Bài 5
Kế thừa

Module: **ADVANCED PROGRAMMING WITH PHP 2.0**

Mục tiêu

- Trình bày được cơ chế kế thừa
- Triển khai được cơ chế kế thừa giữa các lớp
- Trình bày được cơ chế ghi đè phương thức
- Triển khai được cơ chế ghi đè phương thức
- Biểu diễn được mối quan hệ kế thừa bằng các ký hiệu
- Trình bày được ý nghĩa của từ khoá final

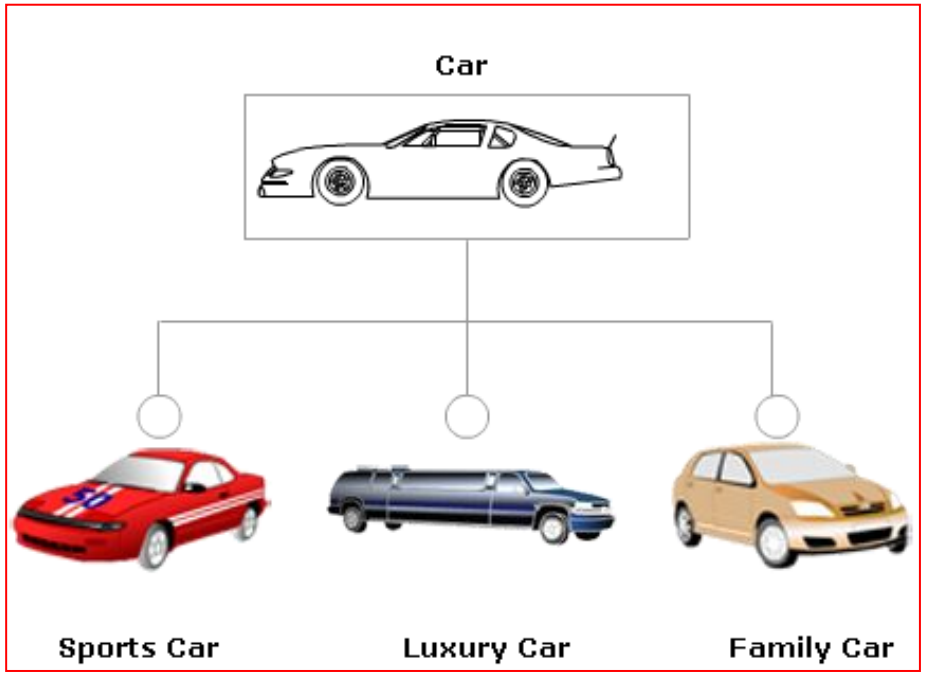
Thảo luận

Thừa kế là gì

Thừa kế được thực thi như thế nào trong code PHP

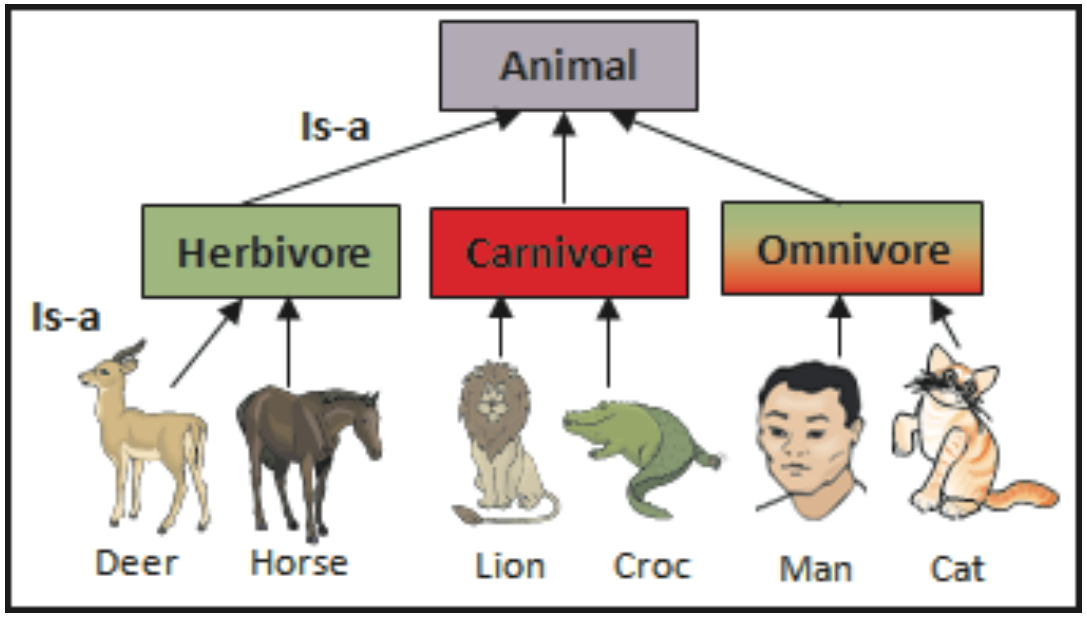
Kế thừa

- Kế thừa là cơ chế cho phép một lớp Con sử dụng lại các đặc điểm và hành vi đã được định nghĩa trong lớp Cha
- Ví dụ
  - Lớp Cha: Car
  - Lớp Con: Sports Car, Luxury Car, Family Car



Quan hệ is-a

- Quan hệ giữa lớp con và lớp cha là quan hệ *is-a (là-một)*
- Ví dụ: Ngựa *là một* động vật ăn cỏ, sư tử *là một* động vật ăn thịt, động vật ăn cỏ *là một* động vật…

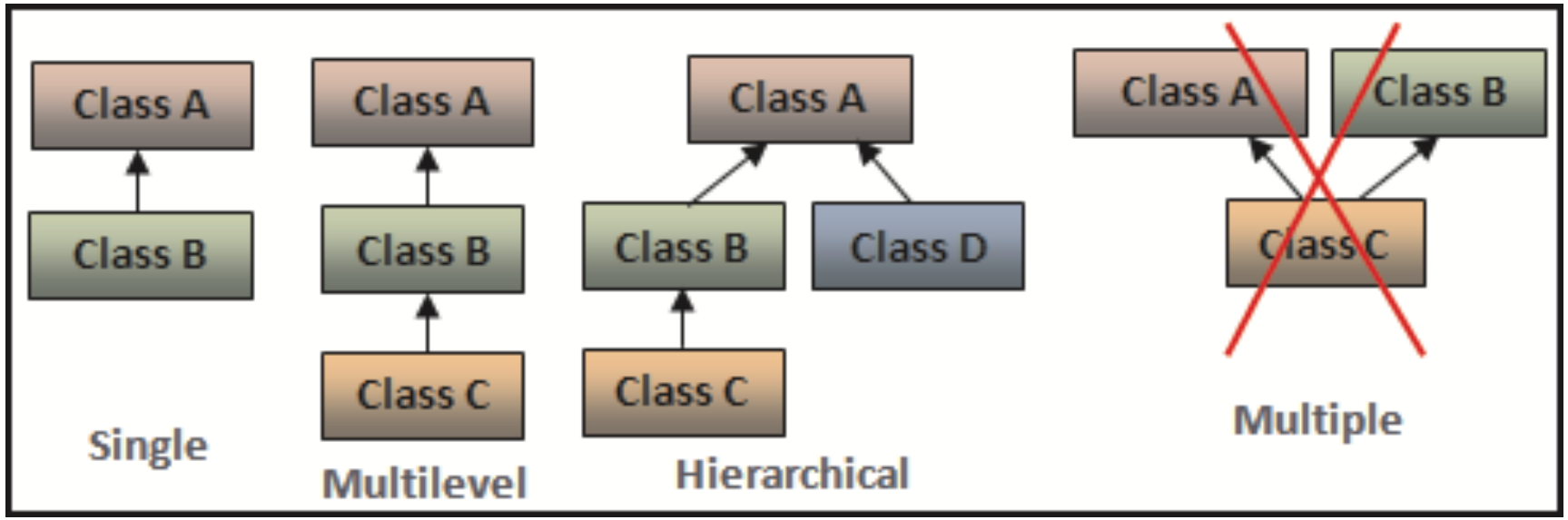


Các khái niệm

- Lớp được kế thừa gọi là lớp cha (parent class) hoặc lớp cơ sở (base class) hoặc super class
- Lớp kế thừa gọi là lớp con (child class) hoặc lớp dẫn xuất (derived class) hoặc subclass
- Lớp con kế thừa tất cả các thành phần của lớp cha, ngoại trừ các thành phần được khai báo là *private*
- Lớp con có thể gọi constructor của lớp cha
- Lớp con có thể định nghĩa thêm các thuộc tính và phương thức mới
- PHP không cho phép đa kế thừa (một lớp kế thừa nhiều lớp cha)

Một số dạng kế thừa

- **Single**: Một lớp kế thừa từ chỉ một lớp cha
- **Multilevel**: Một lớp kế thừa từ một lớp cha, lớp cha lại kế thừa từ lớp khác ở trên nó
- **Hierarchical**: Một lớp cha có nhiều lớp con với nhiều level khác nhau
- **Multiple**: Một lớp con kế thừa từ nhiều lớp cha



Cú pháp kế thừa

- Từ khoá extends được sử dụng để kế thừa một lớp
- Cú pháp:

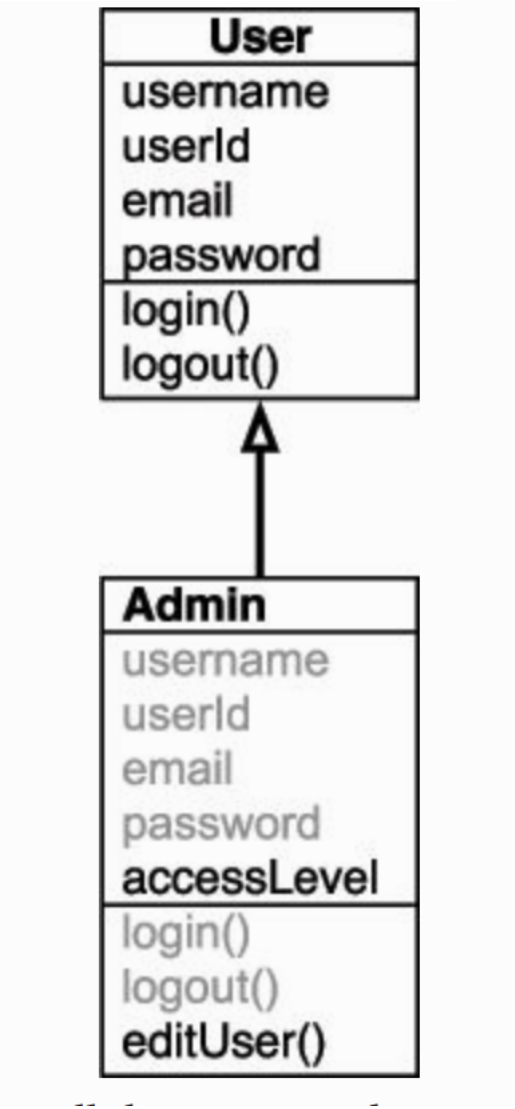
```
class SubClass extends SupperClass
{
   //Class body
}
```

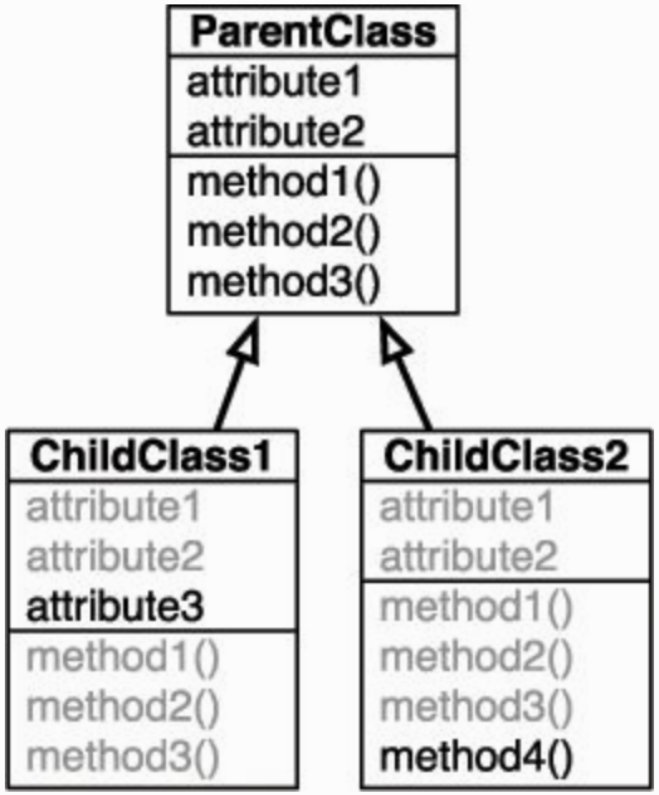
Trong đó:

- **SubClass** là tên của lớp con
- **SupperClass** là tên của lớp cha

Kế thừa - Thiết kế

- Kế thừa (inheritance) là một đặc điểm quan trọng của mô hình Lập trình hướng đối tượng
- Kế thừa là cơ chế cho phép một lớp thừa hưởng các thuộc tính và phương thức của một lớp khác đã được định nghĩa trước đó





Kế thừa - Ví dụ

- Lớp Foo có 2 phương thức là printItem($string) và printPHP()

```php
class Foo
{
    public function printItem($string)
    {
        echo 'Foo: ' . $string . PHP_EOL;
    }

    public function printPHP()
    {
        echo 'PHP is great.' . PHP_EOL;
    }
}
```

Kế thừa - Ví dụ

- Lớp con Bar kế thừa từ lớp Foo và định nghĩa lại phương thức printItem()

```php
class Bar extends Foo
{
    public function printItem($string)
    {
        echo 'Bar: ' . $string . PHP_EOL;
    }
}
```

```php
$foo = new Foo();
$bar = new Bar();
$foo->printItem('baz'); // Output: 'Foo: baz'
$foo->printPHP();       // Output: 'PHP is great'
$bar->printItem('baz'); // Output: 'Bar: baz'
$bar->printPHP();       // Output: 'PHP is great'
```

Method overriding

Method overriding

- Method Overriding (ghi đè phương thức) là cơ chế cho phép lớp con định nghĩa lại các phương thức đã được định nghĩa trước đó ở lớp cha
- Phương thức override ở lớp con có cùng tên, cùng danh sách tham số và kiểu dữ liệu trả về so với phương thức ở lớp cha
- Phương thức ở lớp con phải có access modifier có level bằng hoặc cao hơn so với phương thức ở lớp cha

Ghi đè phương thức - ví dụ

```
class SomeClass {
    function scream($count = 1){
        for($i = 0; $i < $count; $i++){
            echo "Eek! <br/>";
        }
    }
}

class SomeOtherClass extends SomeClass{
    function scream($count = 1){
        for($i = 0; $i < $count; $i++){
            echo "Whoohoo! <br/>";
        }
    }
}
```

Eek!
Eek!
Eek!
Whoohoo!
Whoohoo!
Whoohoo!

Từ khoá parent

- Từ khoá **parent** được sử dụng trong một lớp con để truy xuất các thuộc tính và phương thức của lớp cha

```php
<?php
class A {
   function example() {
      echo "I am A::example() and provide basic functionality.<br />\n";
   }
}

class B extends A {
   function example() {
      echo "I am B::example() and provide additional functionality.<br />\n";
      parent::example();
   }
}

$b = new B;

// This will call B::example(), which will in turn call A::example().
$b->example();
?>
```

parent::__construct

- parent::_construct được sử dụng trong lớp con để gọi đến constructor của lớp cha

```php
<?php
class Circle {
    public int $radius;
    public function __construct(int $radius) {
        $this->radius = $radius;
    }
}

class Cylinder extends Circle {
    public int $height;
    public function __construct(int $radius, int $height) {
        parent::__construct($radius);
        $this->height = $height;
    }
}

$cylinder = new Cylinder(10, 30);
?>
```

Từ khoá final

- Từ khoá final được sử dụng với class hoặc method
  - Với class: Ngăn chặn việc kế thừa một lớp
  - Với method: Ngăn chặn việc ghi đè một phương thức

- Final class không thể là abstract class
  - Tại sao?

Final method: Ví dụ

```php
<?php
class BaseClass {
  public function test() {
    echo "BaseClass::test() called\n";
  }

  final public function moreTesting() {
    echo "BaseClass::moreTesting() called\n";
  }
}

class ChildClass extends BaseClass {
  public function moreTesting() {
    echo "ChildClass::moreTesting() called\n";
  }
}
?>
```

Phương thức final

Lỗi: Không thể ghi đè phương thức moreTesting

Final class: Ví dụ

Lớp final

```php
<?php
final class BaseClass {
  public function test() {
    echo "BaseClass::test() called\n";
  }

  // Here it doesn't matter if you speci
 not
  final public function moreTesting() {
    echo "BaseClass::moreTesting() called\n";
  }
}

class ChildClass extends BaseClass {
}

?>
```

Lỗi: Không thể kế thừa lớp BaseClass

Tóm tắt bài học

- Kế thừa là cơ chế cho phép một lớp thừa hưởng các đặc điểm và hành vi củamột lớp khác
- Lớp được kế thừa gọi là lớp cha, lớp kế thừa gọi là lớp con
- Ghi đè phương thức là hình thức lớp con viết lại các phương thức đã có củalớp cha
- Sử dụng mũi tên rỗng để biểu diễn mối quan hệ kế thừa giữa các lớp
- PHP không hỗ trợ đa kế thừa
- Từ khoá final được sử dụng để ngăn chặn việc kế thừa từ một lớp và việc ghiđè phương thức
- Đa hình là cơ chế cho phép một biến kiểu cha có thể trỏ đến các đối tượng kiểucon
- Phương thức toString() được sử dụng để trả về một chuỗi mô tả đối tượng

Hướng dẫn

- Hướng dẫn làm bài thực hành và bài tập
- Chuẩn bị bài tiếp: ***Advanced Object Oriented Design***