FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

THỰC HỌC – THỰC NGHIỆP

LẬP TRÌNH PHP2

PHP OOP 1

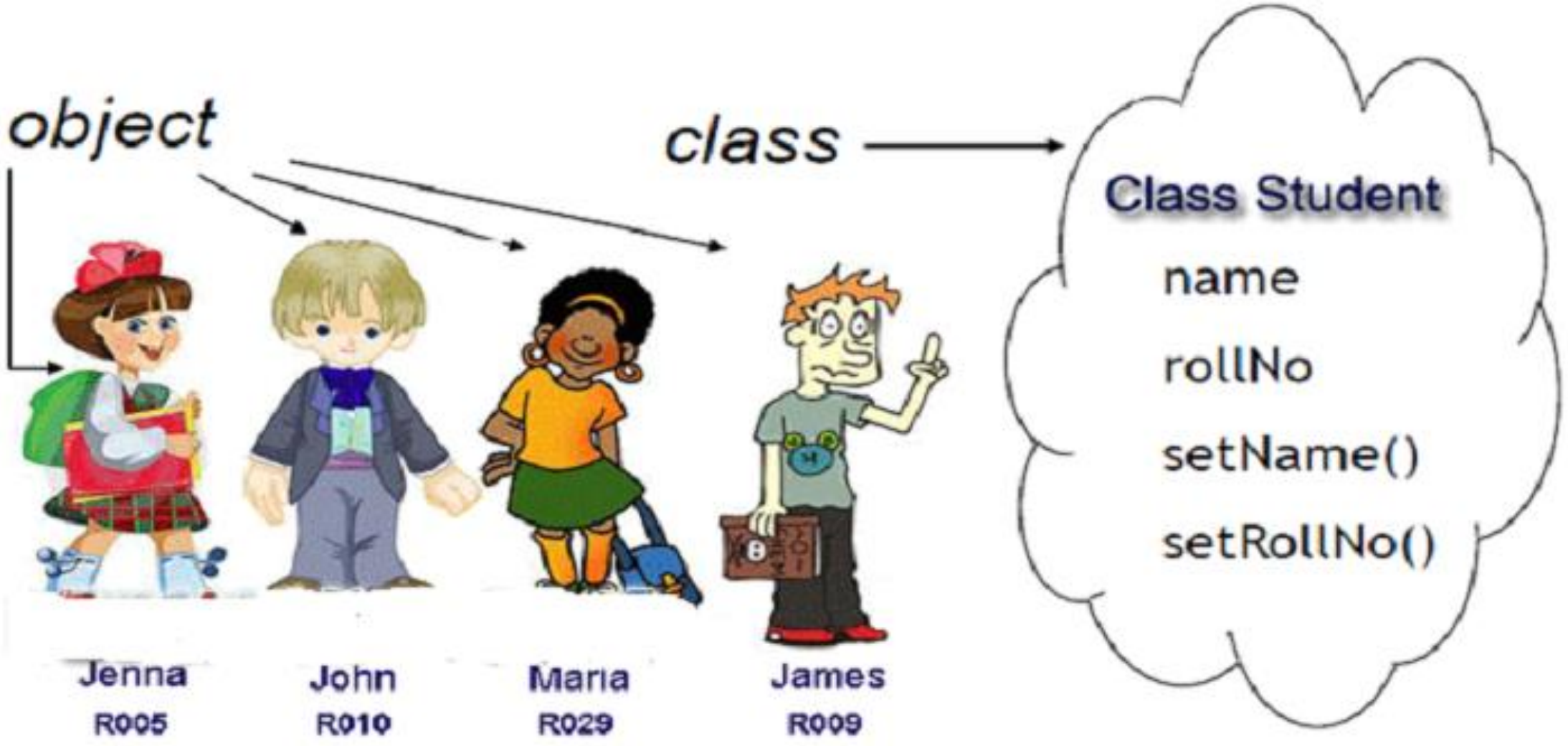
http://www.poly.edu.vn

Mục tiêu

- Class & Object
- Namespace
- Autoloading(prs-4) & composer

PHẦN 1

CLASS & OBJECT



- Object
  - là một kiểu dữ liệu phức hợp.
  - giá trị của nhiều loại có thể được lưu trữ cùng nhau trong một biến duy nhất

```php
<?php
$arr=array("name"=>"Deepak", "age"=>21, "marks"=>75);
$obj=(object)$arr;
print_r($obj);
?>
```

Output

```
stdClass Object(
    [name] => Deepak
    [age] => 21
    [marks] => 75
)
```

CLASS

- ***Object***: là một *instance* của *class* được tạo sẵn do người dùng xác định.
- ***Class***: là *template* được sử dụng để tạo *object*.
  - Tạo class: sử dụng từ khoá ***class***.

```
class MyRectangle {}
```

  - Thân class chứa ***properties*** và ***methods***.
  - Properties: là các biến giữ state (trạng thái) của object.
  - Methods: là các function xác định những gì đối tượng có thể làm.

- Trong PHP, các properties và methods cần có cấp độ truy cập (**access levels** *) rõ ràng

```
class MyRectangle
{
  public $x, $y;
  function newArea($a, $b) { return $a * $b; }
}
```

- public ⇔ var

- Truy xuất các thành phần (member) trong class: dùng biến giả (pseudo) `$this` tham chiếu đến instance hiện tại của class.

```
class MyRectangle
{
  public $x, $y;

  function newArea($a, $b) {
    return $a * $b;
  }

  function getArea() {
    return $this->newArea($this->x, $this->y);
  }
}
```

Thể hiện một đối tượng

- Sử dụng các thành phần của class từ bên ngoài class: tạo một đối tượng của class bằng cách sử dụng từ khoá **new**

```
$r = new MyRectangle(); // object instantiated
```

- Truv xuất các thành phần

```
$r->x = 5;
$r->y = 10;
$r->getArea(); // 50
```

- Cài đặt giá trị của property (thuộc tính)

```
class MyRectangle
{
  public $x = 5, $y = 10;
}
```

CLASS - CONSTRUCTOR

- Trong class có thể có ***constructor*** (phương thức khởi tạo)
- ***Constructor***:
  - phương thức đặc biệt được sử dụng để khởi tạo đối tượng.
  - Cung cấp cách để khởi tạo attribute, không giới hạn constant expression.
  - Là một *magic method*

```php
<?php

class MyRectangle
{
    public $x, $y;
    function __construct()
    {
        $this->x = 5;
        $this->y = 10;
        echo "Constructed";
    }
}

?>
```

CLASS - CONSTRUCTOR

- Khi một new instance của class được tạo > constructor được gọi

```
$r = new MyRectangle(); // "Constructed"
```

- Constructor không có đối số

```
$r = new MyRectangle; // "Constructed"
```

- Constructor có thể chứa các tham số.

```
class MyRectangle
{
  public $x, $y;

  function __construct($x, $y)
  {
    $this->x = $x;
    $this->y = $y;
  }
}

$r = new MyRectangle(5,10);
```

CLASS - CONSTRUCTOR

- PHP8: danh sách tham số trong constructor.

```
class MyRectangle
{
  function __construct(public $x, public $y)
  {
    $this->x = $x;
    $this->y = $y;
  }
}
```

CLASS - CONSTRUCTOR

- *Tham số mặc định* (default parameter) trong constructor.

```
class MyRectangle
{
  public $x;
  function __construct($x = 5, public $y = 10)
  {
    $this->x = $x;
    $this->y = $y;
  }
}
$r = new MyRectangle;
echo $r->x + $r->y; // "15"
```

CLASS - DESTRUCTOR

- ***Destructor*** được khi ngay sau khi không còn tham chiếu đến object.

```
class MyRectangle
{
  // ...
  function __destruct() { echo "Destructed"; }
}

$r = new MyRectangle;
unset($r); // "Destructed"
```

- ***Case sensitivity*** (phân biệt hoa thường).
  - Không phân biệt chữ hoa chữ thường

```
class MyClass {}
$o1 = new myclass(); // ok
$o2 = new MYCLASS(); // ok
```

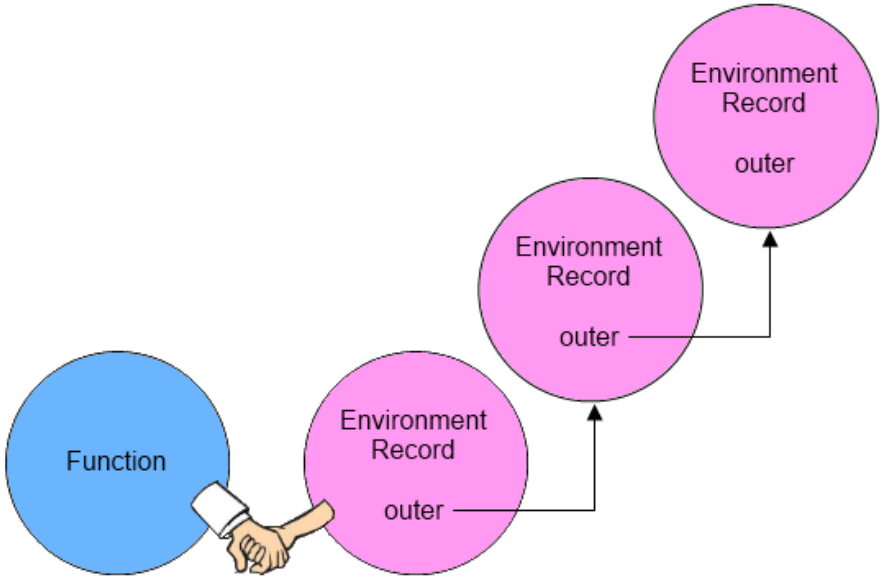
  - So sánh object

```
class Flag
{
  public $flag = true;
}

$a = new Flag();
$b = new Flag();

$c = ($a == $b);  // true (same values)
$d = ($a === $b); // false (different instances)
```

- ***Anonymous class*** (lớp ẩn danh): sử dụng khi 1 class chỉ cần một object đơn lẻ.

```php
$o = new class('Anonymous class')
{
    public $x;
    public function __construct($a)
    {
      $this->x = $a;
    }
};
echo $o->x; // "Anonymous class";
```

- ***Closure object***: Truy xuất biến ngoài object
- `bindTo(newThis, newScope)`
  - newThis: đối tượng mà anonymous function được ràng buộc để closure không bị ràng buộc.
  - newScope: phạm vi lớp mà closure được liên kết.




```
class C { private $x = 'Hi'; }

$getC = function() { return $this->x; };
$getX = $getC->bindTo(new C, 'C');
echo $getX(); // "Hi"
```

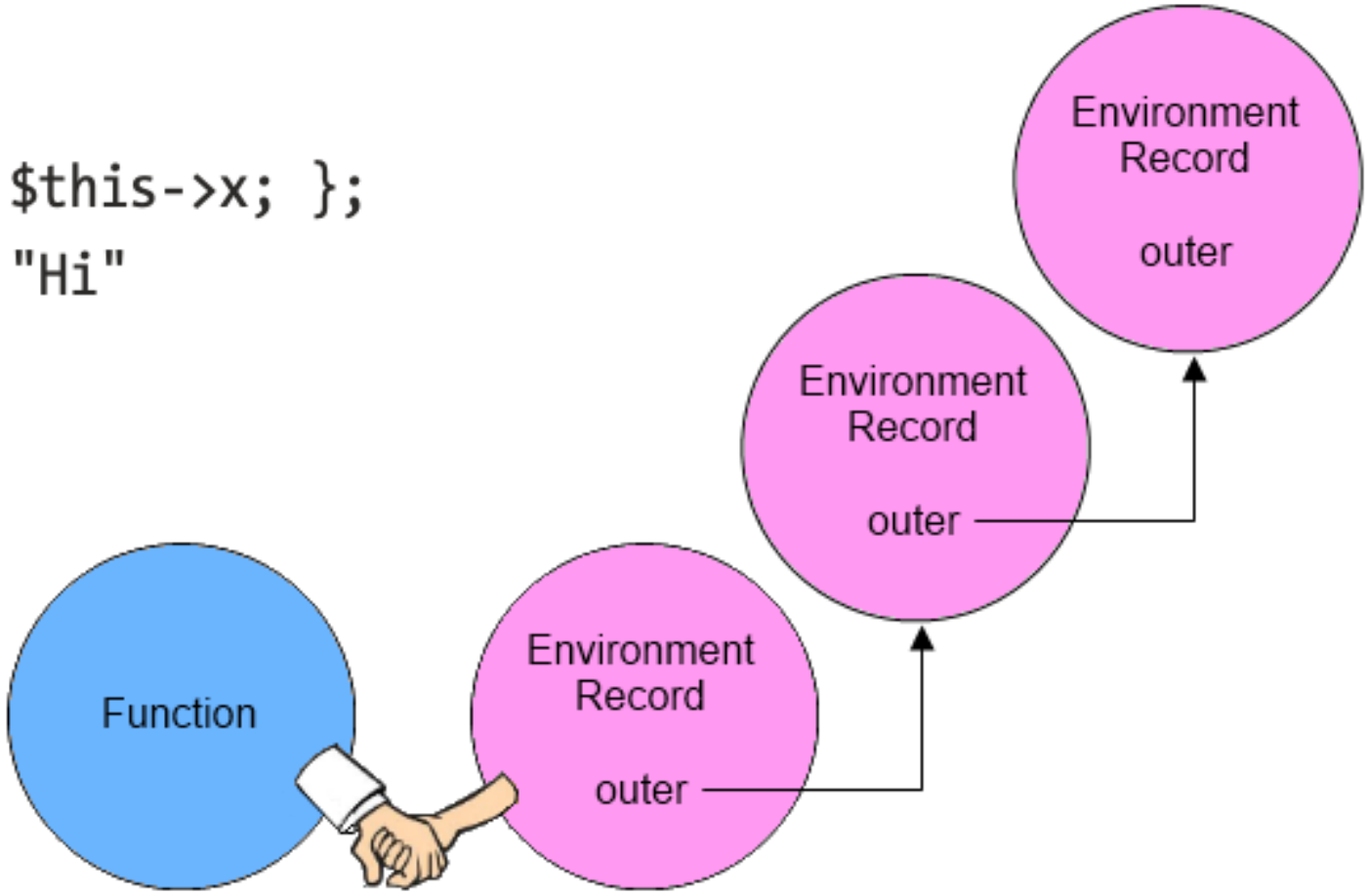
Class name

First closure: `$getC` định nghĩa method truy xuất property x ngoài class C

Second closure: `$getX` là nhân bản của `$getC`

Closure object: trong PHP7

```
// PHP 7
$getX = function() { return $this->x; };
echo $getX->call(new C); // "Hi"
```




demo

PHẦN 2

NAMESPACE

- Nhóm các member thành một hệ thống phân cấp.
- 4 cấu trúc mã bị ảnh hưởng bởi namespace: class, interfaces, function và constants.
- Bất kỳ cấu trúc nào bên dưới lệnh namespace đều thuộc namespace đó.
- Tạo ***namespace***

```
// Global code/namespace
class MyClass {}
```

- Ví dụ

```
namespace my;

// Belongs to my namespace
class MyClass {}
```

- Tập tin chứa namespace phải khai báo namespace ở đầu tập tin.

```
<?php
namespace my;
class MyClass {}
?>
<html><body></body></html>
```

- ***Namespace lồng nhau***:

```
namespace my\sub;
class MyClass {} // my\sub\MyClass
```

- ***Alternative Syntax***: class có thể nằm trong cặp ngoặc nhọn của namespace

```
<?php
namespace my
{
  class MyClass {}
?>
<html><body></body></html>
<?php }?>
```

- ***Referencing namespace***: fully qualified, qualified và unqualified.

```php
namespace my
{
  class MyClass {}
}

namespace other
{
  // Fully qualified name
  $obj = new \my\MyClass();
}
```

```php
namespace my
{
  class MyClass {}
}

namespace
{
  // Qualified name
  $obj = new my\MyClass();
}
```

```php
namespace my
{
  class MyClass {}

  // Unqualified name
  $obj = new MyClass();
}
```

- ***Namespace alias***: tên cho class, interface và namespace có thể được rút gọn

```
namespace my;
class MyClass {}

namespace foo;
use my\MyClass as MyAlias;
$obj = new MyAlias();
```

- PHP7: import nhiều members trong cùng 1 lệnh

```
namespace foo;
use my\Class1 as C1, my\Class2 as C2;

namespace foo;
use my\{ Class1 as C1, Class2 as C2 };
```

- ***Namespace alias***: hỗ trợ cả function và const construct

```
namespace my\space {
  const C = 5;
  function f() {}
}

namespace {
  use const my\space\C;
  use function my\space\f;
}
```


demo

PHẦN 3

- ***sp_autoload_register()***: hàm trợ giúp tự động hóa việc gọi thư viện.

index.php × Database.php

mvc_simple > index.php > html

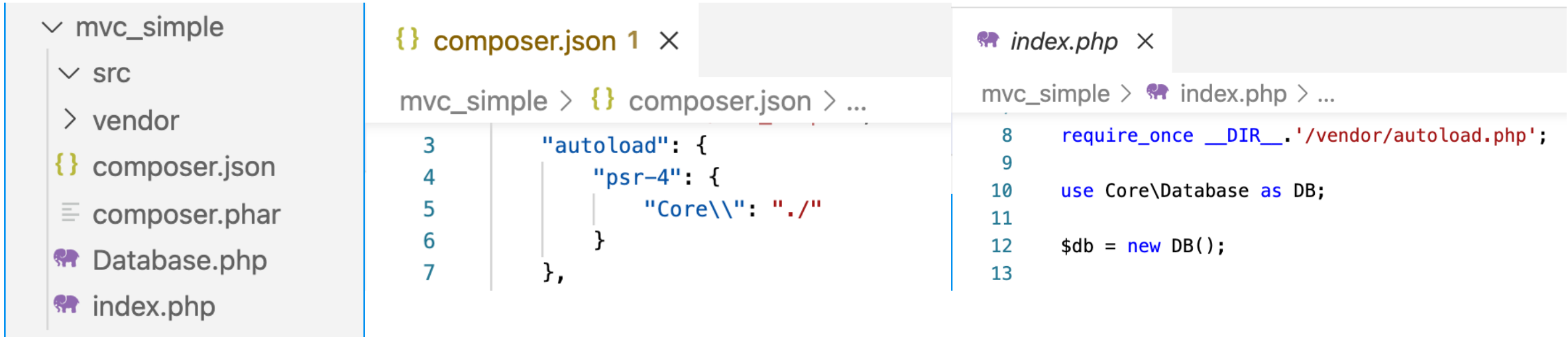
```php
<?php
//require 'Database.php';
spl_autoload_register(function($class){
    var_dump($class);
});

use \App\Database as DB;

$db = new DB();
?>
```

PSR-4 & COMPOSER

- Tiêu chuẩn tự động load **PSR-4** : Quy tắc tổ chức các thư mục code sao cho mọi class (*bao gồm class, interface, trait*) đều có thể được tham chiếu đến bằng cách viết mã như sau:

```
\<NamespaceName>(\<SubNamespaceNames>)*\<ClassName>
```

- Composer



SUMARRY

- ☑ Class & Object
- ☑ Namespace
- ☑ Autoloading(prs-4) & composer


thank
you!