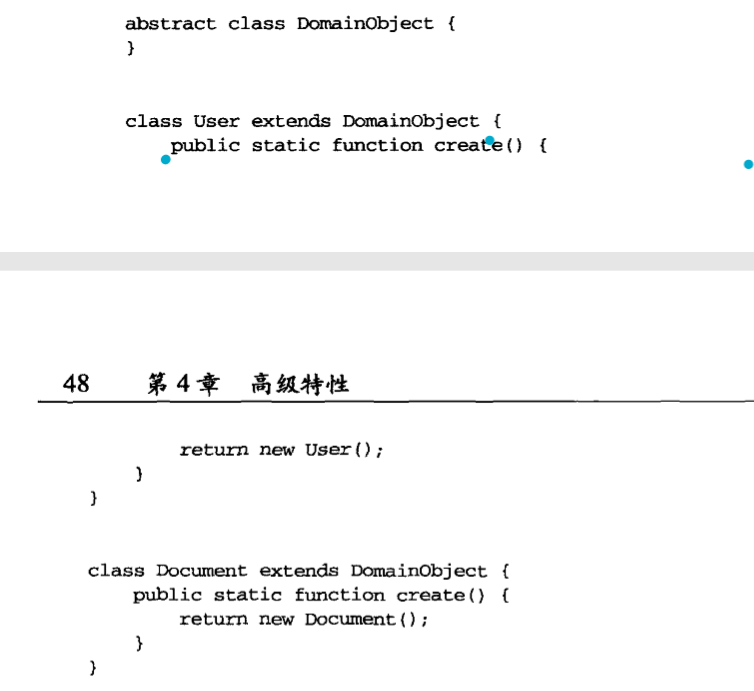


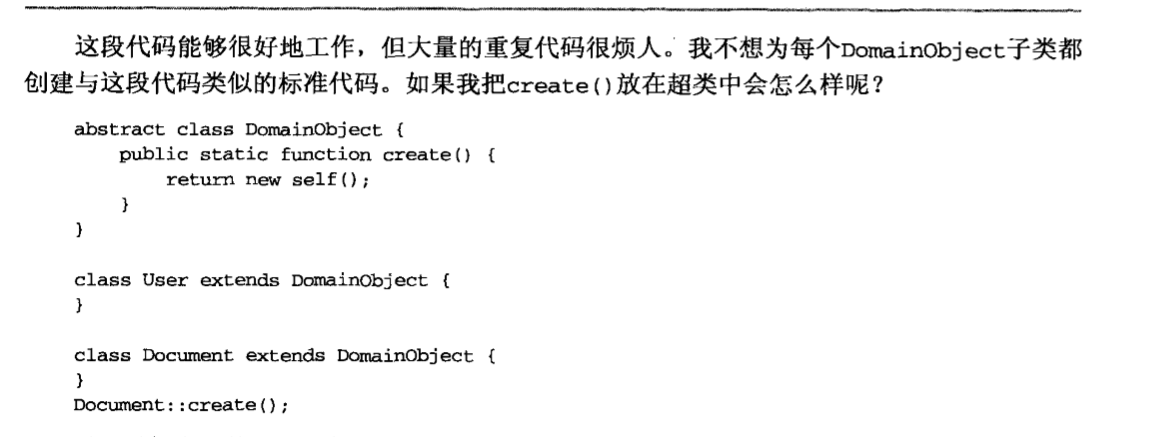


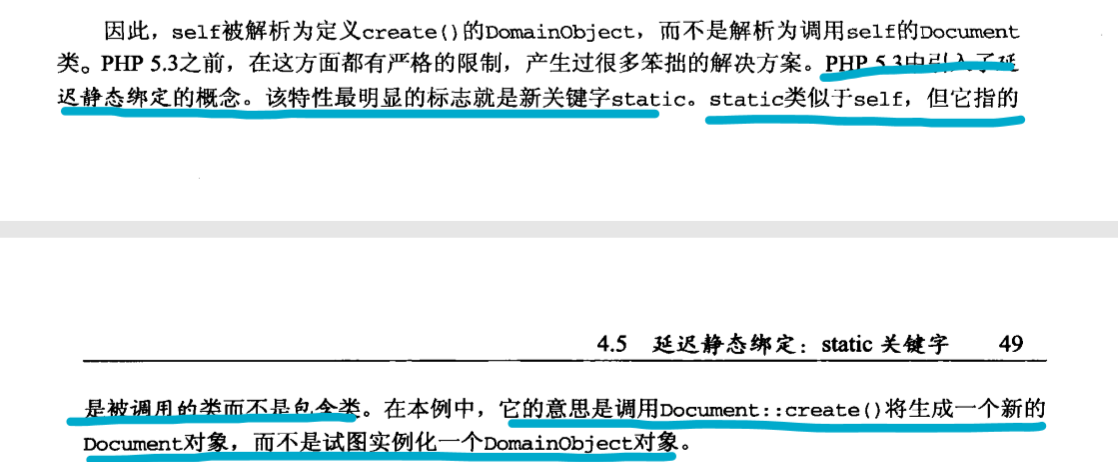
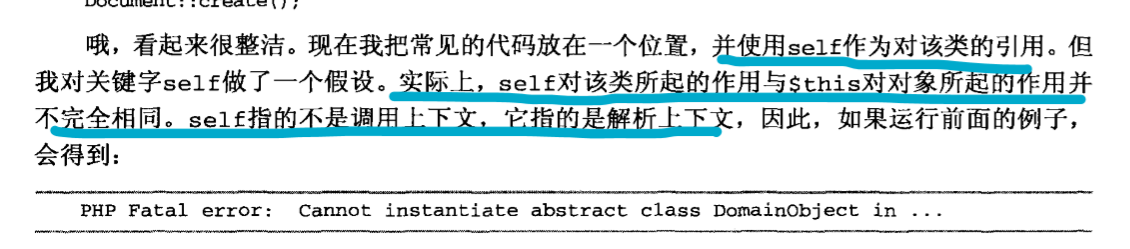
### 4.3接口

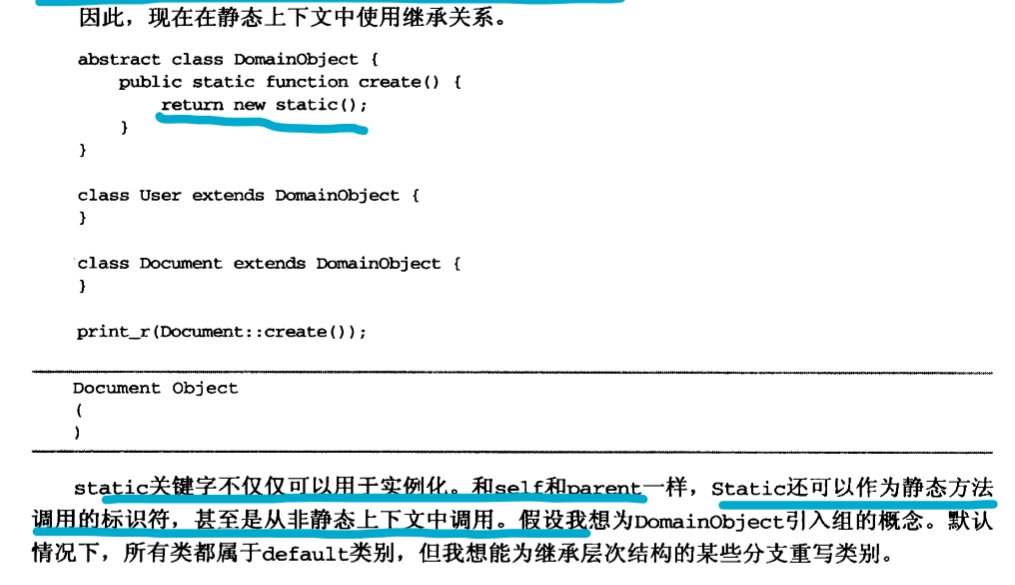
略

### 4.4延迟静态绑定：static关键字





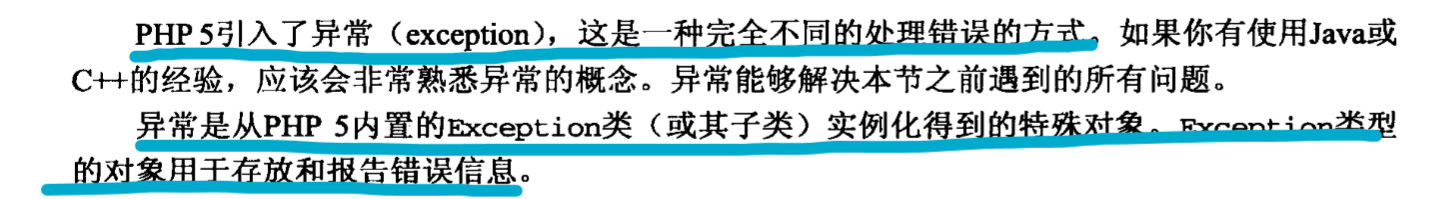


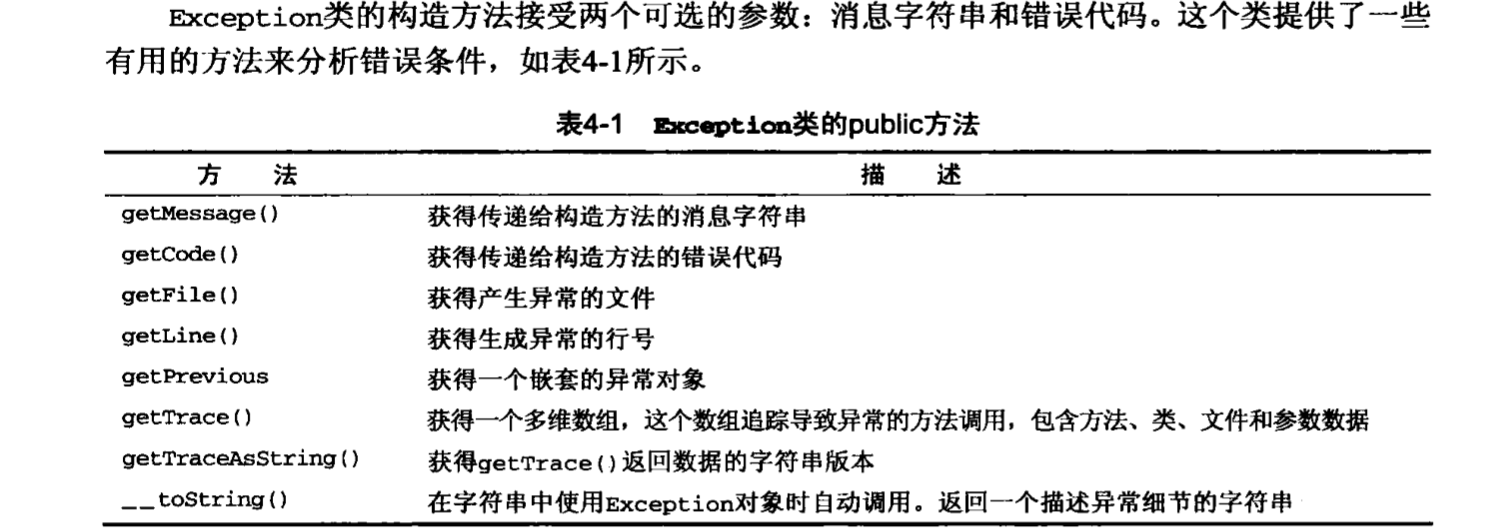


### 4.5错误处理

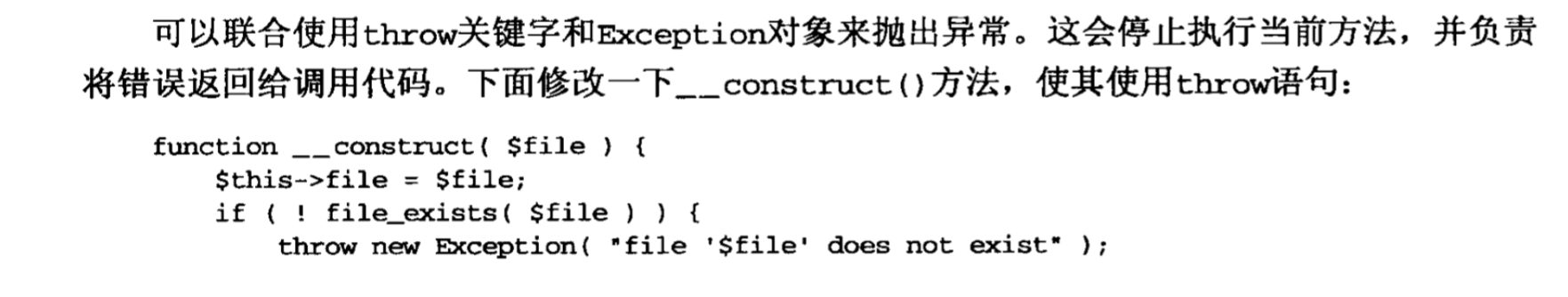
略

### 4.6异常





#### 抛异常

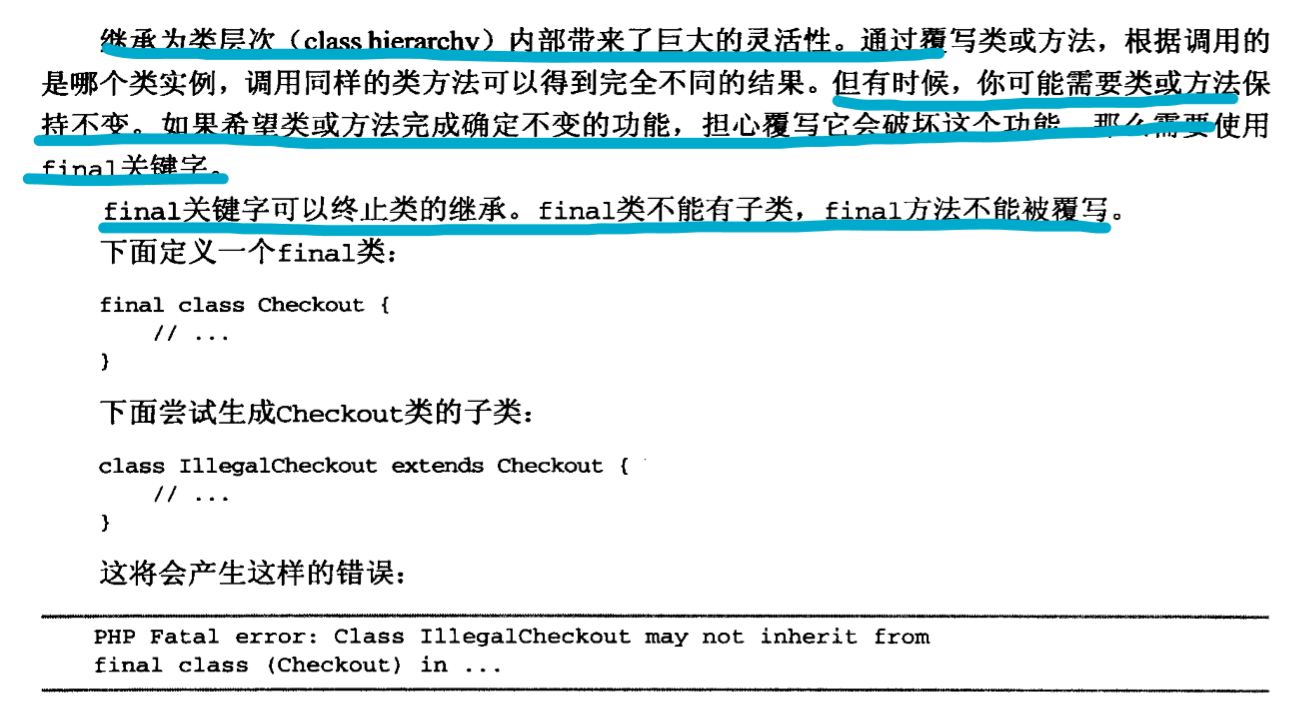


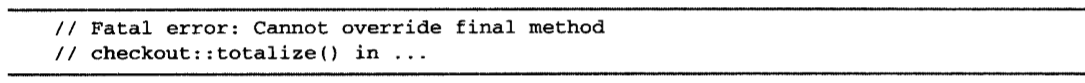
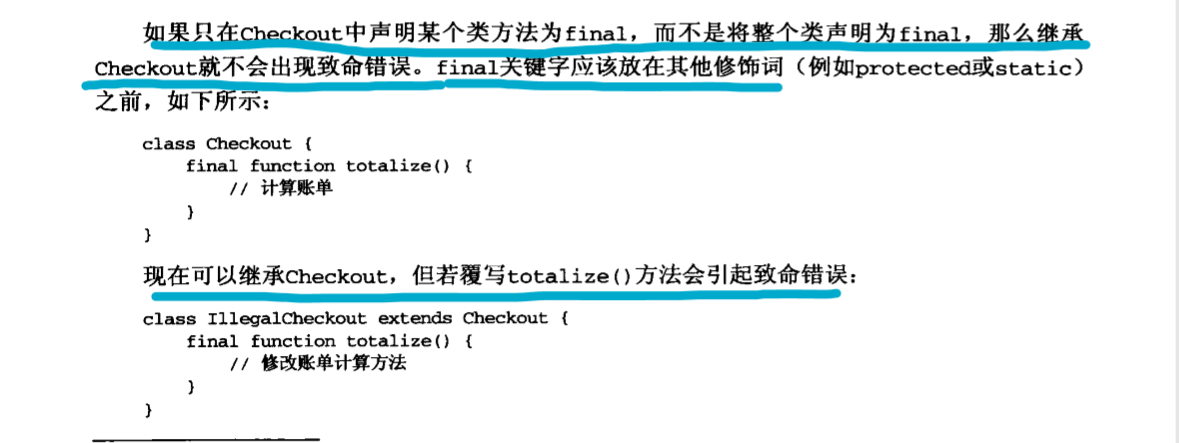
#### 异常的子类化

看一下这本书最新版的内容

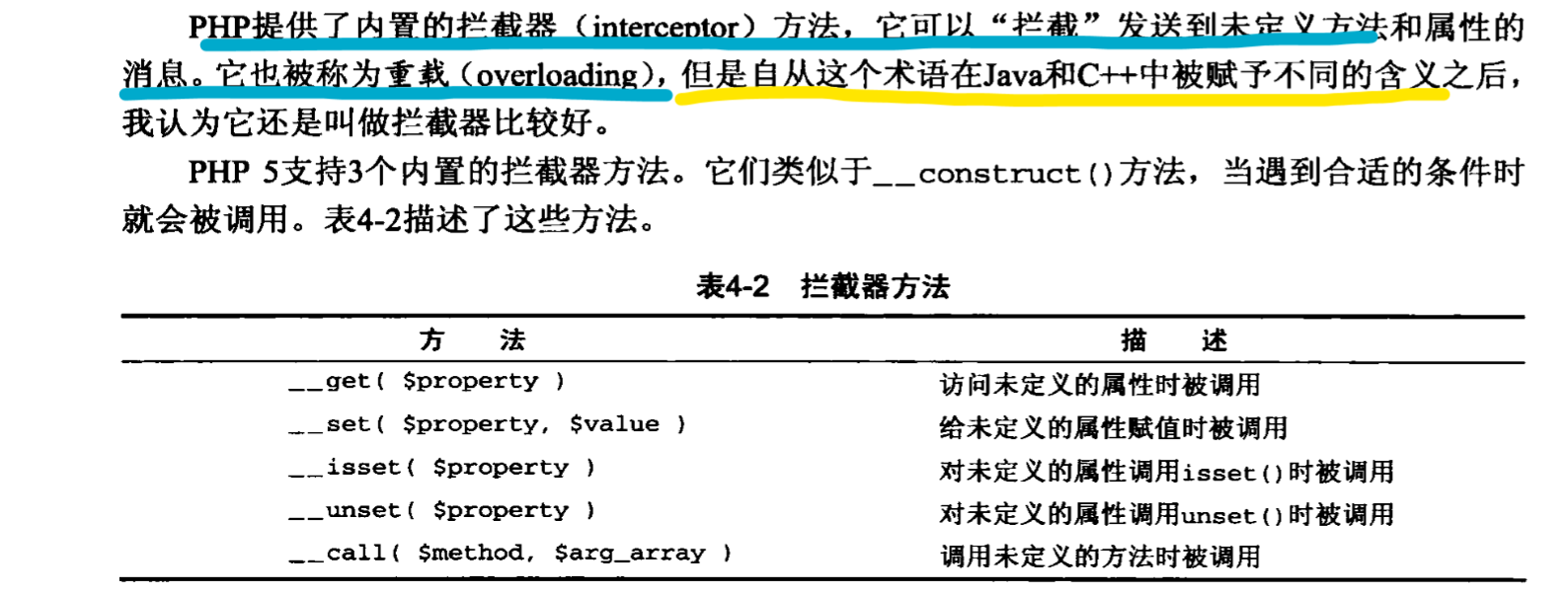
未完待续

### 4.7final类和方法



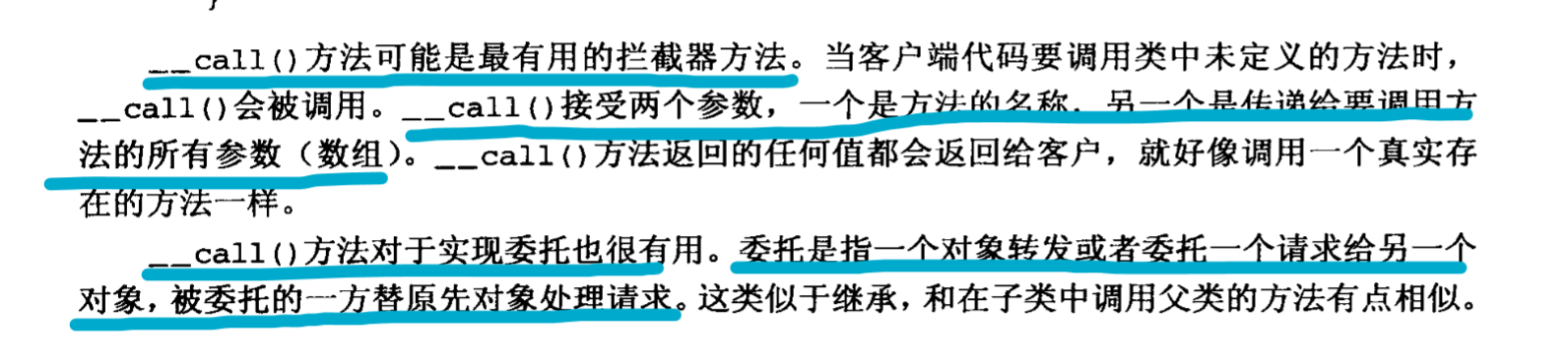


### 4.8拦截器



#### \_\_call()

调用未定义方法时调用



class PersonWriter{

function writeName(Person $p){

print $p->getName()."\n";

}

function writeAge(Person $p){

print $p->getAge()."\n";

}

}

class Person{

private $write;

function \_\_construct(PersonWriter $write){

$this->write=$write;

}

function \_\_call($methodName,$args){

if(method\_exists($this->write, $methodName)){

return $this->write->$methodName($this);

}

}

function getName(){

return "名字：李白";

}

function getAge(){

return "年龄:18";

}

}

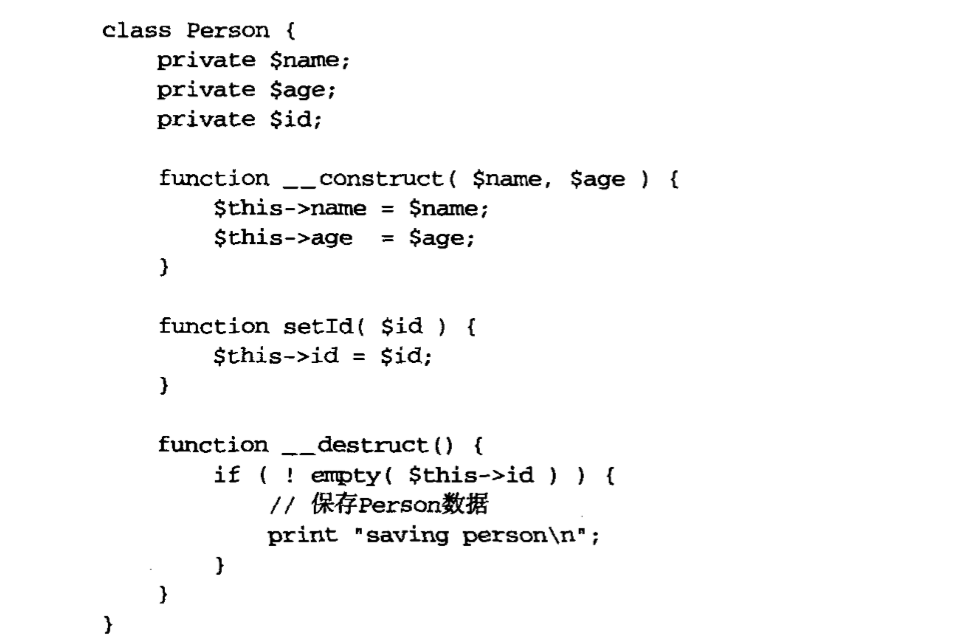
$person=new Person(new PersonWriter());

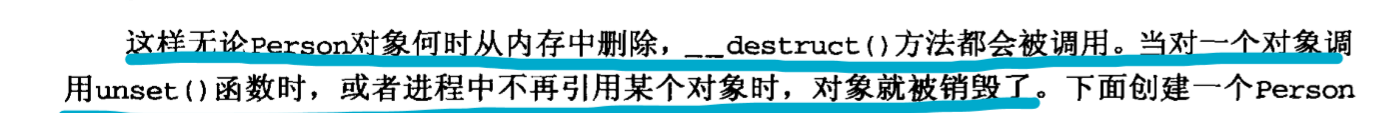
$person->writeName();

$person->writeAge();

### 4.9析构函数

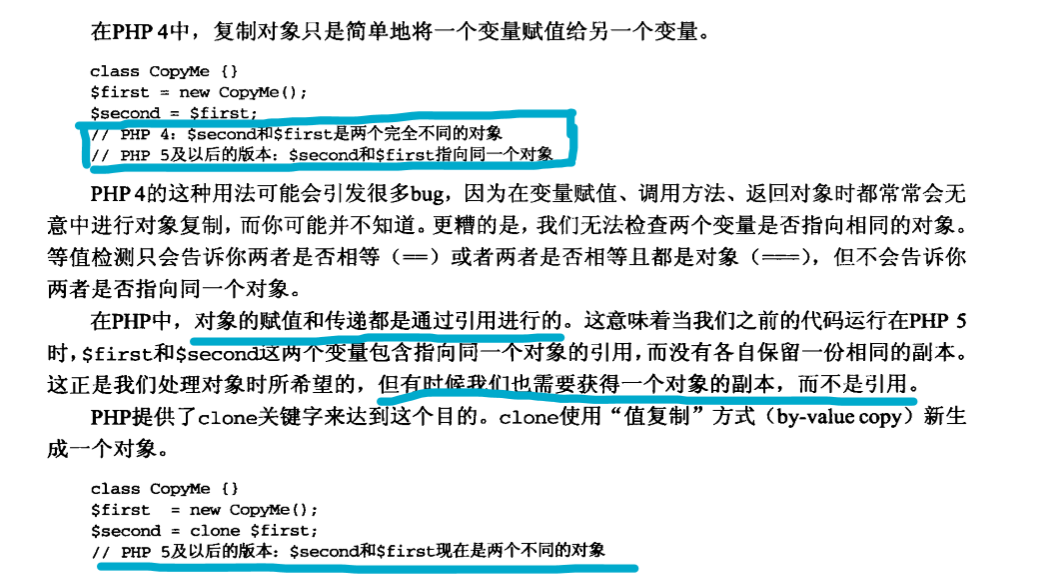
\_\_destruct()

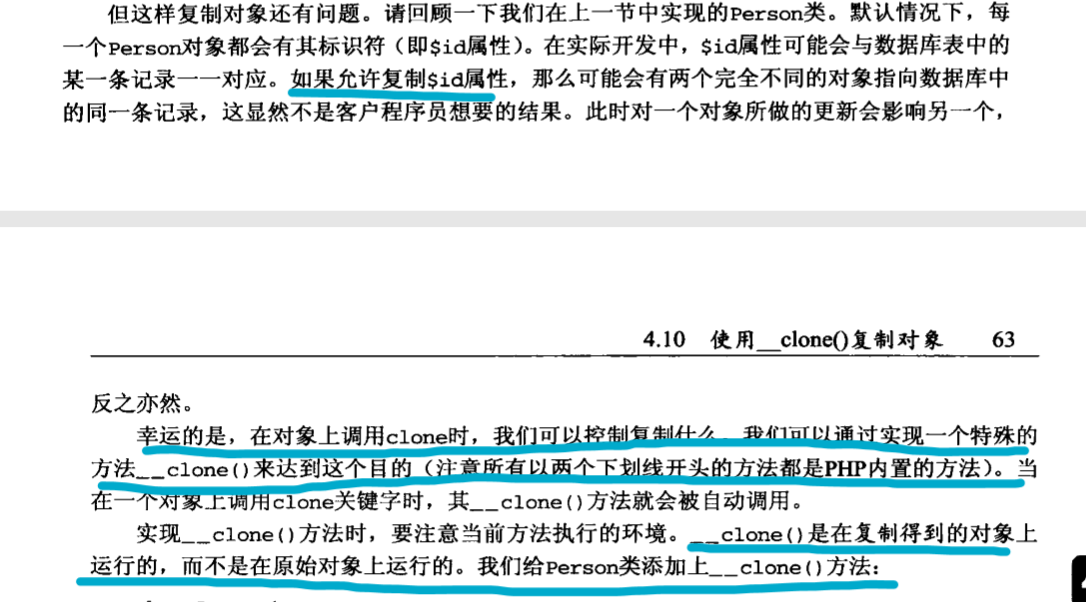


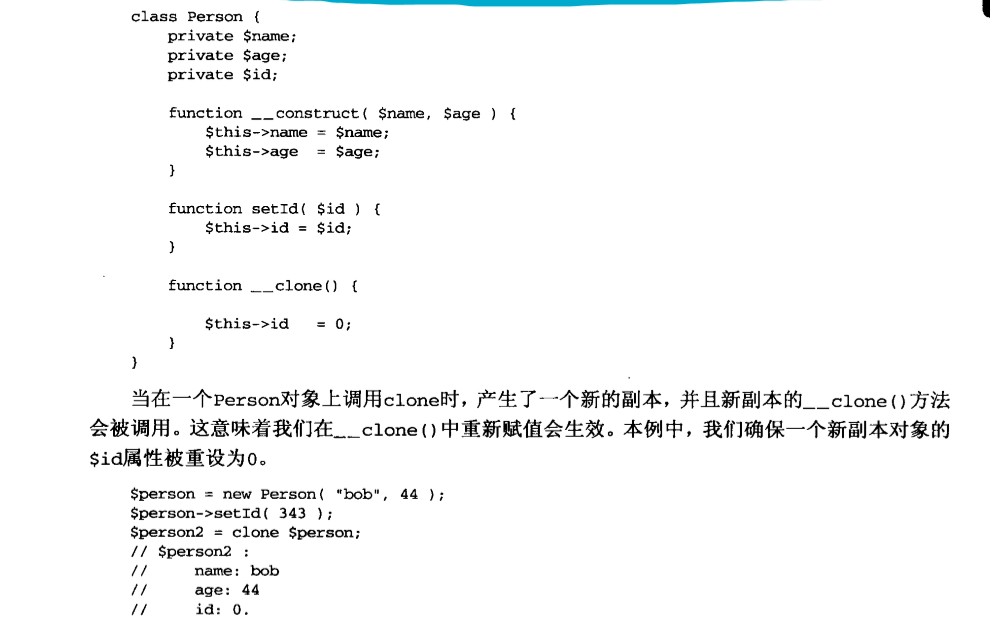


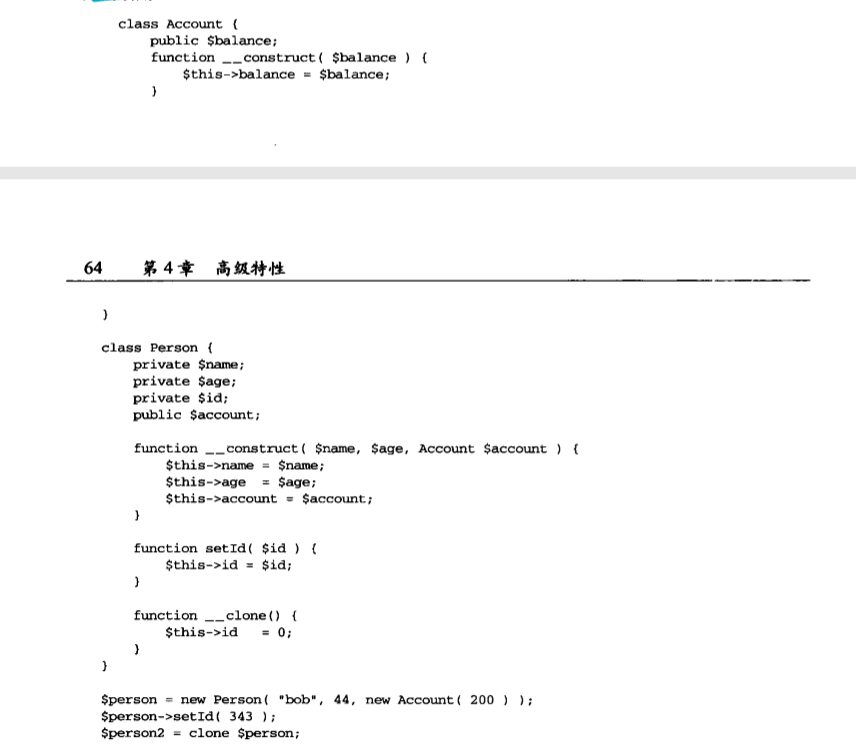


### 4.10\_\_clone()复制对象

6









### 4.11定义对象的字符串值

\_\_toString()print echo当打印对象时，自动调用\_\_toString(),并用它的返回值来代替默认输出的值

把对象解析成一个字符输出

class person{

function getName(){

return "名字：李白";

}

function getAge(){

return "年龄:18";

}

function \_\_toString(){

$des=$this->getName();

$des.=$this->getAge();

return $des;

}

}

$person=new Person();

print $person;

### 4.12回调、匿名函数和闭包

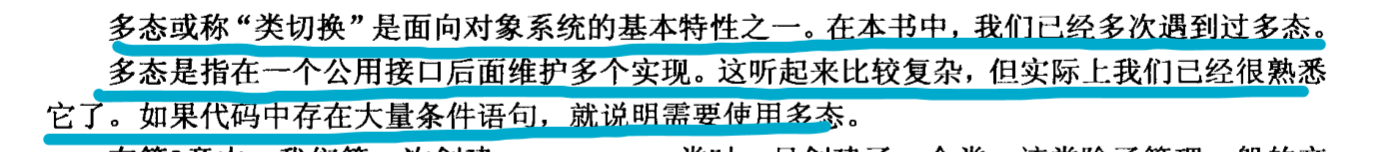
## 第五章．对象工具

包、命名空间、反射API

### 反射API

## 第六章．对象与设计

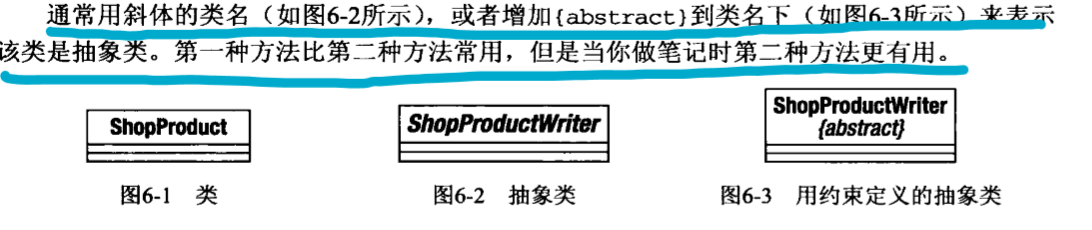
多态

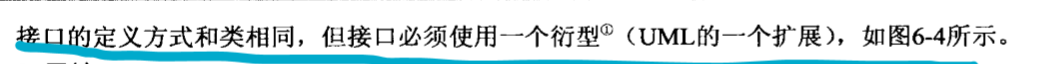


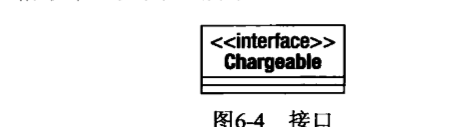
### UML

#### 类图

##### 描述类



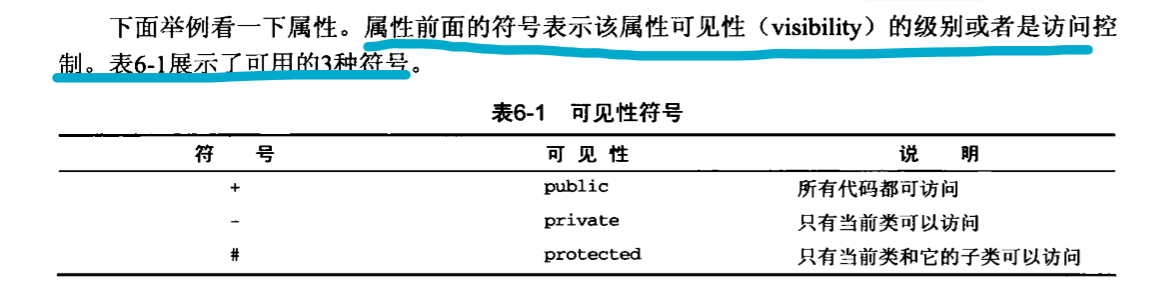




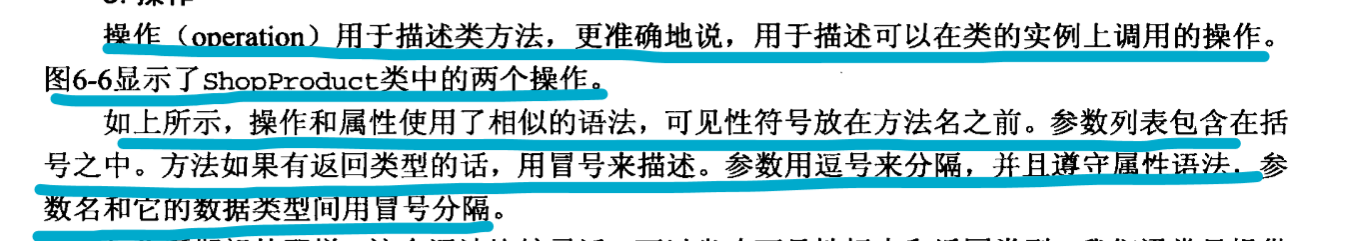
##### 属性

属性直接列在类名下面的格子中，如图6-5



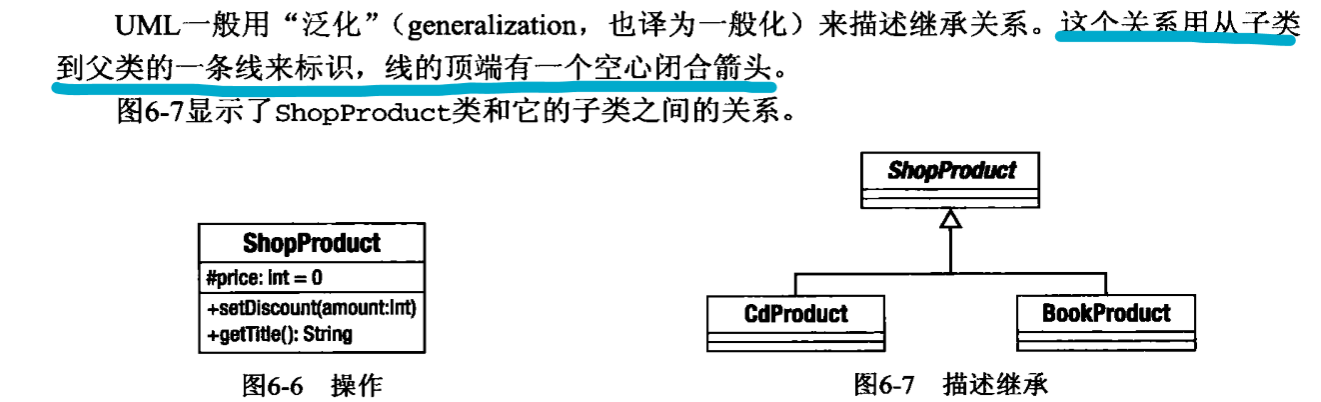


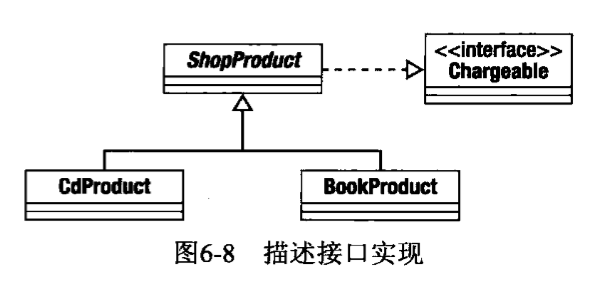
##### 操作



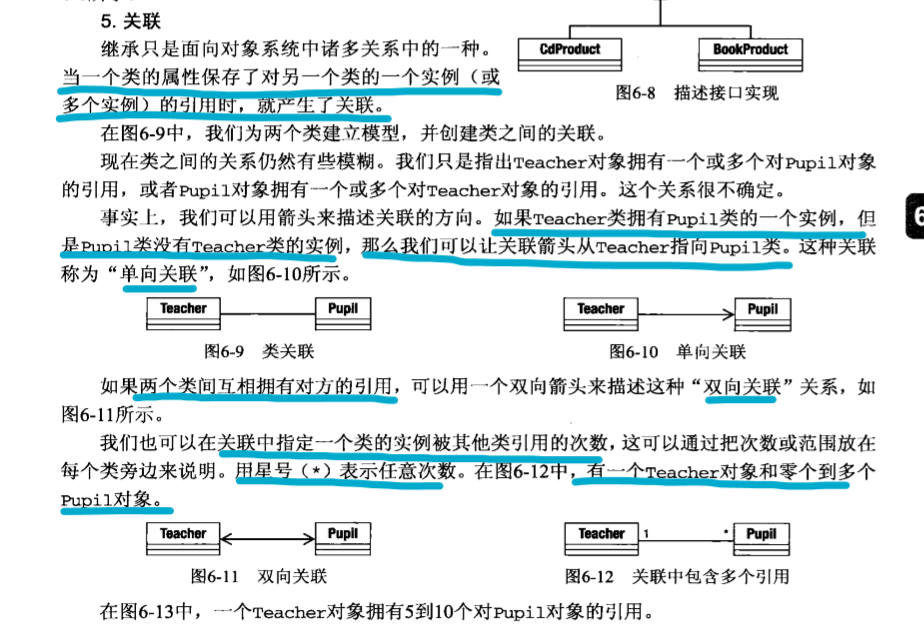


##### 描述继承和实现

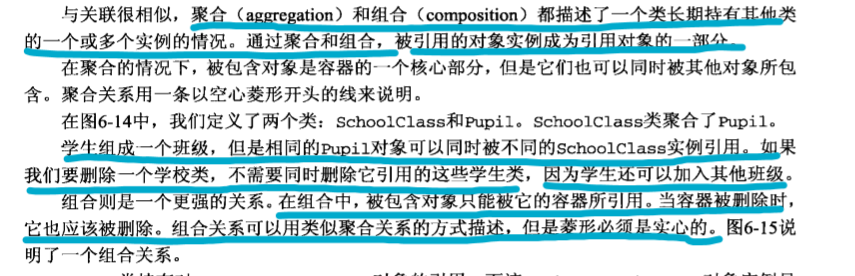


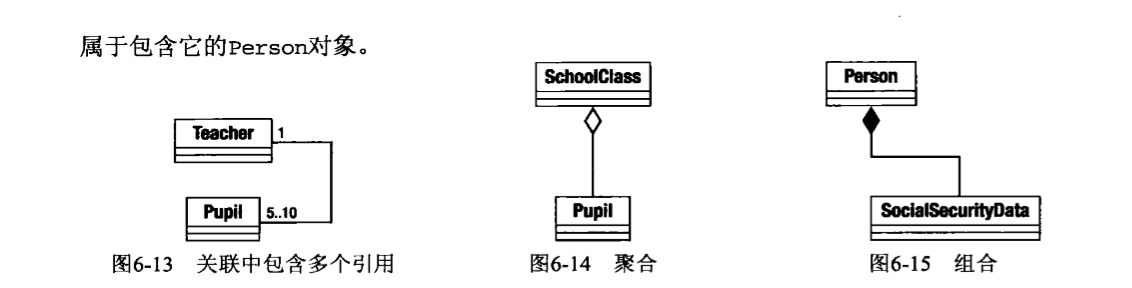


##### 关联

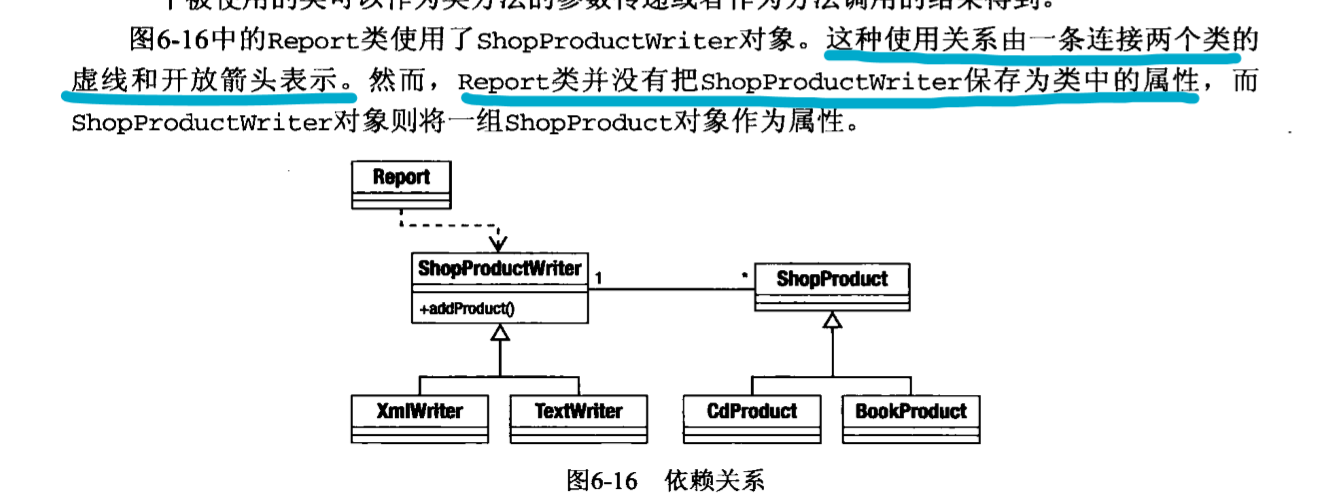


##### 聚合和组合





##### 7.描述使用



##### 8.使用注解

