

## 面向对象编程——单元 4：抽象 abstract 和终极 final

### 一、抽象类与抽象方法

**关键字：**abstract 代表抽象

使用 abstract 关键字修饰类，表示抽象类。抽象类不能实例化对象。

**语法：**abstract class 类名

使用 abstract 关键字修饰方法。没有方法具体实现只有方法的声明，这样的方法就是抽象方法。

**语法：**abstract function 方法名(参数列表)

包含了抽象方法的类一定是抽象类；抽象类不一定非要包含抽象方法；抽象类也可以包含非抽象的方法。

抽象类主要用途就是被继承。继承了抽象类的子类需要实现所有的抽象方法，如果没有全部实现则该子类也必须是抽象类。

```
1  <?php
2      /**
3       * 抽象类 商品类，不能直接实例化一个商品类对象。
4       */
5      abstract class Goods{
6          protected $name;
7          protected $price;
8          protected $numbers;
9          public function __construct($name,$price,$numbers){
10              $this->name = $name;
11              $this->price = $price;
12              $this->numbers = $numbers;
13          }
14          public function getPrice(){
15              return $this->price;
16          }
17          //声明了2个抽象方法，没有具体方法的定义
18          public abstract function discount($discount); //折扣
19          public abstract function introduce(); //介绍商品
20      }
21      /**
22       * 图书类，继承抽象的商品类
23       */
24      class Book extends Goods{
25          private $publisher; //出版社
26          private $author; //作者
27          public function __construct($publisher,$author,$name,$price,$numbers){
28              parent::__construct($name,$price,$numbers);
29              $this->publisher = $publisher;
30              $this->author = $author;
31          }
32          public function discount($discount){
33              //图书折扣允许3折及以上
34              if ($discount>=0.3) {
35                  $this->price = round($this->price*$discount,2) ;
36              } else {
37                  trigger_error("图书商品折扣必须大于等于3折",E_USER_NOTICE);
38              }
39          }
40          public function introduce(){
41              echo "[图书信息: ]<br>";
42              echo "书名: {$this->name}<br>";
43              echo "出版社: {$this->publisher}<br>";
44              echo "作者: {$this->author}<br>";
45              echo "单价: {$this->price}<br>";
46              echo "库存: {$this->numbers}<br>";
47          }
48      }
```

```
49  /**
50  * 手机类，继承抽象的商品类
51  */
52  class Phone extends Goods{
53      private $model; //型号
54      private $brand; //品牌
55      public function __construct($model,$brand,$name,$price,$numbers){
56          parent::__construct($name,$price,$numbers);
57          $this->model = $model;
58          $this->brand = $brand;
59      }
60      public function discount($discount){
61          //手机折扣允许7折及以上
62          if ($discount>=0.7) {
63              $this->price = round($this->price*$discount,2) ;
64
65          } else {
66              trigger_error("手机商品折扣必须大于等于7折",E_USER_NOTICE);
67          }
68      }
69      public function introduce(){
70          echo "[手机信息: ]<br>";
71          echo "名称: {$this->name}<br>";
72          echo "型号: {$this->model}<br>";
73          echo "品牌: {$this->brand}<br>";
74
75          echo "售价: {$this->price}<br>";
76          echo "库存: {$this->numbers}<br>";
77      }
78
79      // $goods1 = new Goods(); //报错
80      $book1 = new Book('电子工业出版社','张三','PHP教程',30.00,50);
81      $book1->discount(0.9);
82      echo $book1->getPrice();
83      echo "<br>";
84      $book1->discount(0.2);//发生用户级提醒，不能打2折
85      echo $book1->getPrice();
86      echo "<br>";
87      $book1->introduce();
88
89      echo "<hr>";
90
91      $phone1 = new Phone('mate 10','华为','HUAWEI Mate 10 4GB+64GB 全网通版',3899.00,10)
92      ;
93      echo $phone1->getPrice();
94      echo "<br>";
95      $phone1->discount(0.85);
96      echo $phone1->getPrice();
97      echo "<br>";
98
99      $phone1->introduce();
100  ?>
```

运行结果:

(!)

Notice: 图书商品折扣必须大于等于3折 in F:\wamp\www\objectDemo\jicheng\goods.class.php on line 37

Call Stack			
#	Time	Memory	Function
1	0.0010	162880	{main} ( )
2	0.0010	163760	Book->discount ( )
3	0.0010	163912	trigger_error ( )

27

[图书信息: ]

书名: PHP教程

出版社: 电子工业出版社

作者: 张三

单价: 27

库存: 50

3899

3314.15

[手机信息: ]

名称: HUAWEI Mate 10 4GB+64GB 全网通版

型号: mate 10

品牌: 华为

售价: 3314.15

库存: 10

为什么要使用抽象类？

在面向对象方法中，抽象类主要用来进行类型隐藏。构造出一个固定的一组行为的抽象描述，但是这组行为却能够有任意个可能的具体实现方式。这个抽象描述就是抽象类，而这一组任意个可能的具体实现则表现为所有可能的派生类。抽象类是不完整的，它只能用作父类。

抽象父类——定义了一组抽象的行为规范，但具体内容不确定。

继承了抽象父类的子类——必须实现这些行为规范，具体内容根据情况不同而实现方法不同。

设计练习 1：

在一个学校的教务管理系统中，有 3 种用户——管理员、教师、学生。三种用户必须先登录才能进入系统。管理员用户可以添加教师、添加学生等操作；教师用户可以给成绩；学生用户可以查看成绩。从

代码重用的角度考虑要实现这些用户类，需要几个类？如何设计？

设计参考：

```
1  <?php
2      //抽象的父类，用户
3      abstract class User{
4          protected $username;
5          protected $password;
6          protected $type;
7          public abstract function login($n,$p);
8      }
9      /**
10     * 管理员类
11     */
12     class Admin extends User{
13         private $authority; //管理员有权限
14         public function __construct(){
15             $this->type = '管理员';
16         }
17         //从抽象父类继承过来的，必须实现
18         public function login($n,$p){
19             /*
20             省略了访问数据库，判断用户是否登录成功
21             */
22             $result = false; //登录成功或失败
23             if($result){
24                 $this->username = $n;
25                 $this->password = $p;
26                 $this->authority = ''; //赋予正确的权限值
27                 return true;
28             }else{
29                 return false;
30             }
31         }
32         public function addTeacher(Teacher $teacher){
33         }
34         public function addStudent(Student $student){
35         }
36     }
37 }
```



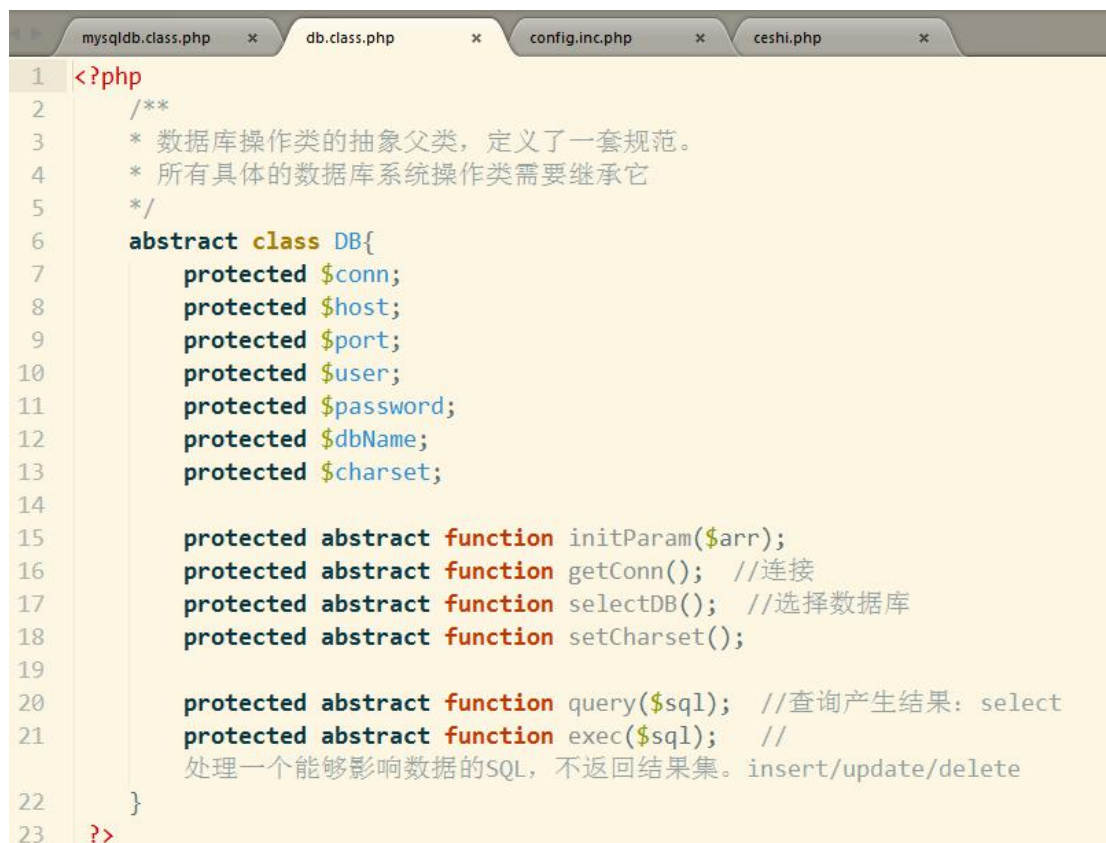
```
38      /**
39       * 教师类
40       */
41      class Teacher extends User{
42          public function __construct(){
43              $this->type = '教师';
44          }
45          public function login($n,$p){
46              $result = false; //登录成功或失败
47              if($result){
48                  $this->username = $n;
49                  $this->password = $p;
50                  return true;
51              }else{
52                  return false;
53              }
54          }
55          public function setScore(Student $stu,$score){
56
57          }
58      }
```

```
59      /**
60       * 学生类
61       */
62      class Student extends User{
63          //新增的其他私有属性
64          private $score;
65          public function __construct(){
66              $this->type = '学生';
67          }
68          public function login($n,$p){
69              $result = false; //登录成功或失败
70              if($result){
71                  $this->username = $n;
72                  $this->password = $p;
73                  return true;
74              }else{
75                  return false;
76              }
77          }
78          public function getScore(){
79              return $this->score;
80          }
81      }
82
83
84      ?>
```

## 设计练习 2:

当我们设计一个管理系统时，如果考虑将来可能需要更换数据库管理系统（比如从 MySQL 数据库管理系统移植到 SQL Server 数据库管理系统），为了提高代码的可移植性，如何在项目初期就设计出针对不同数据库系统的访问操作类。这里主要练习 MySQL 数据库访问类的定义方法。

1、定义一个针对所有数据库系统访问类的设计规范,抽象类 DB。



```
1 <?php
2 /**
3  * 数据库操作类的抽象父类，定义了一套规范。
4  * 所有具体的数据库系统操作类需要继承它
5  */
6 abstract class DB{
7     protected $conn;
8     protected $host;
9     protected $port;
10    protected $user;
11    protected $password;
12    protected $dbName;
13    protected $charset;
14
15    protected abstract function initParam($arr);
16    protected abstract function getConn(); //连接
17    protected abstract function selectDB(); //选择数据库
18    protected abstract function setCharset();
19
20    protected abstract function query($sql); //查询产生结果: select
21    protected abstract function exec($sql); //
22    // 处理一个能够影响数据的SQL，不返回结果集。insert/update/delete
23 }
```

2、定义 MySQLDB 类，继承 DB 类。实现具体功能。

```
mysqlb.class.php x config.inc.php x ceshi.php x
1 <?php
2 include "db.class.php";
3 /**
4  * 封装了mysql数据库的所有操作
5  */
6 class MySQLDB extends DB
7 {
8     private $error;
9     protected function initParam($arr){
10         //用户、密码、数据库名必须由用户指定
11         if (!isset($arr[user])) {
12             die("必须提供连接mysql服务器的用户名");
13         } else if (!isset($arr[password])) {
14             die("必须提供连接mysql服务器的用户密码");
15         } else if (!isset($arr[dbName])) {
16             die("必须提供具体的数据库");
17         }
18         $this->host = isset($arr[host])?$arr[host]:'localhost';
19         $this->port = isset($arr[port])?$arr[port]:'3306';
20         $this->user = $arr[user];
21         $this->password = $arr[password];
22         $this->dbName = $arr[dbName];
23         $this->charset = isset($arr[charset])?$arr[charset]:'utf8';
24     }
25     //建立连接
26     protected function getConn(){
27         $conn = mysql_connect($this->host.":".$this->port,$this->user,$this->password);
28         /*echo mysql_error();*/
29         if (!$conn) {
30             die('连接mysql服务器失败, 请检查服务器地址' . $this->host . ":" . $this->port . "
31                 , 用户名{$this->user},密码{$this->password}是否正确。");
32         } else {
33             $this->conn = $conn;
34         }
35     }
36     //选择数据库
37     protected function selectDB(){
38         $result = mysql_select_db($this->dbName,$this->conn);
39         if (!$result) {
40             die("数据出错, 请检查配置文件的数据配置项, 是否存在此数据库{$this->dbName}".
41                 mysql_error());
42         }
43     }
44     protected function setCharset(){
45         mysql_query("set names {$this->charset}", $this->conn);
46     }
47 }
```



```
45 //查询产生结果集，以二维数组的形式返回.select
46 public function query($sql){
47     $result = mysql_query($sql,$this->conn);
48     if (!$result) {
49         $this->error = mysql_error();
50         return false;
51     } else {
52         $rsArr = array();
53         while ($row = mysql_fetch_assoc($result)) {
54             $rsArr[]=$row;
55         }
56         return $rsArr;
57     }
58 }
59 }
60 //处理一个能够影响数据的SQL，不返回查询结果集insert/update/delete
61 //如果执行成功返回的是受影响的行数int;失败返回false
62 public function exec($sql){
63     $result = mysql_query($sql,$this->conn);
64     if ($result===true) {
65         return mysql_affected_rows(); //可能是0
66     } else {
67         $this->error = mysql_error();
68         return false;
69     }
70 }
71 }
72 function __construct($arr)
73 {
74     $this->initParam($arr);
75     $this->getConn();
76     $this->selectDB();
77     $this->setCharset();
78 }
79 public function getError(){
80     return $this->error;
81 }
82 }
83
84 ?>
```

### 3、简单测试。

```
mysqlb.class.php x config.inc.php x
1 <?php
2
3     /*
4     此文档是项目的配置文件
5     */
6
7     /**
8     * [$config 数据库的初始化参数配置项]
9     * user 、 password、 dbName三项是必填项
10    *
11    */
12    $config = array(
13    /*          'host'=>'localhost',
14              'port'=>'3306',*/
15              'user'=>'root',
16              'password'=>'root',
17              'dbName'=>'jytxl',
18    /*          'charset'=>'utf8'*/
19    );
20 ?>
```

```
mysqlb.class.php x config.inc.php x ceshi.php x
1 <?php
2 include 'mysqlb.class.php'; //包含类文件
3 include 'config.inc.php'; //包含配置文件
4 $dao = new MySQLDB($config); //实例化数据库操作类对象
5 var_dump($dao); //测试对象初始化情况
6
7 //执行一个查询类的操作
8 $rsArr = $dao->query("select * from member where uid=1");
9 //变量是0, '', null, array() ==false
10 if (!empty($rsArr)) {
11     foreach ($rsArr as $row) {
12         echo "$row[id]<br>";
13     }
14
15 } else if($rsArr===false){
16     echo $dao->getError();
17 }else{
18     echo "没有满足条件的查询结果";
19 }
```

```
20
21 //执行一个插入类的操作
22 $rs = $dao->exec("insert into member(name,telephone,uid)
    values('abc','123',1),('def','456',1)");
23 /*'delete from member where id=33'*/
24 if ($rs===false) {
25     echo $dao->getError();
26
27 } else {
28     echo "插入成功{$rs}几条记录";
29 }
30
31 ?>
```

## 二、终极类与终极方法

**关键字：final 代表最终**

1、在类前使用 final 关键字，代表该类不能被继承，成为终极类。

语法：final class 类名

```
1 <?php
2     abstract class Goods{
3
4     }
5     final class Phone extends Goods{
6
7     }
8     class IPhone extends Phone{
9
10    }
11    $i = new IPhone();
12    var_dump($i);
13
14
15 ?>
```

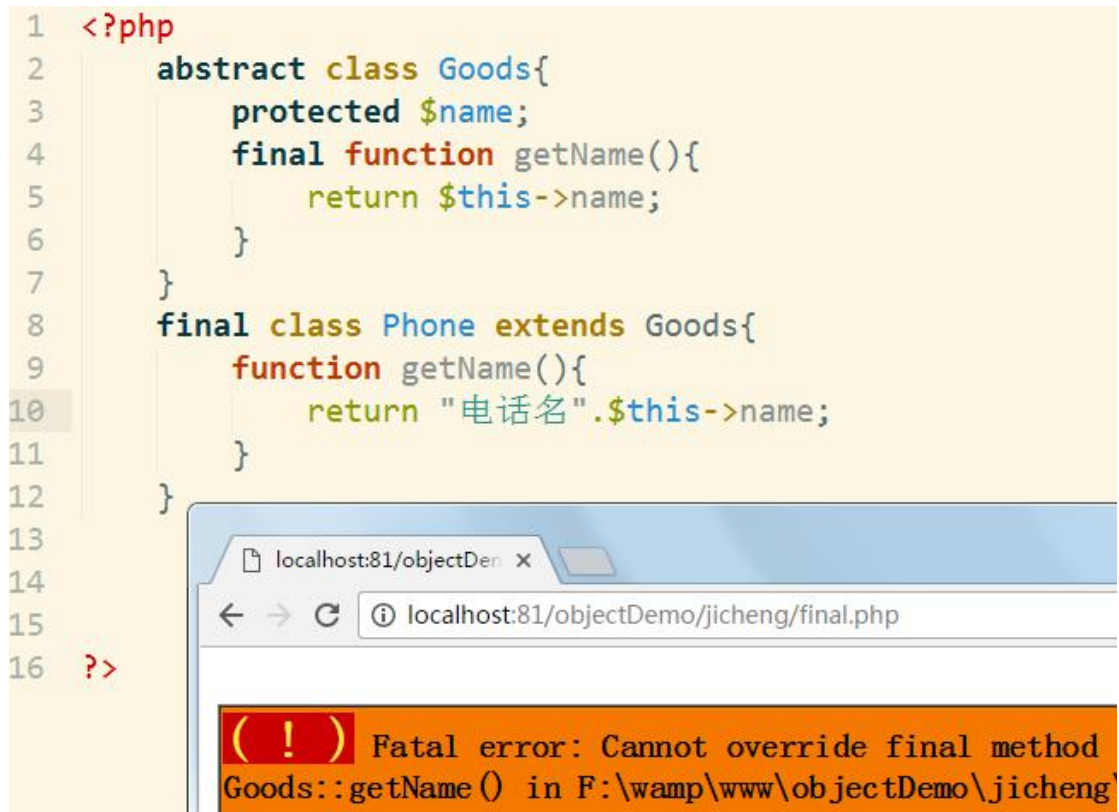


The screenshot shows a web browser window with the address bar displaying 'localhost:81/objectDemo/jicheng/final.php'. Below the address bar, a red error message is displayed: '(!) Fatal error: Class IPhone may not inherit from final class (Phone) in F:\wamp\www\objectDemo\jicheng\final.php on line 8'. The error message is highlighted in a red box.

2、在方法名前使用 final 关键字，代表该方法不能被子类重写，成为

终极方法。

语法：final function 方法名(参数列表){....}



为什么要使用终极类和终极方法？

从设计层面考虑某个子类已经是最终类了，不需要再进行功能扩展，那为了防止该类被继承就可以使用 final 设置其为终极类。

当一个父类的方法确定不能被子类改写的时候，就是说该方法在所有子类中的实现是完全一致时，可以设置其为终极方法避免子类的重写。