BÀI 8:
Lập trình hướng đối tượng với PHP

Mục tiêu bài học

- Giới thiệu tổng quan về phương pháp lập trình hướng đối tượng (OOP)
- Định nghĩa class trong PHP
- Định nghĩa constructor của class
- Khởi tạo đối tượng trong PHP
- Kế thừa trong PHP.
- Abstract class và Interface

Lập trình hướng đối tượng (OOP)

- **Object-oriented programming (OOP)**
  - Là một phương pháp lập trình dựa trên khái niệm của các đối tượng. Các đối tượng chứa cả dữ liệu (còn được gọi là các thuộc tính) và các hành vi (còn được gọi là các method).
  - OOP cho phép phân rã một vấn đề thành một số các thực thể được gọi là các đối tượng, sau đó ta có thể xây dựng các dữ liệu và chức năng cho các đối tượng này.
    - Phần mềm sẽ bao gồm một số các đối tượng.
    - Dữ liệu của các đối tượng chỉ có thể được truy cập thông qua các phương thức của đối tượng.
    - Các đối tượng có mối quan hệ với nhau, chúng có thể gọi đến nhau, hoặc chứa lẫn nhau.

Lập trình hướng đối tượng với Java

- **Các đặc điểm của phương pháp lập trình hướng đối tượng**
  - Tính kế thừa (Inheritance)
  - Trừ tượng hóa dữ liệu (Data Abstraction)
  - Đóng gói dữ liệu (Data Encapsulation)
  - Tính đa hình (Polymorphism)

Đối tượng

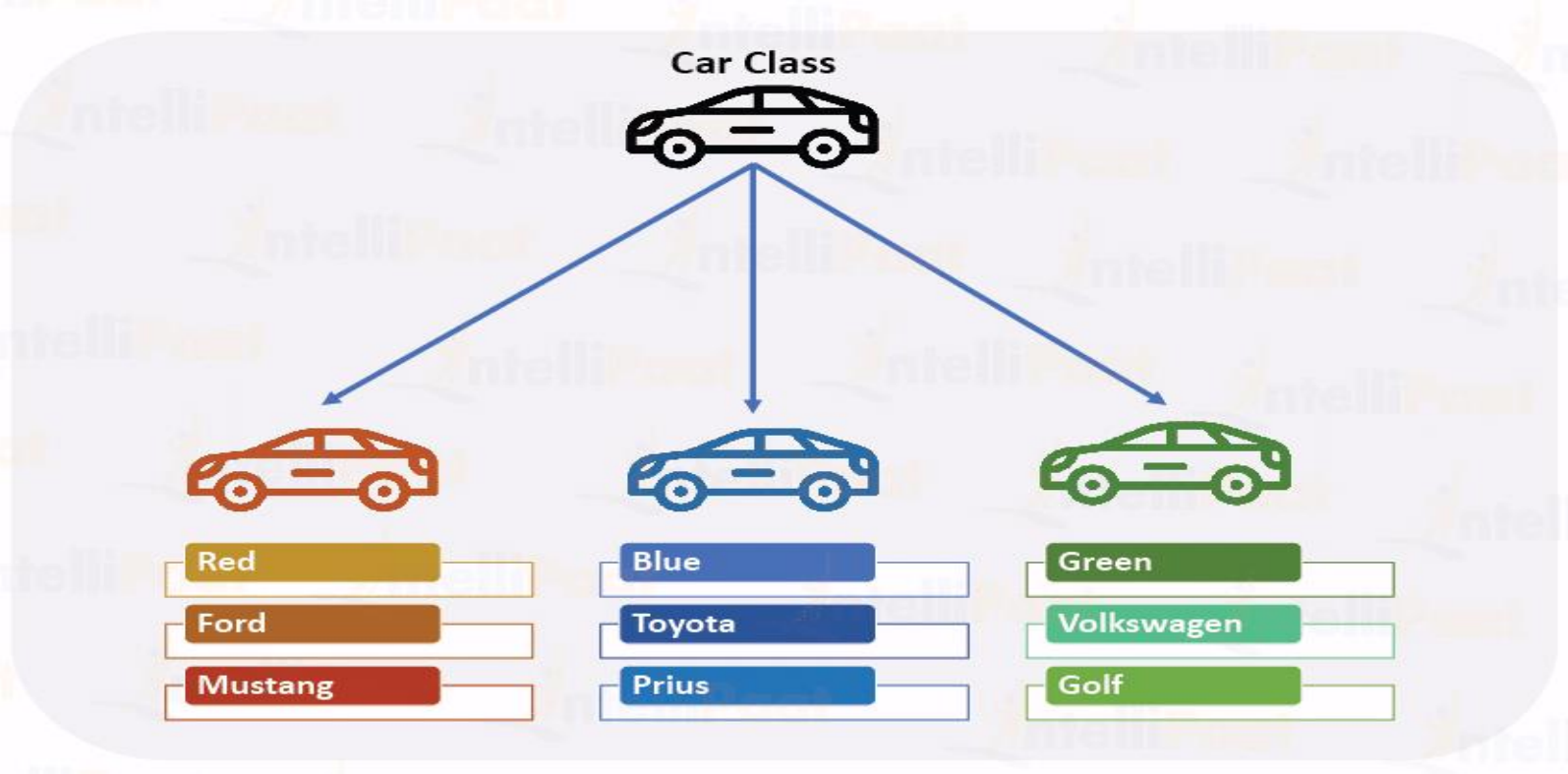
- Đối tượng là một sự mô phỏng trong phần mềm về một thực thể trong thế giới thực.
- Thực thể (entity) là các sự vật, hiện tượng tồn tại trong thế giới thực, VD như oto, xe máy, người, sinh viên, bác sĩ v.v…
- Mọi thực thể đều có các tính chất và có thể thực hiện một số hành động
  - Tính chất (Characteristics) là các thuộc tính (attributes) hoặc đặc điểm mô tả về thực thể.
  - Hành vi: là các hoạt động của một đối tượng.

Đối tượng

- Giống như các thực thể trong thế giới thực, một đối tượng trong phần mềm cũng có các trạng thái và hành vi.
  - Trạng thái (state) của đối tượng thể hiện các thuộc tính hay tính chất của đối tượng đó.
  - Hành vi (behavior) của đối tượng thể hiện các hành động của đối tượng đó.
- Các đối tượng trong phần mềm lưu trữ trạng thái của chúng trong các field (còn được gọi là các biến trong các ngôn ngữ lập trình)và cung cấp các hành vi của chúng thông qua các phương thức (hay còn gọi là các hàm).

Định nghĩa lớp

- Lớp là một khuôn mẫu, hoặc là một bản thiết kế để mô tả về các đối tượng cùng loại.
- Trong một ngôn ngữ lập trình hướng đối tượng, một lớp sẽ bao gồm các thuộc tính và các phương thức, thường được gọi là các thành phần của lớp.
- Thuộc tính thể hiện các trạng thái, còn phương thức thể hiện các hành vi của lớp.



Định nghĩa lớp

- Định nghĩa của lớp sẽ bao gồm 2 thành phần:
  - Các thuộc tính, thể hiện trạng thái của lớp.
  - Các phương thức, thể hiện các hành vi của lớp.
- Cú pháp định nghĩa lớp:

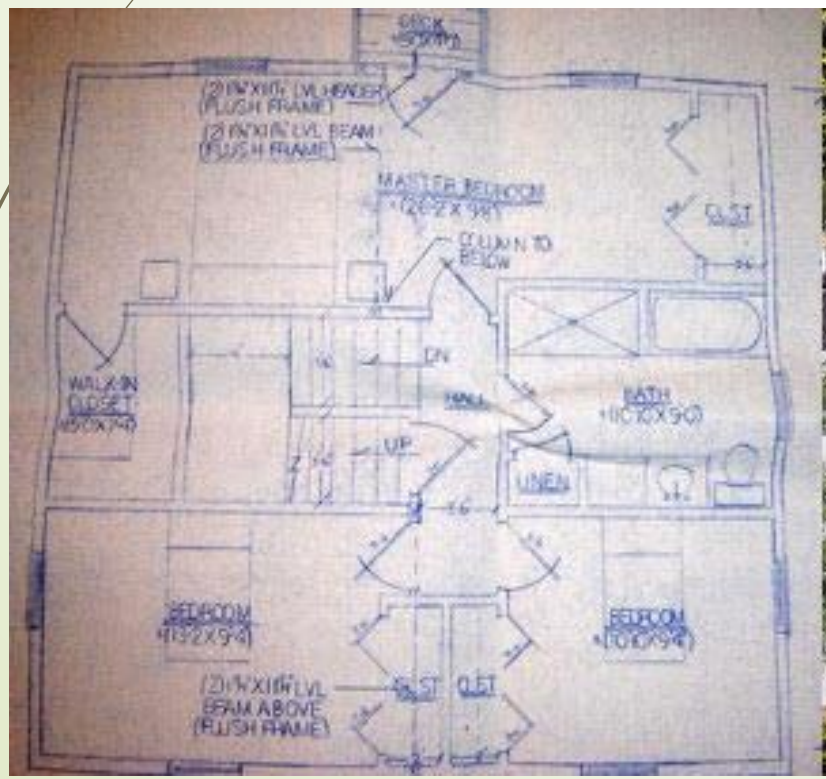
```
class <ClassName>
{
//Khai báo các thuộc tính
//Định nghĩa các phương thức
}
```

Quan hệ giữa lớp và đối tượng

- Lớp là định nghĩa của kiểu dữ liệu, còn đối tượng là các biến thuộc kiểu dữ liệu. Lớp tạo ra các đối tượng.
- Lớp là khuôn mẫu để mô tả về các đối tượng cùng loại. Đối tượng là sự thể hiện cụ thể của lớp.
- Mỗi đối tượng được tạo ra từ lớp sẽ chứa một bản copy của các biến thể hiện và các phương thức thể hiện được định nghĩa trong lớp.
- Để có thể truy cập đến các thành phần của đối tượng, ta sử dụng dấu chấm (.). Dấu chấm sẽ liên kết tên của đối tượng với tên của biến/phương thức thể hiện.

Class & Object

- Class có thể coi như một bản thiết kế cho các căn nhà. Nó mô tả về ngôi nhà trên giấy, mô tả về mối quan hệ giữa các phần của căn nhà, kể cả khi căn nhà chưa tồn tại.
- Object chính là một căn nhà thực tế, được xây dựng dựa trên bản thiết kế (lớp). Căn nhà này sẽ chứa đầy đủ các đặc điểm, tính chất đã được mô tả trong bản thiết kế.



Các khái niệm thông dụng trong OOP

- **Class**
  - Là những kiểu dữ liệu bao gồm dữ liệu và chức năng. Lớp là khuôn mẫu để mô tả về những đối tượng cùng loại.
- **Object**
  - Là một thể hiện cụ thể của lớp, được sinh ra từ lớp. Lớp là kiểu dữ liệu, còn đối tượng là những biến được tạo ra từ kiểu dữ liệu đó. Ta có thể định nghĩa lớp, rồi khởi tạo các đối tượng thuộc lớp đó.
- **Member Variable**
  - Là những biến được khai báo bên trong một class. Những biến này sẽ invisible đối với bên ngoài lớp, và có thể được truy cập thông qua các phương thức của lớp. Những biến này còn được gọi là các thuộc tính của lớp.

Các khái niệm thông dụng trong OOP

- **Member function**
  - Là những hàm được định nghĩa bên trong một class, và được sử dụng để truy cập đến dữ liệu của đối tượng.
- **Parent class**
  - Là lớp được kế thừa bởi các lớp khác. Lớp cha còn được gọi là base class hoặc super class.
- **Child Class**
  - Là lớp thừa kế từ lớp khác. Lớp con còn được gọi là subclass hoặc derived class.
- **Constructor**
  - Là một phương thức đặc biệt, được gọi một cách tự động mỗi khi khởi tạo một đối tượng của lớp.
- **Destructor**
  - Là một phương thức đặc biệt được gọi một cách tự động mỗi khi ta hủy một đối tượng của lớp.

Các đặc điểm của OOP

- **Inheritance**
  - Là quá trình lớp con kế thừa các đặc điểm và hành vi của lớp cha. Một lớp cha có thể có nhiều lớp con, nhưng một lớp con chỉ có một lớp cha.
- **Polymorphism**
  - Là đặc điểm cho phép cùng một hành vi nhưng các đối tượng khác nhau (trong cùng lớp hoặc khác lớp) sẽ thực hiện theo cách khác nhau, và kết quả trả về sẽ khác nhau.
- **Overloading**
  - Là một kiểu polymorphism trong đó cho phép ta định nghĩa nhiều phương thức trong một lớp, các phương thức này có cùng tên, cùng kiểu trả về nhưng có số lượng các tham số khác nhau. Overloading cho phép cung cấp nhiều cách thực hiện khác nhau cho cùng một hành vi của lớp.
- **Data Abstraction**
  - Là quá trình ẩn đi những chi tiết cài đặt các hành vi bên trong các đối tượng.
- **Encapsulation**
  - Là quá trình đóng gói các dữ liệu và hành vi trong các đối tượng.

Định nghĩa class

- Định nghĩa lớp bắt đầu bằng từ khóa **class**, theo sau là tên lớp muốn định nghĩa.
- Cặp dấu ngoặc nhọn ({}) thể hiện than của lớp, tất cả những phương thức hay thuộc tính của lớp sẽ được khai báo bên trong cặp dấu này.
- Để khai báo thuộc tính của lớp, ta bắt đầu bằng từ khóa **var**, sau đó là $ tên biến.
- Định nghĩa của các phương thức trong lớp giống với định nghĩa hàm trong PHP, bắt đầu bằng từ khóa function.
- Biến **$this** là một biến đặc biệt dùng để truy cập đến các thành phần của đối tượng hiện tại.

```php
<?php
   class Books {
      /* Member variables */
      var $price;
      var $title;

      /* Member functions */
      function setPrice($par){
         $this->price = $par;
      }

      function getPrice(){
         echo $this->price ."<br/>";
      }

      function setTitle($par){
         $this->title = $par;
      }

      function getTitle(){
         echo $this->title ." <br/>";
      }
   }
?>
```

Constructor

- Phương thức khởi tạo (Constructor) là những phương thức đặc biệt, sẽ được gọi một cách tự động khi tạo ra đối tượng của lớp. Phương thức khởi tạo được sử dụng khi ta muốn khởi tạo các thuộc tính của đối tượng.
- PHP cung cấp một hàm đặc biệt, được gọi là **__construct()** để định nghĩa một constructor. Ta có thể truyền các đối số vào cho constructor nếu muốn.
- Ví dụ:

```
function __construct( $par1, $par2 ) {
   $this->title = $par1;
   $this->price = $par2;
}
```

Constructor

- Ví dụ:

```
$physics = new Books( "Physics for High School", 10 );
$maths = new Books ( "Advanced Chemistry", 15 );
$chemistry = new Books ("Algebra", 7 );

/* Get those set values */
$physics->getTitle();
$chemistry->getTitle();
$maths->getTitle();

$physics->getPrice();
$chemistry->getPrice();
$maths->getPrice();
```

- Kết quả:

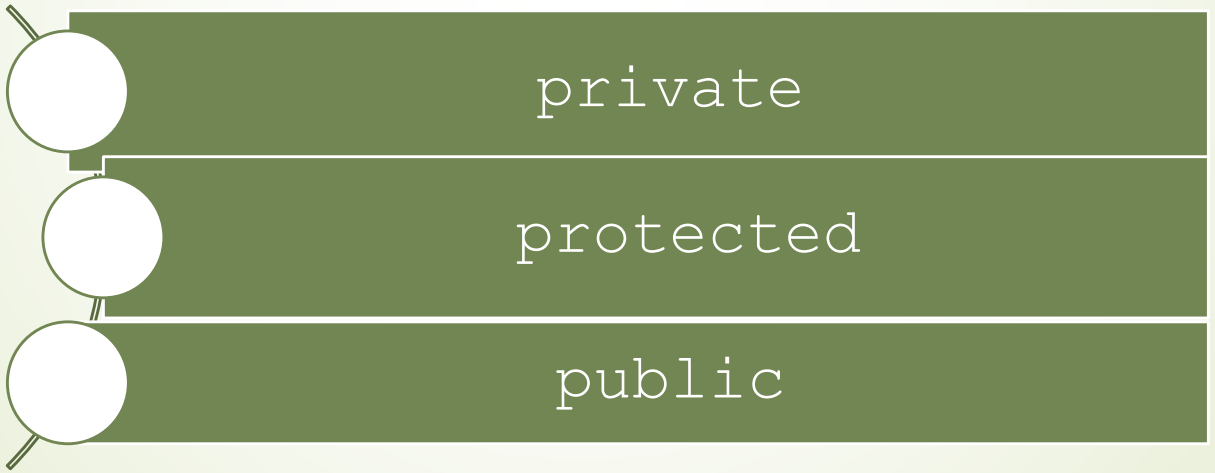
```
Physics for High School
Advanced Chemistry
Algebra
10
15
7
```

Thực hành

- Thực hành định nghĩa lớp và khởi tạo đối tượng từ lớp.

Access Specifiers

- Access specifier: Dùng để kiểm soát việc truy cập đến các thành phần của lớp. Cho phép ẩn đi các chi tiết cài đặt bên trong của các lớp.
- Có 3 access specifiers: public, private, protected và default (hoặc không khai báo modifier).
- Các loại access specifiers trong PHP là:



Sử dụng Access Specifiers

- private access specifiers cho phép một lớp có thể ẩn đi các biến và phương thức thành phần đối với các lớp khác.
- Thành phần private của một lớp sẽ không thể truy cập được bởi các lớp khác. Các thành phần private chỉ có thể truy cập được trong nội bộ lớp.
- Thành phần của lớp có khai báo với protected access specifier có thể được truy cập được trong nội bộ lớp, và từ các lớp con.
- Thành phần của lớp được khai báo với public access specifier có thể được truy cập bởi tất cả các lớp khác.

Thực hành

- Thực hành sử dụng access specifier

Từ khóa Static

- Đối với các thuộc tính và phương thức của lớp, để có thể truy cập chúng, ta cần khởi tạo một đối tượng của lớp.
- Việc khai báo các thành phần của lớp (thuộc tính/phương thức) là static cho phép ta truy cập trực tiếp đến chúng, mà không cần phải khởi tạo đối tượng của lớp.
- Ví dụ:

```php
<?php
    class Foo {
        public static $my_static = 'foo';

        public function staticValue() {
            return self::$my_static;
        }
    }

    print Foo::$my_static . "\n";
    $foo = new Foo();

    print $foo->staticValue() . "\n";
?>
```

Biến Static

- Các biến lớp (class variables) được khai báo bằng cách sử dụng từ khóa static.
- Tất cả các thể hiện của lớp sẽ cùng chia sẻ giá trị của các biến thuộc phạm vi lớp.
- Khi giá trị của một biến lớp thay đổi, tất cả các thể hiện của lớp sẽ được cập nhật để chia sẻ cùng giá trị.
- Ta khai báo các thành phần static (biến/phương thức), khi muốn các thành phần này thuộc lớp, và các lớp khác có thể truy cập trực tiếp thông qua tên lớp, mà không cần phải khởi tạo đối tượng của lớp.

Khởi tạo đối tượng trong PHP

- Ngay sau khi định nghĩa class, ta có thể khởi tạo các đối tượng của lớp bằng cách sử dụng toán tử **new**.
- Ví dụ:

```
$physics = new Books;
$maths = new Books;
$chemistry = new Books;
```

Gọi phương thức của đối tượng

- Sau khi tạo đối tượng của lớp, ta có thể gọi các phương thức của đối tượng đó.
- Mỗi đối tượng được tạo ra từ lớp sẽ có một bản copy các thuộc tính và phương thức được định nghĩa ở trong lớp.
- Ví dụ:

```
$physics->setTitle( "Physics for High School" );
$chemistry->setTitle( "Advanced Chemistry" );
$maths->setTitle( "Algebra" );

$physics->setPrice( 10 );
$chemistry->setPrice( 15 );
$maths->setPrice( 7 );
```

Destructor

- Tương tự như constructor, ta có thể định nghĩa destructor cho class bằng cách sử dụng hàm **__destruct()**.
- Ta có thể viết mã để giải phóng tài nguyên trong phương thức destructor.

Tính kế thừa (Inheritance)

- Class trong PHP có thể kế thừa những đặc điểm từ lớp cha bằng cách sử dụng từ khóa extends.
- Cú pháp:

```
class Child extends Parent {
    <definition body>
}
```

- Qúa trình kế thừa có những lợi ích sau:
  - Giúp tái sử dụng được mã nguồn → Tiết kiệm được thời gian và công sức viết mã.
  - Giúp cho quá trình phát triển ứng dụng được đơn giản hơn.
- Một lớp cha (super class) có thể có nhiều lớp con, nhưng mỗi lớp con (sub class) chỉ có thể kế thừa từ một lớp cha.

Ví dụ

```
class Novel extends Books {
    var $publisher;

    function setPublisher($par){
        $this->publisher = $par;
    }

    function getPublisher(){
        echo $this->publisher. "<br />";
    }
}
```

Public Members

- Unless you specify otherwise, properties and methods of a class are public. Public member may be accessed in three possible situations:
  - From outside the class in which it is declared
  - From within the class in which it is declared
  - From within another class that implements the class in which it is declared
- If you wish to limit the accessibility of the members of a class then you define class members as **private** or **protected**.

Private members

- By designating a member private, you limit its accessibility to the class in which it is declared.
- The private member cannot be referred to from classes that inherit the class in which it is declared and cannot be accessed from
- e made private by
- in front of the

```
class MyClass {
   private $car = "skoda";
   $driver = "SRK";

   function __construct($par) {
      // Statements here run every time
      // an instance of the class
      // is created.
   }

   function myPublicFunction() {
      return("I'm visible!");
   }

   private function myPrivateFunction() {
      return("I'm  not visible outside!");
   }
}
```

Protected members

- A protected property or method is accessible in the class in which it is declared, as well as in classes that extend that class.
- Protected members are not available inds of classes.
- be made otected keyword in

```
class MyClass {
   protected $car = "skoda";
   $driver = "SRK";

   function __construct($par) {
      // Statements here run every time
      // an instance of the class
      // is created.
   }

   function myPublicFunction() {
      return("I'm visible!");
   }

   protected function myPrivateFunction() {
      return("I'm  visible in child class!");
   }
}
```

Interfaces

- Interfaces là một dạng lớp cha đặc biệt, được định nghĩa để cung cấp các giao diện chung cho các lớp cài đặt.
- Một lớp con chỉ có thể kế thừa một lớp cha, nhưng có thể cài đặt nhiều interface.
- Để khai báo một lớp cài đặt một interface, ta sử dụng từ khóa implements.
- Cú pháp:

```
interface Mail {
   public function sendMail();
}
```

- Ví dụ cài đặt interface:

```
class Report implements Mail {
   // sendMail() Definition goes here
}
```

Abstract Classes

- Abstract class
  - Là lớp không thể khởi tạo đối tượng.
  - Là lớp có khai báo từ khóa **abstract**.
  - Thông thường được dùng để định nghĩa một lớp cha.
- Khi kế thừa một abstract class, ta cần cung cấp định nghĩa cho tất cả các phương thức abstract ở lớp cha. Ngoài ra, những phương thức cài đặt ở lớp con phải có mức truy cập (visibility) bằng hoặc hơn phương thức abstract ở lớp cha.
- Ví dụ:

```
abstract class MyAbstractClass {
    abstract function myAbstractFunction() {
    }
}
```

Từ khóa final

- Từ khóa final cho phép ngăn chặn lớp con kế thừa một lớp cha.
- Từ khóa final có thể áp dụng cho lớp hoặc phương thức.
- Nếu một phương thức ở lớp cha được khai báo là final, lớp con sẽ không thể ghi đè (override) phương thức đó.
- Ví dụ trên sẽ dẫn đến một Fatal error: Cannot override final method BaseClass::moreTesting()

```php
<?php

  class BaseClass {
     public function test() {
        echo "BaseClass::test() called<br>";
     }

     final public function moreTesting() {
        echo "BaseClass::moreTesting() called<br>";
     }
  }

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

Demo

- Giáo viên demo các tính năng OOP trong PHP

Tổng kết

- Giới thiệu tổng quan về phương pháp lập trình hướng đối tượng (OOP)
- Định nghĩa class trong PHP
- Định nghĩa constructor của class
- Khởi tạo đối tượng trong PHP
- Kế thừa trong PHP.
- Abstract class và Interface