Method overloading: Khả năng thực hiện đa hình thông qua phương pháp quá tải không được hỗ trợ bởi PHP và có lẽ sẽ không bao giờ.

**Operator overloading:**   
Khả năng gán ý nghĩa bổ sung cho các nhà khai thác dựa trên các kiểu dữ liệu bạn đang cố gắng để sửa đổi hiện không được hỗ trợ bằng PHP

Dựa trên các cuộc thảo luận tìm thấy trong danh sách gửi thư của nhà phát triển PHP, nó là không chắc rằng tính năng này sẽ không bao giờ được thực hiện.

Multiple inheritance: PHP không hỗ trợ đa kế thừa. tuy nhiên ,Thực hiện

của nhiều giao diện được hỗ trợ,.

**Obiject cloning**

destinationObject = clone targetObject;

<?php  
class Corporate\_Drone {  
private $employeeid;  
private $tiecolor;  
www.it-ebooks.info  
CHAPTER 7 ◼ADVANCED OOP FEATURES  
161  
// Define a setter and getter for $employeeid  
function setEmployeeID($employeeid) {  
$this->employeeid = $employeeid;  
}  
function getEmployeeID() {  
return $this->employeeid;  
}  
// Define a setter and getter for $tiecolor  
function setTieColor($tiecolor) {  
$this->tiecolor = $tiecolor;  
}  
function getTieColor() {  
return $this->tiecolor;  
}  
}  
// Create new Corporate\_Drone object  
$drone1 = new Corporate\_Drone();  
// Set the $drone1 employeeid property  
$drone1->setEmployeeID("12345");  
// Set the $drone1 tiecolor property  
$drone1->setTieColor("red");  
// Clone the $drone1 object  
$drone2 = clone $drone1;  
// Set the $drone2 employeeid property  
$drone2->setEmployeeID("67890");  
// Output the $drone1 and $drone2 employeeid properties  
printf("Drone1 employeeID: %d <br />", $drone1->getEmployeeID());  
printf("Drone1 tie color: %s <br />", $drone1->getTieColor());  
printf("Drone2 employeeID: %d <br />", $drone2->getEmployeeID());  
printf("Drone2 tie color: %s <br />", $drone2->getTieColor());  
?>

Result

Drone1 employeeID: 12345  
Drone1 tie color: red  
Drone2 employeeID: 67890  
Drone2 tie color: red

**Các hàm trợ giúp trong việc quản lí và hỗ trợ class**

Ví dụ # 1 class\_alias () (đặt bí danh cho class)

<?php  
  
class foo { }  
  
class\_alias('foo', 'bar');  
  
$a = new foo;  
$b = new bar;  
  
// the objects are the same  
var\_dump($a == $b, $a === $b);  
var\_dump($a instanceof $b);  
  
// the classes are the same  
var\_dump($a instanceof foo);  
var\_dump($a instanceof bar);  
  
var\_dump($b instanceof foo);  
var\_dump($b instanceof bar);  
  
?>

Result:

bool (true)

bool (sai)

bool (true)

bool (true)

bool (true)

bool (true)

bool (true)

vi dụ 2:

<?php  
class A {};  
class B1 extends A {};  
class\_alias('A', 'B2');  
  
$b1 = new B1; echo get\_class($b1); // prints B1  
$b2 = new B2; echo get\_class($b2); // prints A !  
?>

**Class\_exists(kiểm tra xem class có tồn tại hay ko)**

<?php  
// Check that the class exists before trying to use it  
if (class\_exists('MyClass')) {  
    $myclass = new MyClass();  
}  
  
?>

Vd2:

<?php  
function \_\_autoload($class)  
{  
    include($class . '.php');  
  
    // Check to see whether the include declared the class  
    if (!class\_exists($class, false)) {  
        trigger\_error("Unable to load class: $class", E\_USER\_WARNING);  
    }  
}  
  
if (class\_exists('MyClass')) {  
    $myclass = new MyClass();  
}  
  
?>

**Get\_class()** (lấy tên class)

Example:

<?php  
  
class foo {  
    function name()  
    {  
        echo "My name is " , get\_class($this) , "\n";  
    }  
}  
  
// create an object  
$bar = new foo();  
  
// external call  
echo "Its name is " , get\_class($bar) , "\n";  
  
// internal call  
$bar->name();  
  
?>

Result:

Its name is foo

My name is foo

Example 2:

<?php  
  
abstract class bar {  
    public function \_\_construct()  
    {  
        var\_dump(get\_class($this));  
        var\_dump(get\_class());  
    }  
}  
  
class foo extends bar {  
}  
  
new foo;  
  
?>

Result:

string(3) "foo"

string(3) "bar"

**get\_class\_methods(Lấy ra tên các hàm của class )**

<?php  
  
class myclass {  
    // constructor  
    function myclass()  
    {  
        return(true);  
    }  
  
    // method 1  
    function myfunc1()  
    {  
        return(true);  
    }  
  
    // method 2  
    function myfunc2()  
    {  
        return(true);  
    }  
}  
  
$class\_methods = get\_class\_methods('myclass');  
// or  
$class\_methods = get\_class\_methods(new myclass());  
  
foreach ($class\_methods as $method\_name) {  
    echo "$method\_name\n";  
}  
  
?>

The above example will output:

myclass

myfunc1

myfunc2

example2:

<?php  
class C  
{  
    private function privateMethod()  
    {  
          
    }  
    public function publicMethod()  
    {  
          
    }  
    public function \_\_construct()  
    {  
        echo '$this:';  
        var\_dump(get\_class\_methods($this));  
        echo 'C (inside class):';  
        var\_dump(get\_class\_methods('C'));  
    }  
}  
$c = new C;  
echo '$c:';  
var\_dump(get\_class\_methods($c));  
echo 'C (outside class):';  
var\_dump(get\_class\_methods('C'));  
?>  
  
Output:  
  
$this:  
array  
  0 => string 'privateMethod' (length=13)  
  1 => string 'publicMethod' (length=12)  
  2 => string '\_\_construct' (length=11)  
  
C (inside class):  
array  
  0 => string 'privateMethod' (length=13)  
  1 => string 'publicMethod' (length=12)  
  2 => string '\_\_construct' (length=11)  
  
$c:  
array  
  0 => string 'publicMethod' (length=12)  
  1 => string '\_\_construct' (length=11)  
  
C (outside class):  
array  
  0 => string 'publicMethod' (length=12)  
  1 => string '\_\_construct' (length=11)