TRƯỜNG ĐẠI HỌC ĐÀ LẠT
KHOA CÔNG NGHỆ THÔNG TIN

# PHÁT TRIỂN ỨNG DỤNG WEB

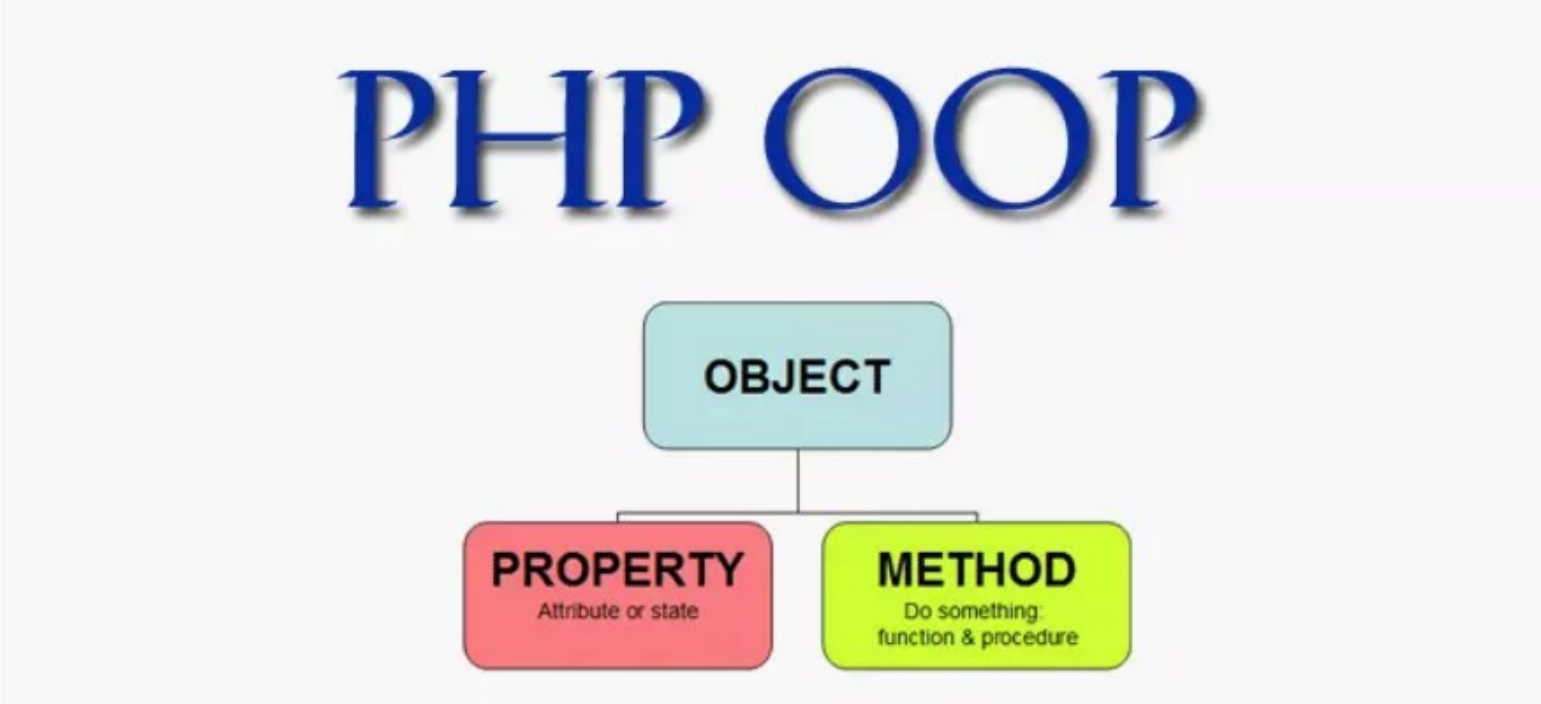
Thái Duy Quý
ITFacDLU – quytd@dlu.edu.vn

# Phần 5. PHP OOP

# NỘI DUNG

1. Giới thiệu
2. Các vấn đề cơ bản hướng đối tượng trong PHP
3. Lớp abstract và lớp interfaces
4. Hàm include và require

# 1/ Giới thiệu




- OOP programming techniques may include four features:

  ***Encapsulation***, ***inheritance, polymorphism*** and ***abstraction***

# 2/ Các vấn đề cơ bản OOP trong PHP

1. **Declaring a Class**

- **Cú pháp khai báo lớp**:

```
class <Tên_lớp>{
    // Your code is here
    …
}
```

- **Ví dụ**:

```
class foo {
        const BAR = "Hello World";
}
echo foo::BAR;
```

# 2/ Các vấn đề cơ bản OOP trong PHP

1. **Declaring a Class**

- **Cú pháp khai báo lớp kế thừa**:

```
class a {
    function test(){ echo "a::test called";}
    function func(){echo "a::func called";}
}
class b extends a {
    function test(){echo "b::test called";}
}
class c extends b {
    function test(){parent::test();}
}
class d extends c {
    function test(){b::test();}
}
```

- **Cú pháp xác định lớp đối tượng**:

```
if ($obj instanceof MyClass) {
        echo "\$obj is an instance of MyClass";
}
```

# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object

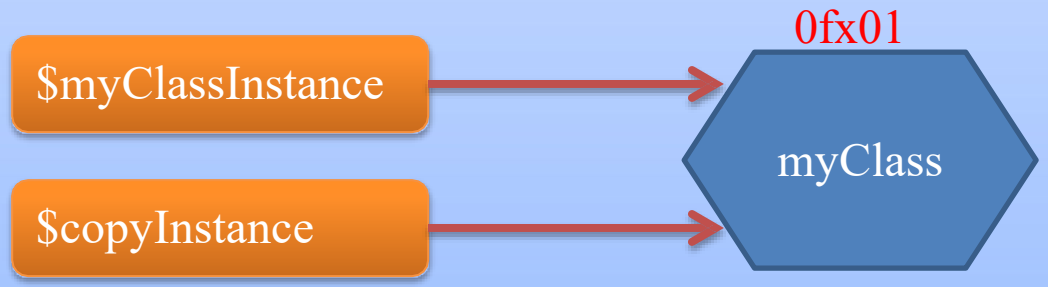
- **Cú pháp tạo đối tượng**:

```
$myClassInstance = new myClass();
```

**Lưu ý**: các đối tượng trong PHP được sử dụng theo dạng tham chiếu

- **Ví dụ**:

```
$myClassInstance = new myClass();
$copyInstance = $myClassInstance();
// Cả 2 biến $myInstance và $copyInstance
cùng trỏ tới một đối tượng thuộc myClass.
```



# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object

- **Phương thức và thuộc tính**:

```
class myClass {
    function myFunction() {
        echo "You called myClass::myFunction";
    }
}
// Access methods of class myClass
$obj = new myClass();
$obj -> myFunction();
```

# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object

- **Con trỏ $this**:

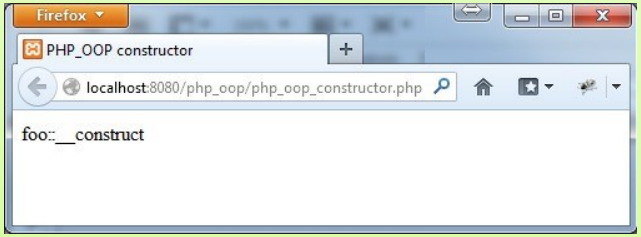
```
class myClass {
    function myFunction($data) {
        echo "The value is $data";
    }
    function callMyFunction($data) {
        // Call myFunction()
        $this->myFunction($data);
    }
}
$obj = new myClass();
$obj->callMyFunction(123);
```

# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object
3. Constructors

- **Cú pháp hàm khởi tạo**:

```
class foo {
    function __construct()
    {
        // PHP 5 new  style constructor
        echo __METHOD__;
    }
    function foo()
    {
        // PHP 4 style constructor
    }
}
new foo();
```

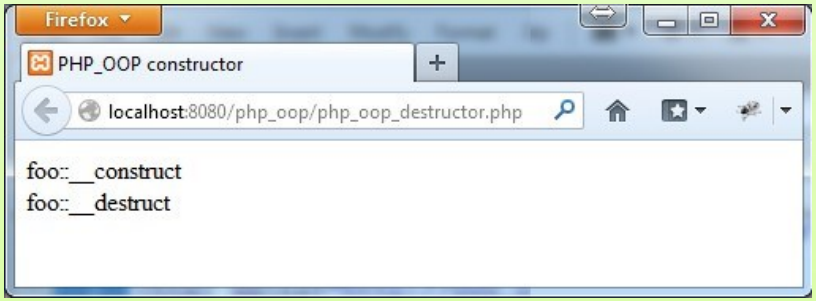


# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object
3. Constructors
4. Destructors

- **Cú pháp hàm hủy**:

```
class foo {
    function__construct()
    {
        echo__METHOD__. PHP_EOL;
    }
    function__destruct()
    {
        echo__METHOD__;
    }
}
new foo();
```



# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object
3. Constructors
4. Destructors
5. Visibility

- **Phạm vi truy cập**:

| Key | Visibility |
|---|---|
| public | The resource can be accessed from any scope. |
| protected | The resource can only be accessed from within the class where it is defined and its descendants |
| private | The resource can only be accessed from within the class where it is defined. |
| final | The resource is accessible from any scope, but cannot be overridden in descendant classes. |

- **Ví dụ**:

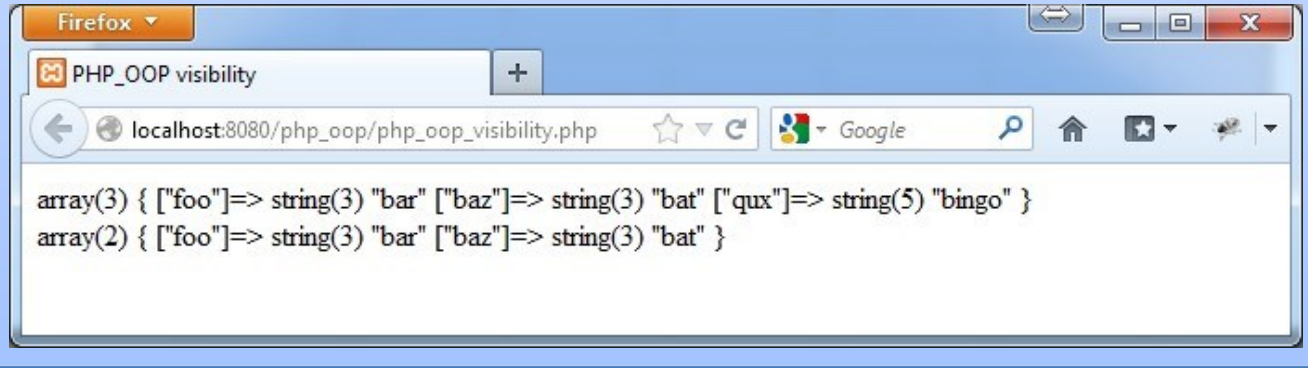
# 2/ Các vấn đề cơ bản OOP trong PHP

1. **Declaring a Class**
2. **Instantiating an Object**
3. **Constructors**
4. **Destructors**
5. **Visibility**

- **Ví dụ 1**: kết quả của đoạn lệnh sau

```
class foo {
    public $foo = 'bar'; protected $baz = 'bat'; private $qux = 'bingo';
    function__construct(){
        var_dump(get_object_vars($this));
    }
}
class bar extends foo {
    function __construct(){
        var_dump(get_object_vars($this));
    }
}
new foo();
new bar();
```

**var_dump():** In thông tin của biến gồm kiểu dữ liệu và giá trị.
**get_object_vars()**: Lấy các thuộc tính của đối tượng.

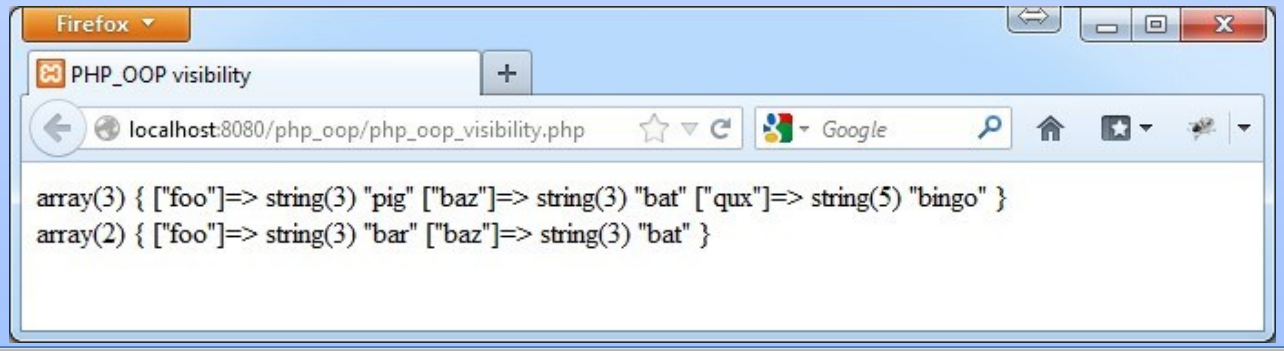


# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object
3. Constructors
4. Destructors
5. Visibility

- **Ví dụ 2**: kết quả của đoạn lệnh sau

```
class foo {
    public $foo = 'bar'; protected $baz = 'bat'; private $qux = 'bingo';
    function___construct(){
        $this->foo="pig";
        var_dump(get_object_vars($this)); echo "<br>";
    }
}
class bar extends foo {
    function __construct(){
        var_dump(get_object_vars($this)); echo "<br>";
    }
}
new foo();
new bar();
```



# 2/ Các vấn đề cơ bản OOP trong PHP

1. Declaring a Class
2. Instantiating an Object
3. Constructors
4. Destructors
5. Visibility
6. Constants, Static Methods and Properties

## Cú pháp khai báo biến và phương thức tĩnh:

```
class foo {
    static $bar = "bat";
    static public function baz(){
        echo "Hello World";
    }
}
echo foo::$bar."<br>";
foo::baz();
```

```
// output:
bat
Hello world
```

## Cú pháp khai báo hằng trong lớp:

```
class foo {
    const BAR = "Hello World";
}
echo foo::BAR;
```

```
// output:
Hello world
```

# 3/ Lớp abstract và lớp interface

## a. Giới thiệu

Animal

Transport

Interface
- ✓ Go
- ✓ Run
- ✓ Fly
- ✓ Swim
- ✓ …

Chim và máy bay có cùng interface Fly nhưng cách thức hoạt động của Fly là khác nhau hoàn toàn

Interface
- ✓ Go
- ✓ …

Interface
- ✓ Run
- ✓ …

## b. Lớp trừu tượng

- **Khái niệm**
  - Lớp trừu tượng là một lớp cha cho tất cả các lớp có cùng bản chất. Do đó mỗi lớp dẫn xuất (lớp con) chỉ có thể kế thừa từ một lớp trừu tượng.
  - Lớp trừu tượng không cho phép tạo instance (không thể tạo được các đối tượng thuộc lớp đó).
- **Cú pháp khai báo lớp trừu tượng**

```
abstract class<class_name>{
  [properties … ]
   abstract function func_name(…);
   public function func_name(…);
}
```

# 3/ Lớp abstract và lớp interface

## b. Lớp trừu tượng

- **Ví dụ**

```
abstract class DataStore_Adapter {
    private $id;
    abstract function insert();
    abstract function update();
    public function save(){
        if (!is_null($this->id)){
            $this->update();
        } else {
            $this->insert();
        }
    }
}
class PDO_DataStore_Adapter extends DataStore_Adapter {
    public__construct($dsn){// ...    }
    function insert(){   // ...    }
    function update(){ //...     }
}
```

## c. Lớp interface

- **Khái niệm**
  - Lớp interface được xem như một mặt nạ cho tất cả các lớp cùng cách thức hoạt động nhưng có thể khác nhau về bản chất.
  - Lớp dẫn xuất có thể kế thừa từ nhiều lớp interface để bổ sung đầy đủ cách thức hoạt động của mình (đa kế thừa - Multiple inheritance).
- **Cú pháp khai báo lớp interface**

```
interface class <class_name>{
  …
}
```

## c. Lớp interface

- **Ví dụ**

```
interface DataStore_Adapter {
    public function insert();
    public function update();
    public function save();
    public function newRecord($name = null);
}

class PDO_DataStore_Adapter implements DataStore_Adapter {
    public function insert(){   // ...    }
    public function update(){ // ...    }
    public function save(){         // ...    }
    public function newRecord($name = null){ }
}
```

# 4/ Hàm include và require

## a. Công dụng

- **Giống nhau**:
  - Chèn file vào file hiện tại, nếu file được chèn có lỗi thì hiện thông báo lỗi
- **Khác nhau**:
  - Khi file được chèn bằng lệnh **require**() có lỗi thì trình biên dịch sẽ dừng lại, không dịch nữa và sẽ xuất hiện thông báo lỗi.
  - Khi file được chèn bằng lệnh **include**() có lỗi thì trình biên dịch vẫn tiếp tục dịch cho đến hết, đồng thời cũng xuất hiện ***warning*** để cảnh báo file đó bị lỗi.

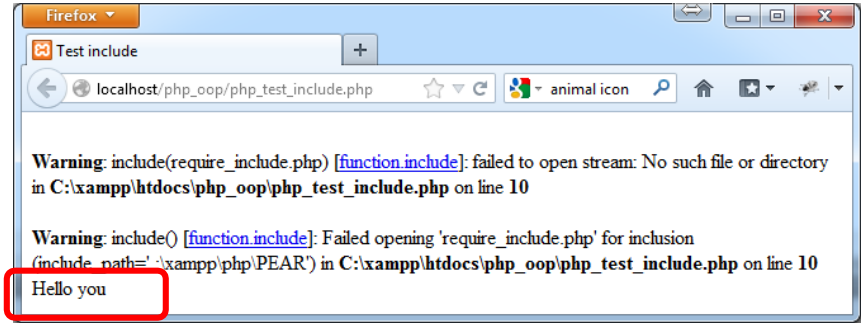
# 4/ Hàm include và require

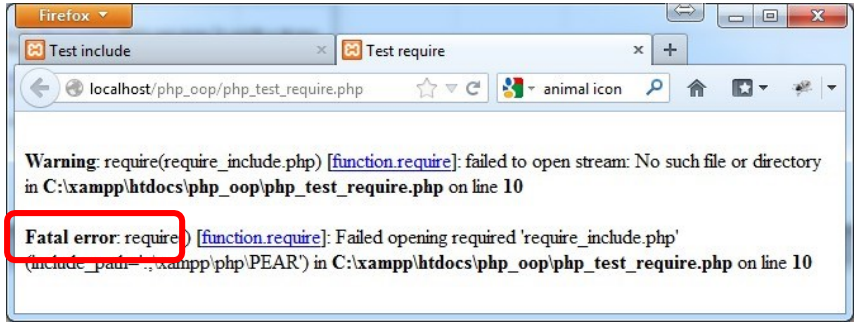
## b. Ví dụ

- Giả sử 2 đoạn chương trình sau cùng sử dụng tập tin **require_include.php** không tồn tại như sau:

```
<html>
<head><title>Test include</title></head>
<body>
<?php
    include("require_include.php");
    echo "Hello you";
?>
</body>
</html>
```

```
<html>
<head><title>Test include</title></head>
<body>
<?php
    require("require_include.php");
     echo "Hello you";
?>
</body>
</html>
```



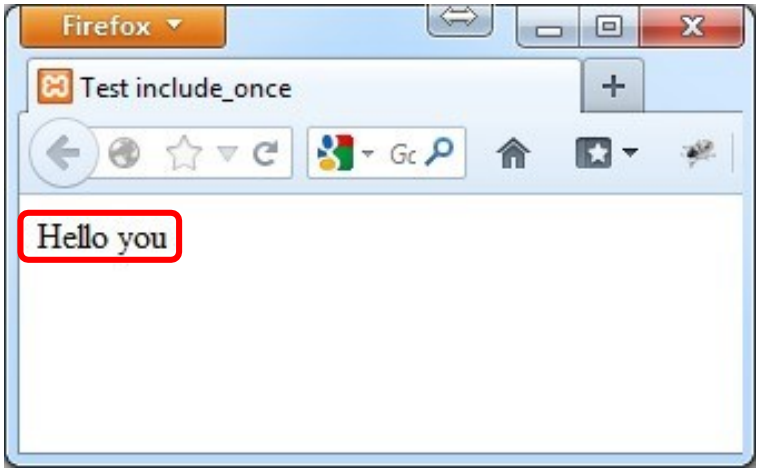
# 4/ Hàm include và require

## c. Hàm include_once và require_once

- Là 2 hàm biến đổi của hàm include và require nhằm mục đích nếu tập tin đã được chèn trước đó thì không chèn nữa.
- **Ví dụ**: giả sử có tập tin **require_include_once.php** như sau:

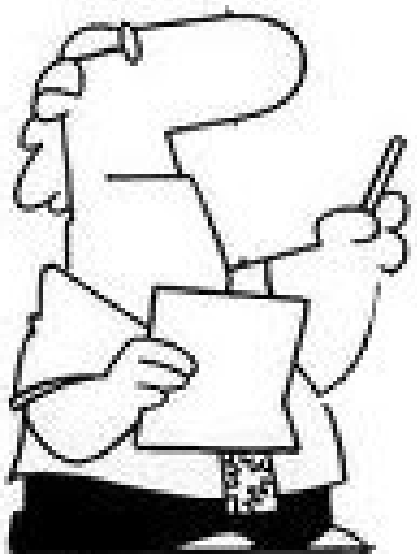
```php
<?php
    // Tập tin require_include_once.php
    echo "Hello you <br>";
?>
```

```php
<!– Tập tin test_require_include_once.php -->
<html>
<head><title>Test include_once</title></head>
<body>
<?php
    include_once('require_include_once.php');
    include_once('require_include_once.php');
?>
</body>
</html>
```



# Thank you !

# PHẦN THẢO LUẬN



- Họ tên:
- Mã SV:
- Lớp:
- Khoá:
- Email:

| | 01 | 02 | … | 19 | 20 |
|---|---|---|---|---|---|
| A | | | | | |
| B | | | | | |
| C | | | | | |
| D | | | | | |

# PHẦN BÀI TẬP

**Câu 1**: Cho biết các phần tử được chứa trong mảng $a->array (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class my_class
2.  {
3.      var $my_value = array();
4.      function my_class ($value) {
5.          $this->my_value[] = $value;
6.      }
7.      function set_value ($value) {
8.          $this->my_value[] = $value;
9.      }
10. }
11. $a = new my_class('a');
12. $a->my_value[] = 'b';
13. $a->set_value ('c');
14. $a->my_class('d');
?>
```

A. $a->array chứa các phần tử: a, b, c và d
B. $a->array chứa các phần tử: a và d
C. Đoạn lệnh sai ở dòng 3
D. Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 2**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class my_class
2.  {
3.      function my_method1 ($value) {
4.          echo $value;
5.      }
6.      function my_method2 ($value) {
7.          $this->my_method1(123);
8.      }
9.      function my_method3 ($value) {
10.         my_class::my_method1(456);
11.     }
12. }
13. $obj = new my_class;
14. $obj->my_method3(123);
?>
```

A. Đoạn lệnh xuất kết quả 123
B. Đoạn lệnh xuất kết quả 456
C. Đoạn lệnh sai ở dòng 7
D. Đoạn lệnh sai ở dòng 10

# PHẦN BÀI TẬP

**Câu 3**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class my_class
2.  {
3.      var $my_var;
4.      function _my_class ($value) {
5.          $this->my_var = $value;
6.      }
7.  }
8.  $a =new my_class (10);
9.  echo $a->my_var;
?>
```

A. Đoạn lệnh báo lỗi
B. Đoạn lệnh xuất kết quả 10
C. Đoạn lệnh xuất kết quả Null
D. Đoạn lệnh không xuất kết quả nào hết.

# PHẦN BÀI TẬP

**Câu 4**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class my_class
2.  {
3.      var $value;
4.  }
5.  $a =new my_class;
6.$a->my_value= 5;
7.  $b = $a;
8.  $b->my_value= 10;
9.  echo $a->my_value;
?>
```

A. Đoạn lệnh xuất kết quả 5
B. Đoạn lệnh xuất kết quả 10
C. Đoạn lệnh xuất kết quả Null
D. Đoạn lệnh không xuất kết quả nào hết.

# PHẦN BÀI TẬP

**Câu 5**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class a {
2.      function a($x =1){
3.          $this->myvar = $x;
4.      }
5.  }
6.  class b extends a {
7.      var $myvar;
8.      function b($x =2) {
9.          $this->myvar = $x;
10.         parent::a();
11.     }
12. }
13. $obj= new b;
14. echo $obj->myvar;
?>
```

A. Đoạn lệnh xuất kết quả 1
B. Đoạn lệnh xuất kết quả 2
C. Đoạn lệnh báo lỗi do biến $myvar chưa khai báo
D. Đoạn lệnh không xuất kết quả nào hết.

# PHẦN BÀI TẬP

**Câu 6**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class a
2.  {
3.      function a(){
4.          echo 'Parent called';
5.      }
6.  }
7.  class b
8.  {
9.      function b(){
10.         parent::a();
11.     }
12. }
13. $c =new b();
?>
```

A. Đoạn lệnh xuất kết quả "Parent called"
B. Đoạn lệnh sai ở dòng số 7
C. Đoạn lệnh sai ở dòng số 10
D. Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 7**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class test
2.  {
3.      public function method1(){
4.          global $x;
5.          $x = 15;
6.          echo $x;
7.      }
8.      public function method2(){
9.          echo $x;
10.     }
11. }
12. $obj = new test();
13. $obj->method1();
14. $obj->method2();
?>
```

A. Đoạn lệnh xuất kết quả "15" và "15"
B. Đoạn lệnh xuất kết quả "15" và một thông báo notice
C. Đoạn lệnh sai ở dòng số 4
D. Đoạn lệnh sai ở dòng số 9

# PHẦN BÀI TẬP

**Câu 8**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class myClass {
2.      private $myNumber = NULL;
3.      public function___construct() {
4.          global $myNumber;
5.$this->myNumber = &$myNumber;  6.
        }

7.      public function foo() {
8.          echo $this->myNumber;
9.      }
10. }
11. $obj =new myClass();
12. $myNumber = 123;
13. $obj->foo();
?>
```

A. Đoạn lệnh sai ở dòng số 4
B. Đoạn lệnh sai ở dòng số 5
C. Đoạn lệnh sai ở dòng số 12
D. Đoạn lệnh xuất kết quả 123

# PHẦN BÀI TẬP

**Câu 9**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  $type = 'cc';
2.  $obj = new $type;
3.  class cc {
4.      function__construct() {
5.echo 'hi';  6.
        }
7.  }
?>
```

A. Đoạn lệnh xuất kết quả "Hi" Đoạn
B. lệnh sai ở dòng số 1 Đoạn lệnh
C. sai ở dòng số 2
D. Đoạn lệnh sai ở dòng số 1 và dòng số 2

# PHẦN BÀI TẬP

**Câu 10**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
1.  <?php class p {
2.              function p() {
3.                  print "1";
4.              }
5.              function p_test() {
6.                  print "2";
7.                  $this->c_test();
8.              }
9.          }
10.         class c extends p {
11.             function c_test() {
12.                 print "3";
13.             }
14.         }
15.         $obj = new c;
16.         $obj->p_test(); ?>
```

A. Đoạn lệnh sai ở dòng số 7
B. Đoạn lệnh xuất kết quả 2
C. Đoạn lệnh xuất kết quả 3
D. Đoạn lệnh xuất kết quả 123

# PHẦN BÀI TẬP

**Câu 11**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

❖ Tập tin cMyMath.php

```
<?php
1.   class mymath {
2.function mysqrt($a) { return $a*$a;}  3.
     }
?>
```

❖ Tập tin Test_cMyMath.php

```
<?php
1.        include("cMyMath.php ");
2.        $obj = new mymath;
3.        echo mysqrt(3);
?>
```

A. Đoạn lệnh xuất kết quả 9

B. Đoạn lệnh xuất kết quả null

C. Đoạn lệnh sai ở dòng số 3 trong tập tin cMyMath.php

D. Đoạn lệnh sai ở dòng số 3 trong tập tin Test_cMyMath.php

# PHẦN BÀI TẬP

**Câu 12**: Chọn phát biểu đúng thay cho từ key1 và key2 để đoạn lệnh sau xuất ra kết quả là 9 (đoạn lệnh thực hiện trong PHP5):

❖ Tập tin cMyMath.php

```
<?php
1.  class mymath {
2.key1 function mysqrt($a) { return $a*$a;}  3.
    }
?>
```

❖ Tập tin Test_cMyMath.php

```
<?php
1.      include("cMyMath.php ");
2.      $obj = new mymath;
3.      key2
?>
```

A. key1: public và key2: echo $obj->mysqrt(3);

B. key1: để trống và key2: echo $obj->mysqrt(3);

C. Cả A và B đều đúng

D. Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 13**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

- Tập tin cTest.php

```
<?php
1.  class test {
2.function myprint() {echo "Hello";}  3.
    }
?>
```

- Tập tin Test_cTest.php

```
<?php
1.  include("cTest.php");
2.  class test{
3.       function myprint() {echo "Good bye";}
4.  }
5.  $obj = new test;
6.  echo $obj->myprint();
?>
```

A. Đoạn lệnh xuất kết quả "Hello" Đoạn lệnh
B. xuất kết quả "Good bye"
C. Đoạn lệnh không xuất kết quả nào hết Đoạn
D. lệnh báo lỗi: Fatal error

# PHẦN BÀI TẬP

**Câu 14**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

- Tập tin cTest.php

```
<?php
1.  class test {
2.function myprint() {echo "Hello";}  3.
    }
?>
```

- Tập tin Test_cTest.php

```
<?php
1.  for ($i=0;$i<3;$i++){
2.        require("cTest.php");
3.        $obj = new  test;
4.$obj->myprint();  5.
    }
?>
```

A. Đoạn lệnh xuất kết quả "Hello" và báo lỗi Fatal Error
B. Đoạn lệnh xuất kết quả "HelloHelloHello"
C. Đoạn lệnh không xuất kết quả nào hết
D. Đoạn lệnh báo lỗi: Fatal error

# PHẦN BÀI TẬP

**Câu 15**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

❖ Tập tin cTest.php

```
<?php
1.  echo "Hello";
?>
```

❖ Tập tin Test_cTest.php

```
<?php
1.  require("cTest.php");
2.  include("cTest.php");
?>
```

A. Đoạn lệnh xuất kết quả "Hello"
B. Đoạn lệnh xuất kết quả "HelloHello"
C. Đoạn lệnh báo lỗi: Fatal error
D. Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 16**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.  class Foo{
2.    public $x;
3.    public function Foo(){$this->x=1;}
4.public function Foo($val){$x=$val;}  5.
    }
6.  $obj = new Foo(3);
7.echo $obj->x;
8.  ?>
```

A. Đoạn lệnh xuất kết quả 1
B. Đoạn lệnh xuất kết quả 3
C. Đoạn lệnh không xuất kết quả nào hết
D. Đoạn lệnh báo lỗi: Fatal error

# PHẦN BÀI TẬP

**Câu 17**: Chọn phát biểu đúng thay cho từ key để đoạn lệnh sau xuất ra kết quả là animal (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.      class Animal{
2.              public $type;
3.              public function Animal(){$this->type="animal";}
4.function getDescription(){return $type;}  5.          }
6.      class Cow extends Animal{
7.private $breed;  8.};
9.      $obj = new Cow();
10.     key
?>
```

A.

B.

C. echo $obj->type;

D. echo $obj::type;

Cả A và B đều đúng

Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 18**: Chọn phát biểu đúng thay cho từ key để đoạn lệnh sau xuất ra kết quả là vehicle (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.      class Vehicle{
2.          protected $type;
3.function Vehicle(){$this->type="vehicle";}  4.
        }
5.      class Car extends Vehicle{
6.          function__construct(){
7.              key
8.              echo $this->type;
9.          }
10.     };
11.     $obj = new Car();
?>
```

A. $type = "vehicle";
B. global $type = "vehicle";
C. parent::Vehicle();
D. Cả A và B đều đúng

# PHẦN BÀI TẬP

**Câu 19**: Cho biết kết quả đoạn lệnh (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.      abstract class AbstractClass{
2.          abstract protected function getValue();
3.public function printOut() {print $this->getValue() . "\n";}  4.
        }
5.      class ConcreteClass1 extends AbstractClass{
6.protected function getValue() {return "ConcreteClass1";}  7.
        }
8.      class ConcreteClass2 extends AbstractClass{
9.          public function getValue() {return "ConcreteClass2";}
10.     }
11.     $class1 = new ConcreteClass1; $class1->printOut();
12.     $class2 = new ConcreteClass2; $class2->printOut();
?>
```

A. ConcreteClass1 ConcreteClass2

B. Đoạn lệnh báo lỗi dòng 3 vì trong lớp abstract chỉ chứa phương thức abstract

C. Đoạn lệnh báo lỗi dòng 3 vì trong lớp abstract phương thức printOut() không được định nghĩa trước.

D. Tất cả đều sai

# PHẦN BÀI TẬP

**Câu 20**: Chọn phát biểu đúng (đoạn lệnh thực hiện trong PHP5):

```
<?php
1.      abstract class newClass{
2.final public function_____construct(){echo 'You failed';}
3.      }
4.      class MyClass extends newClass{
5.        public function___construct(){echo 'Will this work?';}
6.public static function myFunc(){echo 'FooBar';}  7.
        }
8.      $init = new MyClass();
?>
```

A. Đoạn lệnh biên dịch bị lỗi ở dòng số 2
B. Đoạn lệnh biên dịch bị lỗi ở dòng số 6
C. Đoạn lệnh biên dịch thành công nhưng thực thi báo lỗi Fatal error Đoạn lệnh
D. biên dịch thành công nhưng thực thi thông báo warning