**面向对象的PHP：**

**对象做动作：**

**类的创建**

**类在它里面定义了一些属性和方法（方法中执行一些操作，但是需要有参数的支持），相当于一种协议**

**类的继承是对协议的完善，添加或重载（覆盖）新的属性和方法**

**类的实例化->得到一个对象**

**这个对象通过给对应的类传递参数，对应的类被创建**

**类和对象：**

对象：具有属性（成员变量）和操作（方法）的集合

类：对对象进行分类，是对象的集合，根据分类的标准不同，分出来的类会不同。

是一种抽象，相当于一种协议（要实现他就需要遵守它的协议）

如：自行车和汽车在交通工具类里

qq和奥迪在汽车类里

类里的对象要具有一些共同点。

**多态性：**

不同的类（像协议）对同一操作可以有不同的行为，也就是行为的特性，在PHP中只有类的方法可以是多态的，当对象确定了，其行为就确定了（**对象做动作**）

**继承：**

使用子类在类之间创建层次关系，子类将从他的父类继承属性和动作

用关键字**extends**指明继承关系

如：class B extends A //B类继承自A类

1.如果一个属性或方法被指定为private，那么他不会被继承

如果一个属性或方法被指定为protected，那么他会被继承，但只能在子类里使用，相当于在子类中被指定为private

2.PHP支持单重继承，每个类至多一个父类

不支持多重继承，每个类可以有多个父类，可以通过接口解决

**类的创建：**

1. 以关键字**class**开始

如：class class\_name{ }

2.构造函数，名称必须是 **\_ \_construct( )**

如：function \_ \_construct( ){ }

PHP支持构造函数的重载（这里指的重载相当于C++的覆盖，指在子类再次声明和父类相同的属性和操作）

3.类里的属性（成员变量），有public，private，protected， 决定了变量的作用域

**类的实例化：**

实例化一个类，使用关键字**new**创建一个对象，需要指明是哪一个类的实例（**我这个对象遵守的是哪个协议**），并通过构造函数提供需要的参数

**使用类的属性：**

public的可以直接访问，

private的要通编写过访问器函数**\_ \_get( )**和 **\_ \_set( )**，我们不会直接访问这些函数，函数名称前面的双下划线表明这些函数在PHP中有特殊的意义

如：$a = new classname( );

$b = $a->$attribute; //该语句间接调用了\_ \_get( )函数

$a->$attribute = 5; //该语句间接调用了\_ \_set( )函数

在\_ \_get( )和 \_ \_set( )函数中可以编写对数据进行检查

**Per-Class常量：（PHP的类常量）**

1.在类中始终保持不变的值定义为常量。在定义和使用常量的时候不需要使用 $ 符号。

常量的值必须是一个定值，不能是变量，类属性，数学运算的结果或函数调用。

接口（interface）中也可以定义常量。

1. 每个类中可以定义自己类中的常量，用**const**关键字
2. 在类外通过**：：（双冒号）**操作符指定常量所属的类来访问

如：

<?Php

class Math{

**const pi** = 3.14159;

}

?>

echo **Math :: pi** ;

或者

$a = new Math();

echo $a::pi;

就是说允许在未初始化类的情况下就可以调用这个常量

**静态方法：**

在对应的方法前使用**static**关键字，允许在未初始化类的情况下就可以调用这个方法

<?Php

**static** function squared($input){

return $input\*$input;

}

?>

echo **Math :: squared(8)** ;

或者

$a = new Math();

echo $a::squared(8) ;

注意：在一个静态方法中，不能使用this关键字，因为可能没有可以引用的对象实例（因为允许在未初始化类的情况下就可以调用这个方法）。

**延迟静态绑定：**

允许一个静态继承的上下文中（**test( )**）对一个被调用类（**who( )**）的引用，父类可以使用子类重载的静态方法

class A{  
 public **static** function who(){  
 echo \_\_CLASS\_\_;  
 }  
 public **static** function test(){  
 **static::**who(); //延迟静态绑定  
 }  
}  
  
  
class B **extends** A{  
 public **static** function who(){  
 echo \_\_CLASS\_\_;  
 }  
}  
  
A::who();  
A::test();  
B::who();  
B::test();

输出为：AABB

test( )引用的who为他所在类的who（）

如果是self::who();

输出：AABA

取决于定义当前方法所在的类

**克隆对象：**

在变量前使用**clone**关键字，允许复制一个已有的对象（副本）；

如：$c = clone $b;

需要**\_ \_clone（）**方法，类似于\_ \_get( )和\_ \_set( )，在使用默认行为创建一个副本后能够被调用，所以在\_ \_clone（）方法定义自己需要的修改

**\_ \_call( )重载方法：**

\_ \_call( 被调用的方法名称，传递给该方法的参数数组)

当一个类中包含了\_ \_call( )方法，那么当调用这个类中不存在的方法，\_ \_call( )方法会被自动调用。

作用：

避免当调用不存在的方法而产生的错误

提供和重载的方法

例子：

class A{  
 function \_\_call($method,$args)  
 {  
 if ($method == "display") {  
 if (is\_object($args[0])) {  
 $this->displayObject(**$args[0]**);  
 }elseif (is\_array($args[0])) {  
 $this->displayArray($args[0]);  
 }else{  
 $this->displayScalar($args[0]);  
 }  
 }  
 }  
 function displayObject($a){  
 echo $a;  
 }  
 function displayArray($a){  
 foreach($a as $b){  
 echo $b;  
 }  
 }  
 function displayScalar($a){  
 echo strval($a);  
 }  
}

$a = new A;  
$a->display(array(1));  
$a->display('abc');

输出：1abc

注意：$args代表的是参数数组，当$a->display(1，2，'abc');数组为a[0]=1;a[1]=;a[2]=abc;

**\_ \_autoload( )方法：**

定义一个 [\_\_autoload($classname待加载的类名)](http://php.net/manual/zh/function.autoload.php) 函数，它会在试图使用尚未被定义的类（该类包含在其他的php源文件中）时自动调用。通过调用此函数，脚本引擎在 PHP 出错失败前有了最后一个机会加载所需的类。

注意：php文件名需要和类名一样，一个文件中只有一个类

如：

function \_\_autoload($name){  
 include\_once $name.".php";  
}  
  
$obj = new main(); //在实例化前，包含进main.php文件的main类，main当作参数传递给$name  
$obj->display();