FPT Education

FPT POLYTECHNIC

cdio™

Conceive Design Implement Operate

LẬP TRÌNH PHP2

PHP OOP 2

THỰC HỌC – THỰC NGHIỆP

http://www.poly.edu.vn

Mục tiêu

- Inheritance
- Access level & Class constant
- Interface
- Abstract
- Trait

PHẦN 1

INHERRITANCE (KẾ THỪA)

- ***Kế thừa***: cho phép 1 class yêu cầu các thành phần của class khác. Sử dụng từ khoá `extends`.

```
// Parent class (base class)
class Rectangle
{
  public $x, $y;
  function __construct($a, $b)
  {
    $this->x = $a;
    $this->y = $b;
  }
}

// Child class (derived class)
class Square extends Rectangle {}
```

```
$s = new Square(5,10);

$s->x = 5;
$s->y = 10;
```

INHERRITANCE (KẾ THỪA)

- ***Overriding (ghi đè) member***: một thành phần trong lớp con (child class) có thể redefine (định nghĩa lại) một thành phần của lớp cha (parent class).

```
// Parent class (base class)
class Rectangle
{
  public $x, $y;
  function __construct($a, $b)
  {
    $this->x = $a;
    $this->y = $b;
  }
}
```

```
class Square extends Rectangle
{
  function __construct($a)
  {
    $this->x = $a;
    $this->y = $a;
  }
}
```

Overriding member

- ***Overriding (ghi đè) member***: một thành phần trong lớp con (child class) có thể redefine (định nghĩa lại) một thành phần của lớp cha (parent class).

```
// Parent class (base class)
class Rectangle
{
  public $x, $y;
  function __construct($a, $b)
  {
    $this->x = $a;
    $this->y = $b;
  }
}
```

```
class Square extends Rectangle
{
  function __construct($a)
  {
    parent::__construct($a,$a);
  }
}
```

Alias cho class name của Retangle

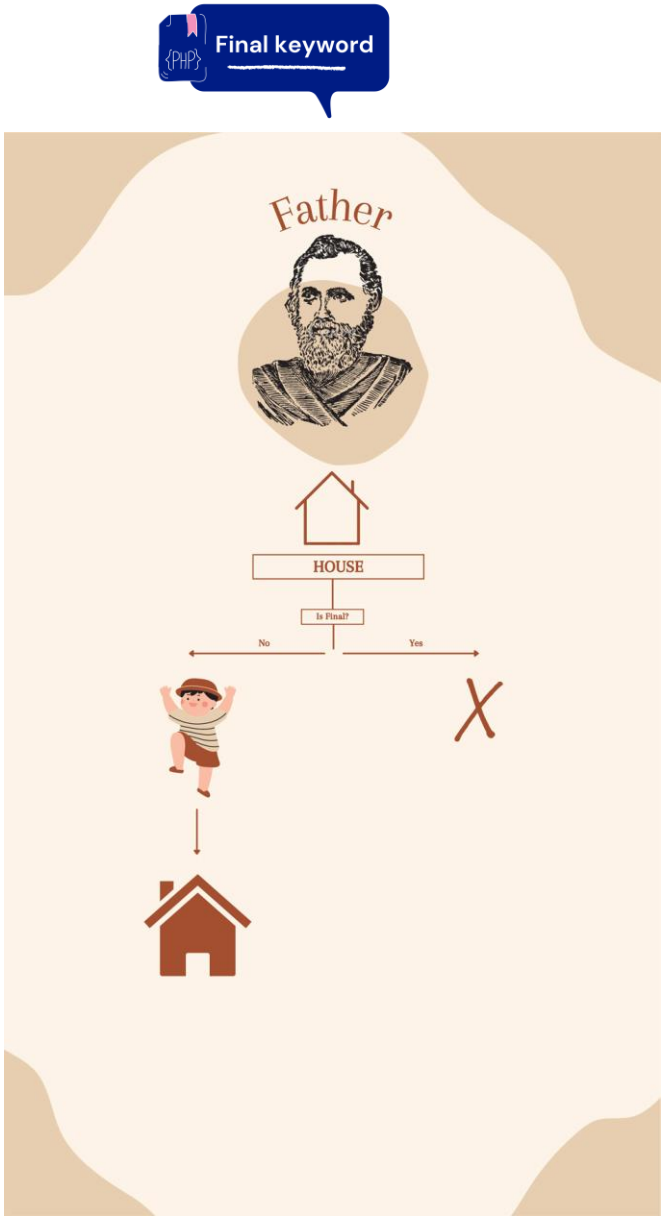
Retangle::__construct

Scope resolution operator

❑ *Final keyword*:

ngăn 1 lớp con overriding (ghi đè) một method (phương thức)

```
final class NotExtendable
{
  final function notOverridable() {}
}
```



- ***Instanceof Operator***: kiểm tra một object có thể được truyền tới một class cụ thể hay không

parent

```
// Parent class (base class)
class Rectangle
{
  public $x, $y;
  function __construct($a, $b)
  {
    $this->x = $a;
    $this->y = $b;
  }
}
```

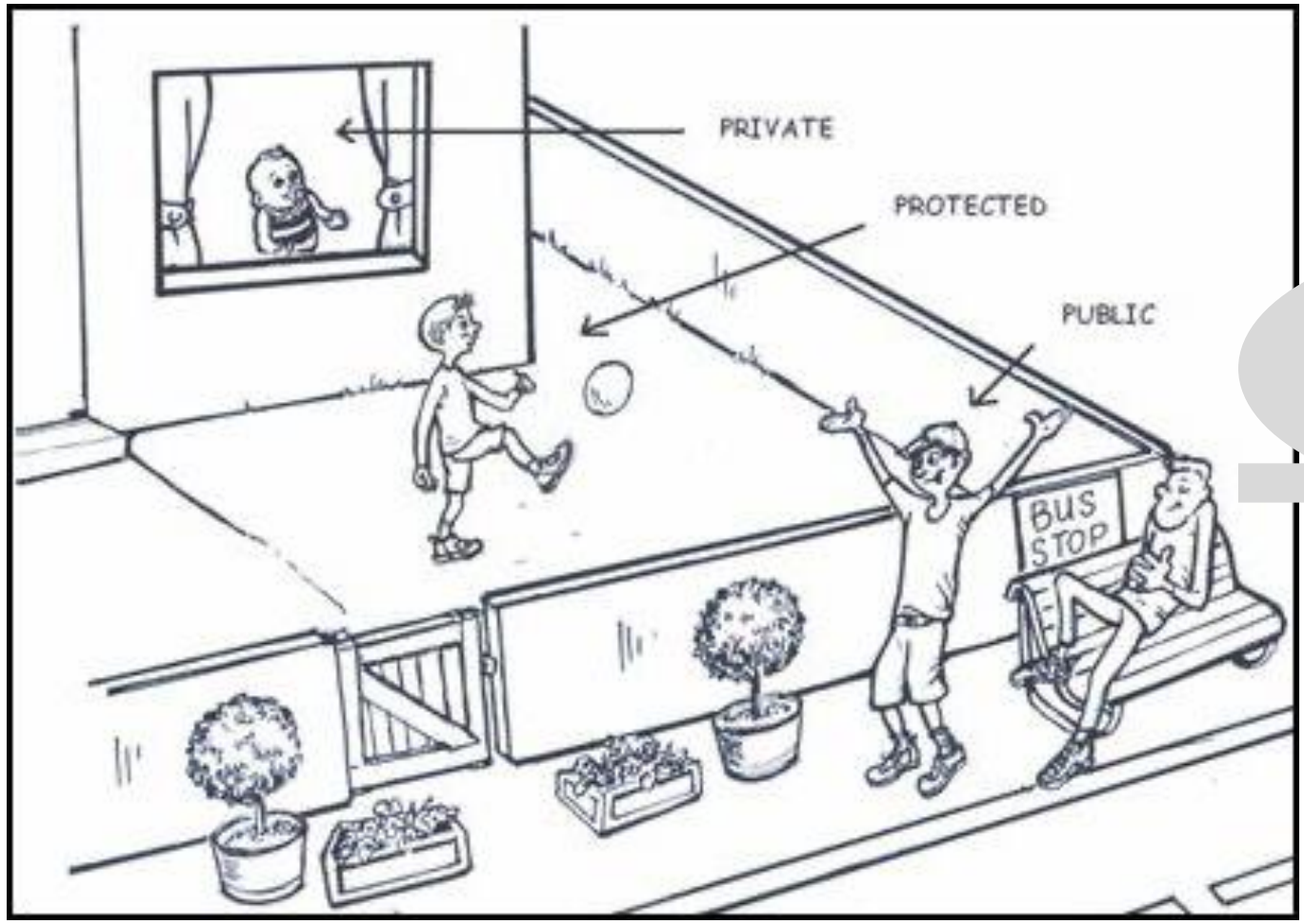
child

```
class Square extends Rectangle
{
  function __construct($a)
  {
    parent::__construct($a,$a);
  }
}
```

```
$s = new Square(5);
$s instanceof Square; // true
$s instanceof Rectangle; // true
```

ACCESS LEVEL (CÁC MỨC TRUY XUẤT)

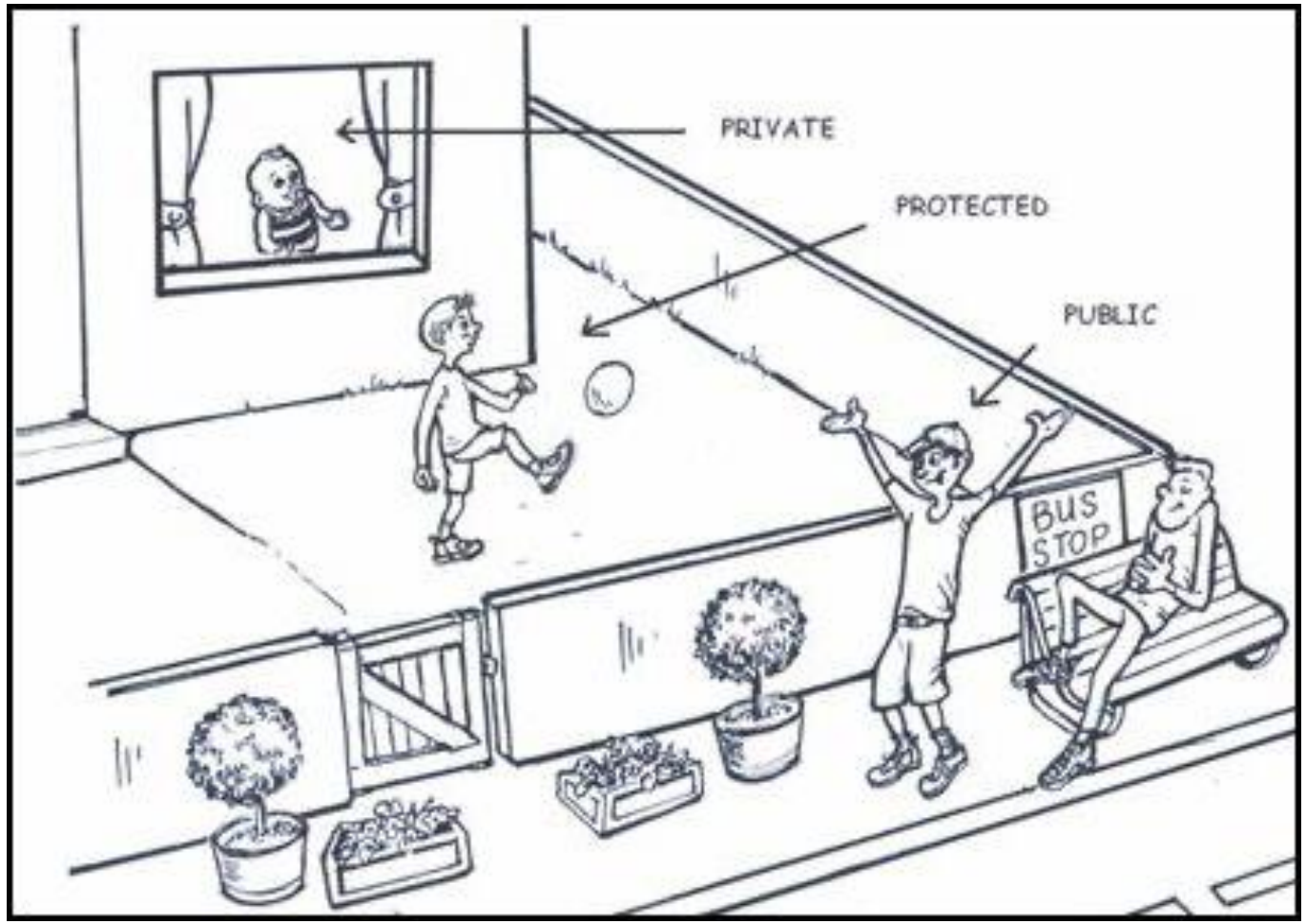
❑ *Public, protected, private*



Var keyword?

ACCESS LEVEL (CÁC MỨC TRUY XUẤT)

❑ *Public, protected, private*




```php
class MyClass
{
    public    $myPublic    = 'public';
    protected $myProtected = 'protected';
    private   $myPrivate   = 'private';
    function test()
    {
        echo $this->myPublic;    // allowed
        echo $this->myProtected; // allowed
        echo $this->myPrivate;   // allowed
    }
}

class MyChild extends MyClass
{
    function test()
    {
        echo $this->myPublic;    // allowed
        echo $this->myProtected; // allowed
        echo $this->myPrivate;   // inaccessible
    }
}

$m = new MyClass();
echo $m->myPublic;    // allowed
echo $m->myProtected; // inaccessible
echo $m->myPrivate;   // inaccessible
```

ACCESS LEVEL (CÁC MỨC TRUY XUẤT)

Get & set

Hạn chế dùng public

```
class Time
{
  private $minutes;

  function getMinutes() {
    return $this->minutes;
  }

  function setMinutes($val) {
    $this->minutes = $val;
  }
}
```

ACCESS LEVEL (CÁC MỨC TRUY XUẤT)

Static

```
class MyCircle
{
  // Instance members (one per object)
  public $r = 10;
  function getArea() {}

  // Static/class members (only one copy)
  static $pi = 3.14;
  static function newArea($a) {}
}
```

```
function getArea()
{
  return self::newArea($this->r);
}
```

Reference static member

Reference static member

```
static function newArea($a)
{
  return self::$pi * $a * $a; // ok
  return MyCircle::$pi * $a * $a; // alternative
}
```

ACCESS LEVEL (CÁC MỨC TRUY XUẤT)

Static

```php
class MyParent
{
  protected static $val = 'parent';
  public static function getVal()
  {
    return self::$val;
  }
}
class MyChild extends MyParent
{
  protected static $val = 'child';
}
echo MyChild::getVal(); // "parent"
```

```php
class MyParent
{
    protected static $val = 'parent';
    public static function getLateBindingVal()
    {
        return static::$val;
    }
}
class MyChild extends MyParent
{
    protected static $val = 'child';
}
echo MyChild::getLateBindingVal(); // "child"
```

Late static binding

CLASS CONSTANT

- ***Const*** modifier: dùng để tạo class constant

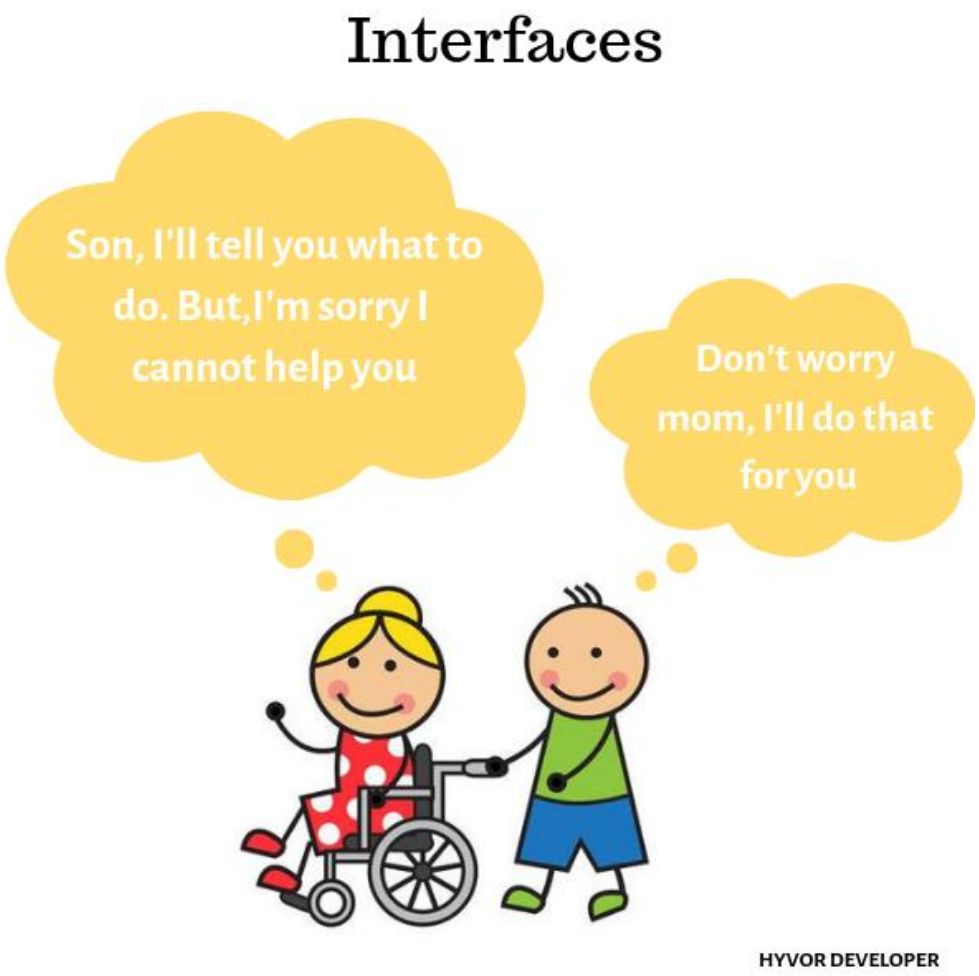
```
class MyCircle
{
  const PI = 3.14;
}

echo MyCircle::PI; // "3.14"
```

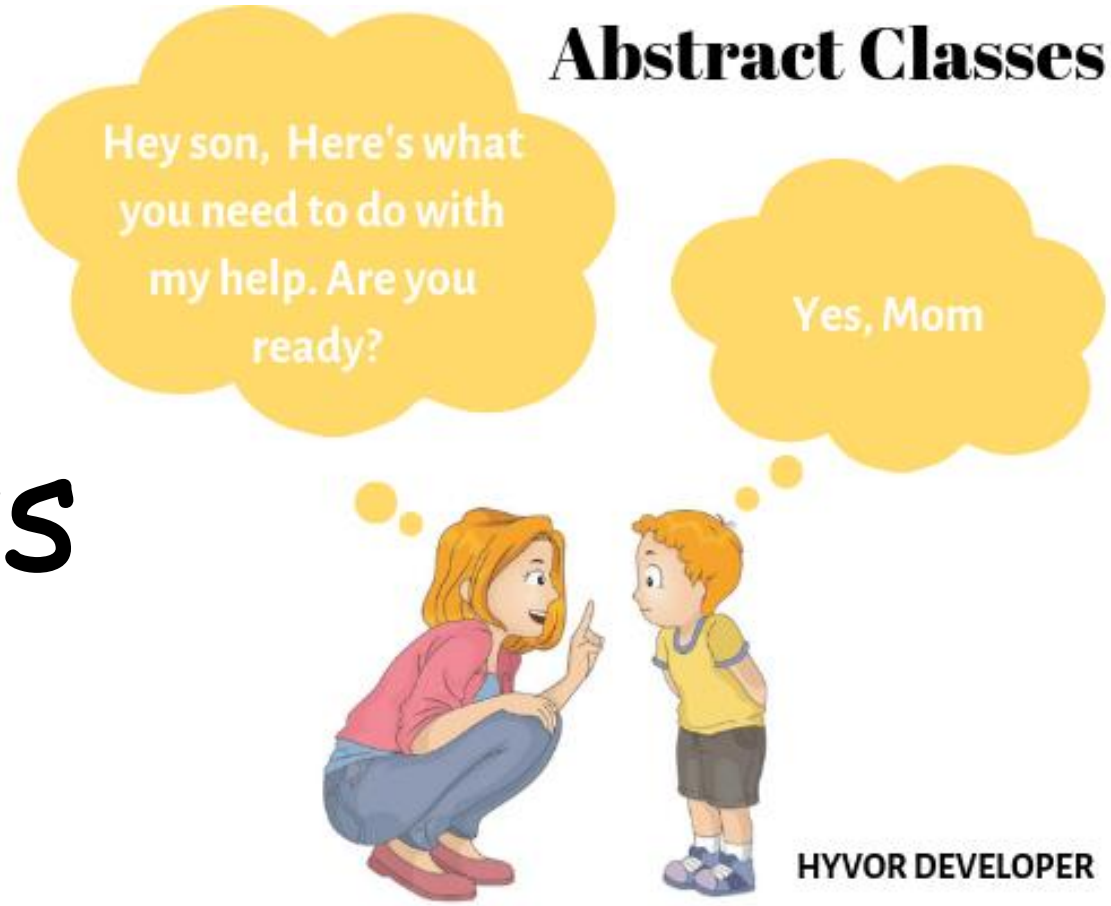
reference

PHẦN 2

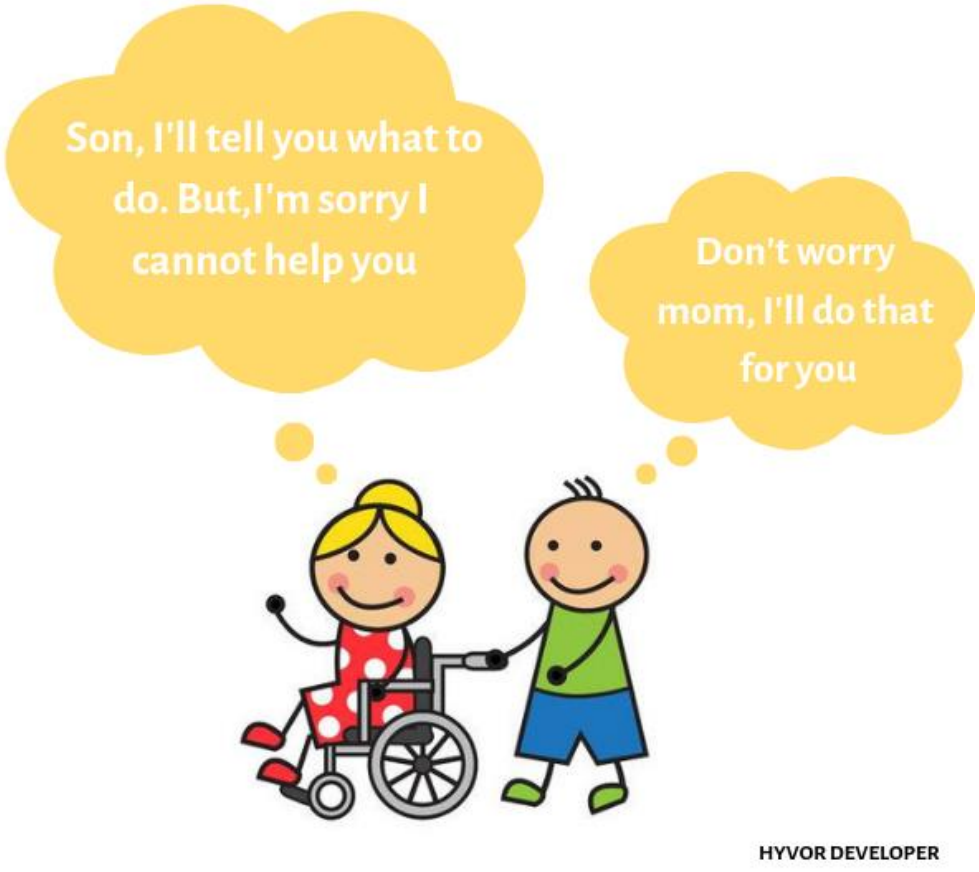
INTERFACE & ABSTRACT



VS



Interfaces



keyword

```
interface iComparable
{
    public function compare(iComparable $o);
}

class Circle implements iComparable
{
    public $r;
    public function compare(iComparable $o)
    {
        return $this->r - $o->r;
    }
}
```

INTERFACE & ABSTRACT

keyword

```
abstract class Shape
{
  private $x = 100, $y = 100;
  abstract public function getArea();
}

class Rectangle extends Shape
{
  public function getArea()
  {
    return $this->x * $this->y;
  }
}
```

- ***Trait***: nhóm các methods được chèn vào class

keyword

```
Trait Hello {
    public function hello() {
        echo "Hello";
    }
}
Trait World {
    public function world() {
        echo "World";
    }
}
class MyClass {
    use Hello, World;
}
$obj = new MyClass();
$obj -> hello();
$obj -> world();
```

Trait & inheritance

```php
trait PrintFunctionality
{
  public function myPrint() { echo 'Hello'; }
}
class MyParent
{
  public function myPrint() { echo 'Base'; }
}
```

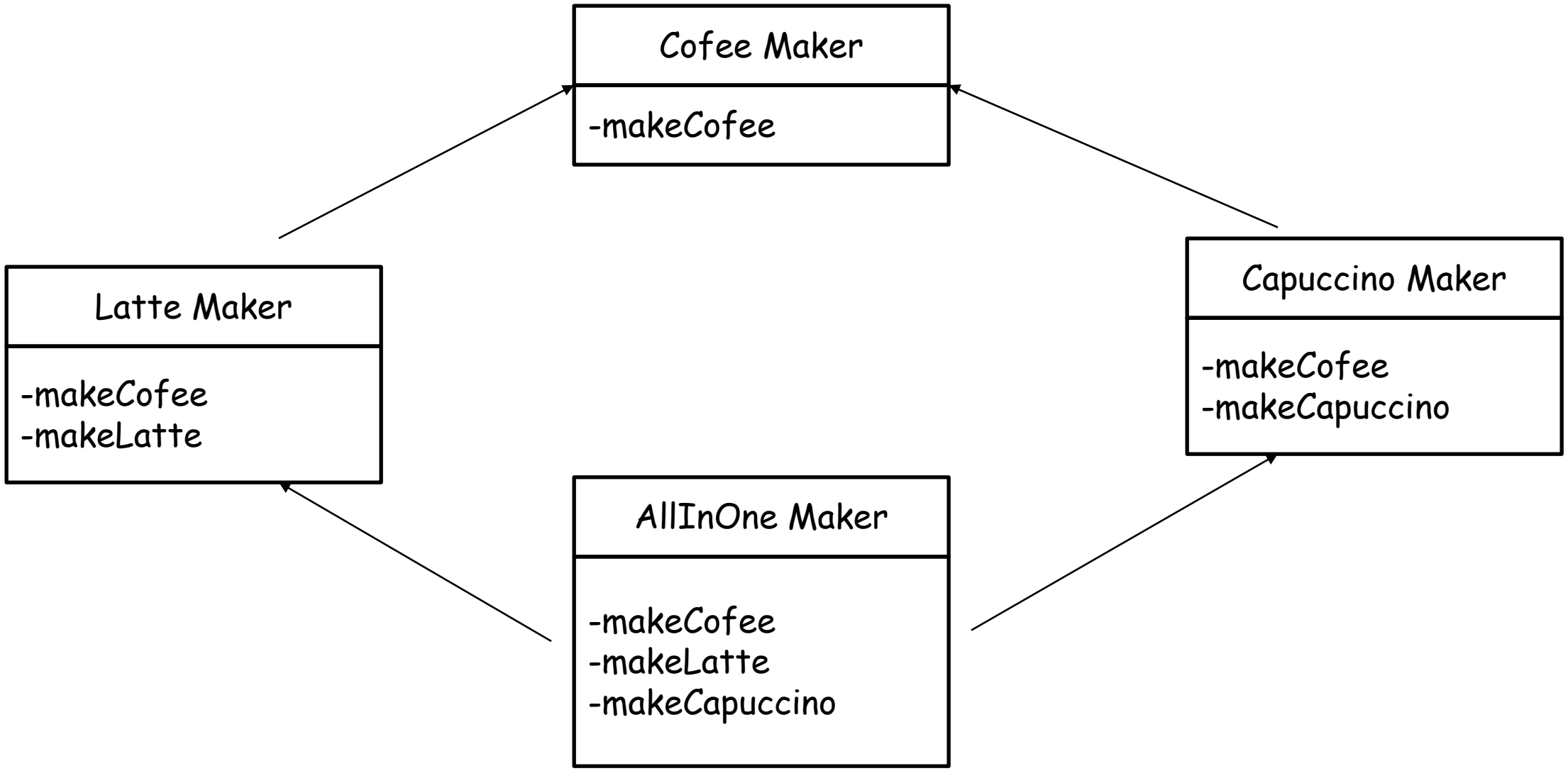
Override inherited method

Override trait insert method

```php
class MyChild extends MyParent
{
  use PrintFunctionality;
  public function myPrint() { echo 'Child'; }
}
$o = new MyChild();
$o->myPrint(); // "Child"
```

Gợi ý

TRAIT, ABSTRACT, INHERITANCE



TRAIT, ABSTRACT, INHERITANCE

Cofee Maker

```php
class CoffeeMaker
{
  public function makeCoffee()
  {
    echo static::class.' is making coffee'.PHP_EOL;
  }
}
```

Latte Maker

```php
trait LatteTrait{
  public function makeLatte()
  {
    echo static::class.' is making latte';
  }
}
class LatteMaker extends CoffeeMaker
{
  use LatteTrait;
}
```

Capuccino Maker

```php
trait CapuccinoTrait{
  public function makeCapuccino()
  {
    echo static::class.' is making capuccino'.PHP_EOL;
  }
}
class CapuccinoMaker extends CoffeeMaker
{
  use LatteTrait;
  public function getMilkType():string{
    return '';
  }
}
```

Capuccino Maker

```php
class AllInOne extends CoffeeMaker{
  use LatteTrait, CapuccinoTrait;
  public function getMilkType():string{
    return '';
  }
}
```

TRAIT, ABSTRACT, INHERITANCE

properties

Abstract method

Cofee Maker

```php
trait LatteTrait{
  protected string $milkType = 'whole-milk';
  public function makeLatte()
  {
    echo static::class.' is making latte with '.$this->milkType.PHP_EOL;
  }
  abstract public function getMilkType():string;
}
```

Latte Maker

```php
trait LatteTrait{
  public function makeLatte()
  {
    echo static::class.' is making latte';
  }
}
class LatteMaker extends CoffeeMaker
{
  use LatteTrait;
}
```

Capuccino Maker

```php
trait CapuccinoTrait{
  public function makeCapuccino()
  {
    echo static::class.' is making capuccino'.PHP_EOL;
  }
}
class CapuccinoMaker extends CoffeeMaker
{
  use LatteTrait;
  public function getMilkType():string{
    return '';
  }
}
```

Capuccino Maker

```php
class AllInOne extends CoffeeMaker{
  use LatteTrait, CapuccinoTrait;
  public function getMilkType():string{
    return '';
  }
}
```

SUMARRY

- ☑ Inheritance
- ☑ Access level & Class constant
- ☑ Interface
- ☑ Abstract
- ☑ Trait


thank you!