

# Lập trình hướng đối tượng trong PHP

## Bài 03

# <sup>2</sup>Nội dung

- Khái niệm OOP
- Tạo class, attributes, và operations trong PHP
- Sử dụng lớp
- Truy cập thuộc tính, phương thức
- Phạm vi truy cập private và public

# Tạo class, attribute, và operation trong PHP

- Cấu trúc class (lớp) trong PHP

Ví dụ 1

```
class classname {  
  
}
```

Ví dụ 2

```
class Person {  
  
}
```

# Tạo class, attribute, và operation trong PHP

- Tạo class với các thuộc tính (attribute)

Ví dụ 1

```
class classname {  
    public $attribute1;  
    public $attribute2;  
}
```

Ví dụ 2

```
class Person {  
    public $name = '';  
    public $age ;  
}
```

# Tạo class, attribute, và operation trong PHP

- Tạo class với các thuộc tính (attribute), phương thức

## Ví dụ 1

```
class classname {  
    public $attribute1;  
    public $attribute2;  
    function operation1(){  
  
    }  
  
    function operation2($param1, $param2){  
  
    }  
}
```

## Ví dụ 2

```
class Person {  
    public $name = '';  
    public $age ;  
  
    function getName() {  
        return $this->name;  
    }  
  
    function setName($newName) {  
        $this->name = $newName;  
    }  
    // attribute $age???  
}
```

# Tạo class, attribute, và operation trong PHP

- Khai báo phương thức với từ khóa static

```
class HTMLStuff {  
    static function startTable() {  
        echo "<table border=\"1\">\n";  
    }  
    static function endTable() {  
        echo "</table>\n";  
    }  
}
```

```
HTMLStuff::startTable();  
// print HTML table rows and columns  
HTMLStuff::endTable();
```

# Tạo class, attribute, và operation trong PHP

- Hàm khởi tạo `__construct`

```
class classname {  
    function __construct($param) {  
        echo "Constructor called with parameter ".$param."<br  
>";  
    }  
}
```

- Instance class

```
$a = new classname("First");  
$b = new classname("Second");
```

## Kết quả

Constructor called with parameter First

Constructor called with parameter Second

# Tạo class, attribute, và operation trong PHP

- Sử dụng các thuộc tính classes

```
class classname {  
    public $attribute;  
    function operation($param) {  
        $this->attribute = $param;  
        echo $this->attribute;  
    }  
}
```

```
class classname {  
    public $attribute;  
}  
  
$a = new classname();  
$a->attribute = "value";  
echo $a->attribute;
```



# Tạo class, attribute, và operation trong PHP

- Gọi các phương thức của classes

```
class classname {  
    function operation1() {  
    }  
  
    function operation2($param1, $param2){  
    }  
}
```

- Tạo đối tượng classname
- Truy cập các thuộc tính của đối tượng
- Nếu phương thức có giá trị trả về → lấy giá trị trả về và gán vào \$biến

```
$a = new classname();  
$a->operation1();  
$a->operation2(12, "test");  
  
$x = $a->operation1();  
$y = $a->operation2(12, "test");
```

# Tạo class, attribute, và operation trong PHP

- Khai báo phương thức với từ khóa:
  - public
  - protected
  - private

```
class Person {  
    public $age;  
    public function __construct() {  
        $this->age = 0;  
    }  
    public function incrementAge() {  
        $this->age += 1;  
        $this->ageChanged();  
    }  
    protected function decrementAge() {  
        $this->age -= 1;  
        $this->ageChanged();  
    }  
    private function ageChanged() {  
        echo "Age changed to {$this->age}";  
    }  
}
```

# <sup>11</sup>Tạo class, attribute, và operation trong PHP

- Phạm vi truy cập: public, protected, private

```
$person = new Person;  
$person->incrementAge();  
$person->decrementAge(); // not allowed  
$person->ageChanged(); // also not allowed
```

# <sup>12</sup>Khai báo hằng số

- Khai báo hằng số
  - Truy cập trực tiếp thông qua lớp
  - Truy cập trong phương thức đối tượng thông qua từ khóa self

```
class PaymentMethod {  
    public const TYPE_CREDITCARD = 0;  
    public const TYPE_CASH = 1;  
}  
echo PaymentMethod::TYPE_CREDITCARD;
```

# Ví dụ: Xây dựng lớp phân số

```
class PhanSo {  
    // khai báo thuộc tính  
    private $tuSo;  
    private $mauSo;  
    // xây dựng các phương thức cho lớp  
  
}
```

# Ví dụ: Xây dựng lớp phân số

```
class PhanSo {  
    // khai báo thuộc tính  
    private $tuSo;  
    private $mauSo;  
    // xây dựng các phương thức cho lớp  
    // hàm khởi tạo  
    public function __construct($tuSo, $mauSo) {  
        if ($mauSo == 0) {  
            throw new Exception("Mẫu số phải khác 0.");  
        }  
        $this->tuSo = $tuSo;  
        $this->mauSo = $mauSo;  
    }  
}
```

# Ví dụ: Xây dựng lớp phân số

```
class PhanSo {  
    //....  
    // hàm in phân số  
    public function inPhanSo() {  
        return $this->tuSo . '/' . $this->mauSo;  
    }  
    // hàm tìm ước số chung lớn nhất  
    public function USCLN($a, $b){  
        $so_nho = ($a<$b)?$a:$b;  
        for ($i = $so_nho; $i>0; $i--) {  
            if(($a%$i == 0) && ($b%$i == 0)){  
                return $i;  
                break;  
            }  
        }  
    }  
}
```

# Ví dụ: Xây dựng lớp phân số

```
<?php
class PhanSo {
    //....
    //hàm tối giản phân số
    public function toiGianPhanSo() {
        $uscln = $this->USCLN($this->tuSo, $this->mauSo);
        $this->tuSo /= $uscln;
        $this->mauSo /= $uscln;
    }
    // hàm tính tổng hai phân số

    public function congPhanSo(PhanSo $ps) {
        $tuSoMoi = $this->tuSo * $ps->mauSo + $ps->tuSo * $this->mauSo;
        $mauSoMoi = $this->mauSo * $ps->mauSo;
        return new PhanSo($tuSoMoi, $mauSoMoi);
    }
}
?>
```



# <sup>17</sup>Ví dụ: Xây dựng lớp phân số

Tiếp tục xây dựng thêm các phương thức

Hiệu hai phân số

Tích hai phân số

Chia phân số

# Ví dụ: Sử dụng lớp đối tượng

```
// Khởi tạo đối tượng
$ps = new PhanSo(2, 4);
// Gọi phương thức của lớp
echo "Phân số : ". $ps->inPhanSo(). "<br/>";
$ps -> toiGianPhanSo();
echo "Phân số tối giản : ". $ps->inPhanSo(). "<br/>";

// Khởi tạo đối tượng
$ps1 = new PhanSo(3, 4);
$ps2 = new PhanSo(2, 5);

echo "Phân số 1: ". $ps1->inPhanSo(). "<br/>";
echo "Phân số 2: ". $ps2->inPhanSo(). "<br/>";

$psTong = $ps1->congPhanSo($ps2);
echo "Tổng: " . $psTong->inPhanSo(). "<br/>"; ;
```

# 19 Kế thừa

```
class A {  
    public $attribute1;  
    function operation1(){  
  
    }  
}
```

```
class B extends A {  
    public $attribute2;  
    function operation2(){  
  
    }  
}
```

**Tạo đối tượng và gọi phương thức**

```
$b = new B();  
$b -> operation1();
```

```
$b -> attribute1 = 10;  
$b -> operation2();  
$b -> attribute2 = 10;
```

# Kế thừa

Tạo đối tượng và truy cập phương thức ?

```
class Person {  
    public $name, $address, $age;  
}  
  
class Employee extends Person {  
    public $position, $salary;  
}
```

# Kế thừa : private, protected keyword

```
class A {  
    private function operation1(){  
        echo "operation1 called";  
    }  
    protected function operation2(){  
        echo "operation2 called";  
    }  
    public function operation3(){  
        echo "operation3 called";  
    }  
}
```

```
class B extends A {  
    function __construct() {  
        $this->operation1();// error  
        $this->operation2();  
        $this->operation3();  
    }  
}  
  
$b = new B;
```

`$this -> operation1();` // **Fatal error:** Call to private method A::operation1() from context 'B'

`$b -> operation2();` // **Fatal error:** Call to protected method A::operation2() from context ''

`$b -> operation3();` // OK

# Kế thừa - Overriding

```
class A {  
    public $attribute = 'default value';  
  
    function operation(){  
        echo 'Something<br />';  
        echo 'The value of $attribute is '. $this->attribute.'<br />';  
    }  
}
```

```
class B extends A {  
    public $attribute = 'different value';  
    function operation(){  
        echo 'Something else<br />';  
        echo 'The value of $attribute is '. $this->attribute.'<br />';  
    }  
}
```

# <sup>23</sup>Kế thừa - Overriding

- Tạo đối tượng vào gọi phương thức

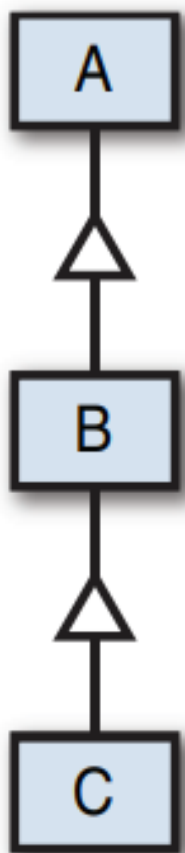
```
$b = new B();  
$b->operation();
```

- Kết quả

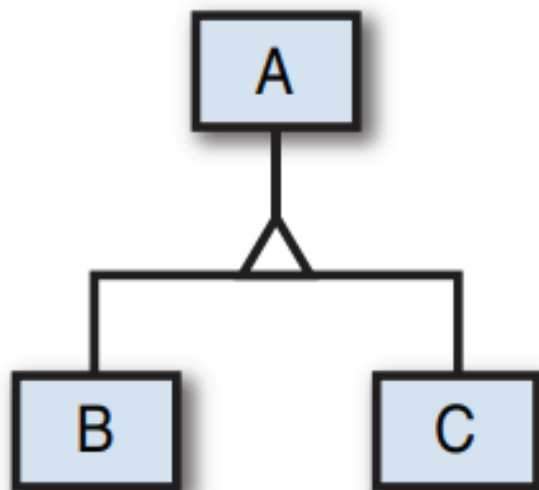
Something else

The value of \$attribute is different value

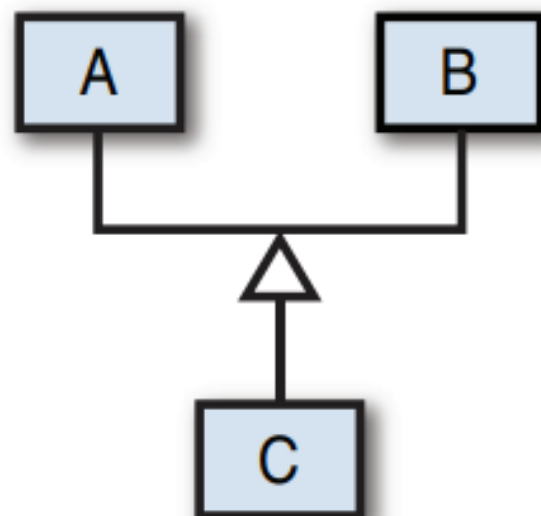
# Đa kế thừa



Single Inheritance



Single Inheritance



Multiple Inheritance



# 25 PHP Traits

- Traits là nhóm các phương thức để tích hợp nó vào lớp khác
- Cú pháp:

```
trait traitname [ extends baseclass ] {  
    [ use traitname, [ traitname, ... ]; ]  
  
    [ visibility $property [ = value ]; ... ]  
  
    [ function functionname (args) {  
        // code  
    }  
    ...  
]  
}
```

# PHP Trials (1)

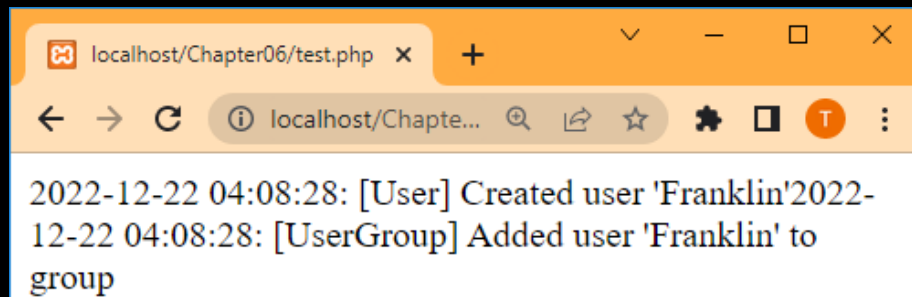
```
trait Logger {  
    public function log($logString) {  
        $className = __CLASS__;  
        echo date("Y-m-d h:i:s", time()) .  
            ": [{$className}] {$logString}";  
    }  
}
```

# 27 PHP Trials (2)

```
class User {  
    use Logger;  
    public $name;  
    function __construct($name = '') {  
        $this->name = $name;  
        $this->log("Created user '{$this->name}'");  
    }  
    function __toString() {  
        return $this->name;  
    }  
}
```

# PHP Trials (3)

```
class UserGroup {  
    use Logger;  
    public $users = array();  
    public function addUser(User $user) {  
        if (!in_array( $user, $this->users)) {  
            $this->users[] = $user;  
            $this->log("Added user '{$user}' to group");  
        }  
    }  
}  
  
$group = new UserGroup;  
$group->addUser(new User("Franklin"));
```



localhost/Chapter06/test.php x +

localhost/Chapte...

2022-12-22 04:08:28: [User] Created user 'Franklin'

2022-12-22 04:08:28: [UserGroup] Added user 'Franklin' to group

# PHP Trial: khai báo nhiều trait

```
trait First {  
    public function doFirst() {  
        echo "first\n";  
    }  
}  
  
trait Second {  
    public function doSecond() {  
        echo "second\n";  
    }  
}
```

```
trait Third {  
    use First, Second;  
    public function doAll() {  
        $this->doFirst();  
        $this->doSecond();  
    }  
}  
  
class Combined {  
    use Third;  
}  
  
$object = new Combined;  
$object->doAll();  
// firstsecond
```

# PHP Trait: khai báo như abstract methods

```
trait Command {  
    function run() {  
        echo "Executing a command\n";  
    }  
}  
trait Marathon {  
    function run() {  
        echo "Running a marathon\n";  
    }  
}
```

# PHP Trial: khai báo như abstract methods

```
class Person {  
    use Command, Marathon {  
        Marathon::run insteadof Command;  
    }  
}  
  
$person = new Person;  
$person->run();  
// Running a marathon
```

# PHP Trial: khai báo như abstract methods

```
class Person {  
    use Command, Marathon {  
        Command::run as runCommand;  
        Marathon::run insteadof Command;  
    }  
}  
$person = new Person;  
$person->run();  
$person->runCommand();  
// Running a marathonExecuting a command
```