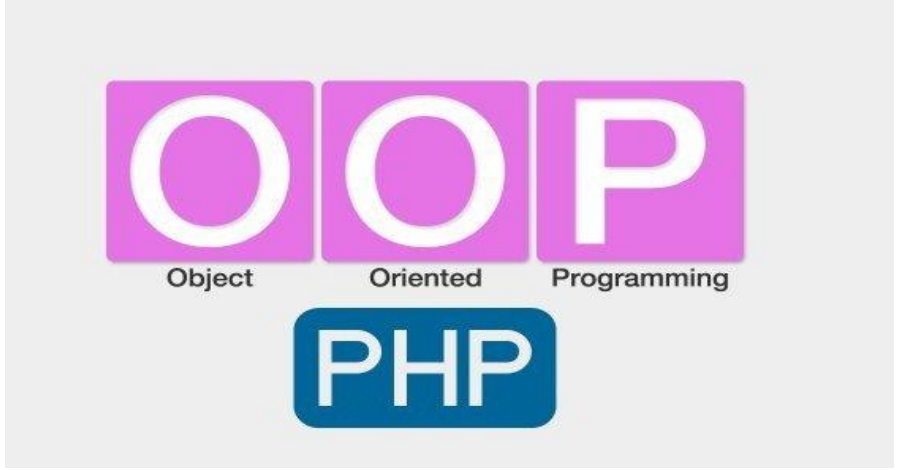
# Bài 01:
# HƯỚNG ĐỐI TƯỢNG TRONG PHP

Giảng Viên: ThS. Giang Hào Côn

# 1.1/ OOP trong PHP là gì ?

- Trong PHP, khi nói đến OOP thì chúng ta hiểu đó chính là chữ viết tắt của cụm từ Object-Oriented Programming (lập trình hướng đối tượng).
- Lập trình theo kiểu truyền thống được gọi là "Lập trình thủ tục (Procedural programming)". Với kiểu lập trình thủ tục thì chúng ta sẽ viết các thủ tục hoặc các hàm thực hiện các thao tác trên dữ liệu, còn đối với lập trình hướng đối tượng thì chúng ta sẽ tạo ra các đối tượng chứa cả dữ liệu và hàm.

# 1.2/ Ưu điểm của OOP

- Tốc độ nhanh, dễ thực thi hơn.
- Việc viết mã PHP theo kiểu hướng đối tượng sẽ cung cấp cho bạn một cấu trúc rõ ràng, dễ nhìn, dễ hiểu.
- Với cấu trúc rõ ràng sẽ giúp đơn giản hóa việc bảo trì, sửa đổi, gỡ lỗi.
- Việc lập trình hướng đối tượng giúp bạn tạo ra những ứng dụng có thể tái sử dụng đầy đủ với ít mã hơn, thời gian phát triển ngắn hơn.

# 1.3/ Tìm hiểu sơ qua khái niệm Class & Object

- Lớp (class) & đối tượng (object) là hai khía cạnh chính trong lập trình hướng đối tượng.
- Lớp (class) là một khuôn mẫu cho các đối tượng, một đối tượng (object) là một thể hiện của một lớp.
- Khi các đối tượng được tạo thì chúng sẽ kế thừa tất cả thuộc tính & phương thức từ lớp, nhưng mỗi đối tượng sẽ có các giá trị khác nhau cho các thuộc tính.
- Ví dụ

| Class | Object |
|---|---|
| Mobile | Nokia |
| | Samsung |
| | LG |

| Class | Object |
|---|---|
| Laptop | ASUS |
| | DELL |
| | HP |

# 1.3.1/ Cách khai báo Class

- Trong lập trình hướng đối tượng, một lớp là một cái khuôn mẫu cho các đối tượng, trong khi đó một đối tượng chính là một thể hiện của lớp.
- Ví dụ như chúng ta có ***một lớp tên là Fruit (trái cây)***. Một trái cây thường có các thuộc tính như tên loại trái cây, màu sắc, trọng lượng, . . . . Chúng ta có thể định nghĩa các biến như $name, $color, $weight để lưu giữ giá trị của các thuộc tính này.
- Khi các đối tượng riêng lẻ (banana, orange, apple) được tạo, chúng sẽ kế thừa các thuộc tính và hành vi từ lớp, nhưng mỗi đối tượng sẽ có các giá trị khác nhau cho các thuộc tính.

# 1.3.1/ Cách khai báo lớp (Class)

- Cú pháp:

```
class tên_lớp{
        //các thuộc tính & phương thức được khai báo tại đây
}
```

```php
<?php
    class Fruit {
        // Các thuộc tính (property)
        public $name;
        public $color;
        // Các phương thức (method)
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
    }
?>
```

Đoạn mã hình bên dùng để khai báo một lớp có tên là Fruit, lớp này có hai thuộc tính ($name, $color), hai phương thức là set_name() và get_name(). Trong đó, *phương thức set_name() dùng để thiết lập giá trị cho thuộc tính $name*, còn *phương thức get_name() thì dùng để lấy giá trị của thuộc tính $name*.

# 1.3.2/ Cách tạo đối tượng (Object)

- Các lớp sẽ không mang lại giá trị gì nếu không có đối tượng. Chúng ta có thể tạo nhiều đối tượng từ một lớp, mỗi đối tượng sẽ kế thừa tất cả các thuộc tính & phương thức được định nghĩa bên trong lớp, nhưng giá trị thuộc tính của các đối tượng có thể khác nhau.
- Để tạo một đối tượng từ một lớp thì ta sử dụng từ khóa **new**.
- Cú pháp: $variable_name = new class_name();

# 1.3.2/ Cách tạo đối tượng (Object)

- Ví dụ 01:

```php
<?php

    class Fruit {
        public $name;
        public $color;
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
    }

    $apple = new Fruit();
    $apple->set_name('Apple');

    $banana = new Fruit();
    $banana->set_name('Banana');

    echo $apple->get_name();
    echo "<br>";
    echo $banana->get_name();

?>
```

Apple
Banana

# 1.3.2/ Cách tạo đối tượng (Object)

- Ví dụ 02:

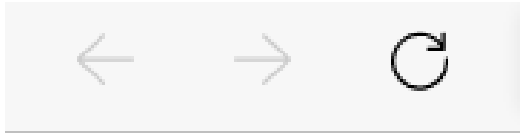
```php
<?php

    class Fruit {
        public $name;
        public $color;
        function set_name($name) {
            $this->name = $name;
        }
        function get_name() {
            return $this->name;
        }
        function set_color($color) {
            $this->color = $color;
        }
        function get_color() {
            return $this->color;
        }
    }

    $banana = new Fruit();
    $banana->set_name('Banana');
    $banana->set_color('Yellow');

    echo $banana->get_name();
    echo "<br>";
    echo $banana->get_color();

?>
```



Banana
Yellow

# 1.3.3/ Từ khóa $this trong PHP

- Từ khóa $this dùng để ám chỉ đối tượng hiện tại, nó chỉ có thể được sử dụng bên trong các phương thức.
- Hãy xem ví dụ sau:

```php
<?php
    class Fruit {
        public $name;
    }
    $banana = new Fruit();
?>
```

Vậy muốn thay đổi giá trị của thuộc tính $name thì ta làm bằng cách nào?

# 1.3.3/ Từ khóa $this trong PHP

- **Cách 01**: thay đổi Bên trong lớp (bằng cách thêm phương thức set_name() và sử dụng $this):

```php
<?php
    class Fruit {
        public $name;
        function set_name($name) {
            $this->name = $name;
        }
    }
    $banana = new Fruit();
    $banana->set_name("Banana");
?>
```

# 1.3.3/ Từ khóa $this trong PHP

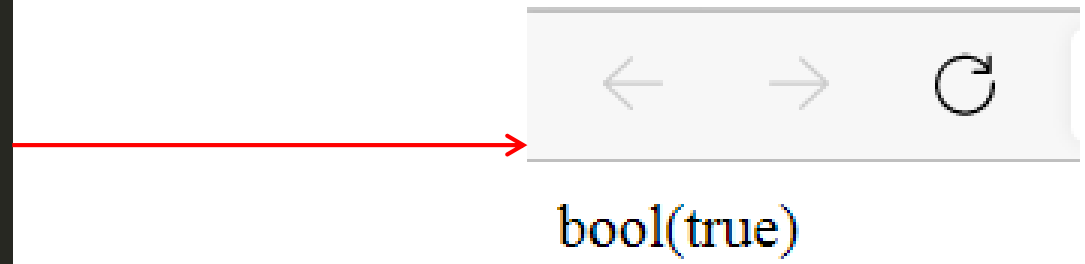
- **Cách 2**: thay đổi bên ngoài lớp (bằng cách thay đổi trực tiếp giá trị của thuộc tính):

```php
<?php
    class Fruit {
        public $name;
    }
    $banana = new Fruit();
    $banana->name = "Banana";
?>
```

# 1.3.4/ Từ khóa instanceof trong PHP

- Chúng ta có thể sử dụng từ khóa **instanceof** để kiểm tra một đối tượng có thuộc một lớp cụ thể hay không, nó sẽ trả về giá trị TRUE nếu phải, còn ngược lại thì trả về giá trị FALSE.

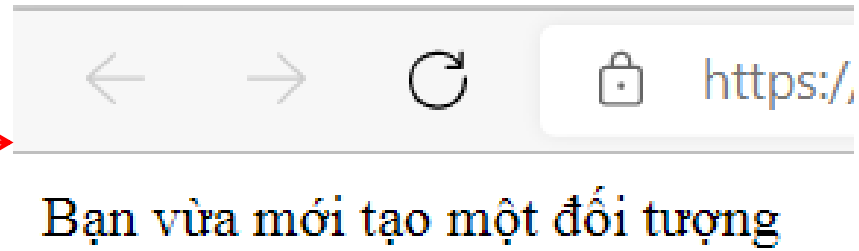
```php
<?php
    class Fruit {
        public $name;
        function set_name($name) {
            $this->name = $name;
        }
    }
    $banana = new Fruit();
    var_dump($banana instanceof Fruit);
?>
```



# 1.4/ hàm __construct() trong PHP

- Hàm **__construct()** thường được khai báo bên trong lớp, *khi ta tạo một đối tượng từ lớp đó* thì hàm __construct() sẽ tự động được gọi đến.

```php
<?php
    class Mobile{
        public $model;
        public $color;
        public $price;
        function __construct(){
            echo "Bạn vừa mới tạo một đối tượng";
        }
    }
    $nokia = new Mobile();
?>
```



# 1.4/ hàm __construct() trong PHP

- Ví dụ:

```php
<?php
    class Mobile{
        public $model;
        public $color;
        public $price;
        function __construct($input_model, $input_color, $input_price){
            $this->model = $input_model;
            $this->color = $input_color;
            $this->price = $input_price;
        }
        function get_model(){
            return $this->model;
        }
        function get_color(){
            return $this->color;
        }
        function get_price(){
            return $this->price;
        }
    }
    $nokia = new Mobile('Nokia 8.1','Black','5.000.000');
    echo "Model: " . $nokia->model . "<br>";
    echo "Màu sắc: " . $nokia->color . "<br>";
    echo "Giá tiền: " . $nokia->price;
?>
```

Model: Nokia 8.1
Màu sắc: Black
Giá tiền: 5.000.000

# 1.4/ hàm __construct() trong PHP

## Ưu điểm của việc sử dụng hàm __construct()

- Sử dụng hàm __construct() sẽ giúp *giảm thiểu việc viết mã lệnh*, chúng ta không cần phải viết các hàm dùng để thiết lập giá trị cho thuộc tính, khi tạo đối tượng thì cũng không cần sử dụng các hàm đó để gán giá trị cho thuộc tính của đối tượng, điều đó khiến cho cấu trúc của chương trình rõ ràng hơn.
- Dưới đây là đoạn mã có ý nghĩa tương tự như ví dụ ở trên, nhưng đoạn mã này *không dùng hàm __construct(),* các bạn hãy so sánh nó với đoạn mã phía trên để thấy được ưu điểm của việc sử dụng hàm __construct().

# 1.4/ hàm __construct() trong PHP

## Ưu điểm của việc sử dụng hàm __construct()

```php
<?php
    class Mobile{
        public $model;
        public $color;
        public $price;
        function set_model($input_model){
            $this->model = $input_model;
        }
        function set_color($input_color){
            $this->color = $input_color;
        }
        function set_price($input_price){
            $this->price = $input_price;
        }
        function get_model(){
            return $this->model;
        }
        function get_color(){
            return $this->color;
        }
        function get_price(){
            return $this->price;
        }
    }
    $nokia = new Mobile();
    $nokia->set_model("Nokia 8.1");
    $nokia->set_color("Black");
    $nokia->set_price("5.000.000");
    echo "Model: " . $nokia->model . "<br>";
    echo "Màu sắc: " . $nokia->color . "<br>";
    echo "Giá tiền: " . $nokia->price;
?>
```

# 1.4/ hàm __construct() trong PHP

## Một số điều cần lưu ý khi sử dụng hàm __construct()

### ✅ Phía trước construct phải có hai dấu gạch dưới

- Sợ các bạn nhìn không rõ nên tôi nói thêm, phía trước construct phải có hai dấu gạch dưới `__construct`

### ✅ Số lượng đối số của hàm __construct()

- Số lượng đối số của hàm __construct() không nhất thiết phải bằng với số lượng thuộc tính.

### ✅ Số lượng giá trị truyền vào khi tạo đối tượng

- Khi tạo một đối tượng, số lượng giá trị truyền vào không được nhỏ hơn số lượng đối số của hàm __construct().

# 1.4/ hàm __construct() trong PHP

## Một số điều cần lưu ý khi sử dụng hàm __construct()

Ví dụ 01: Bên trong lớp Mobile có ba ba thuộc tính, trong khi hàm __construct() chỉ có hai đối số, nhưng điều đó vẫn không gây ra sai lầm gì.

```
class Mobile{
    public $model;
    public $color;
    public $price;
    function __construct($input_model, $input_price){
        $this->model = $input_model;
        $this->price = $input_price;
    }
}
```

Ví dụ 02 : Hàm __construct() có ba đối số, nhưng khi tạo một đối tượng thì các bạn chỉ truyền có hai giá trị, như thế là sai.

```
class Mobile{
    public $model;
    public $color;
    public $price;
    function __construct($input_model, $input_color, $input_price){
        $this->model = $input_model;
        $this->color = $input_color;
        $this->price = $input_price;
    }
}
$nokia = new Mobile("Nokia 8.1", "Black");
```

# 1.5/ hàm __destruct() trong PHP

Trong lập trình hướng đối tượng, hàm **__destruct()** thường được khai báo bên trong một lớp, PHP sẽ tự động gọi hàm này ở cuối tập lệnh..

```php
<?php
    class Mobile{
        public $model;
        function __construct($input_model){
            $this->model = $input_model;
        }
        function __destruct(){
            echo "Model: " . $this->model;
        }
    }
    $samsung = new Mobile("Samsung Galaxy 3");
    echo "------ Dưới đây là thông tin sản phẩm ------ <br>";
?>
```

```php
<?php
    class Mobile{
        public $model;
        public $color;
        public $price;
        function __construct($input_model, $input_color, $input_price){
            $this->model = $input_model;
            $this->color = $input_color;
            $this->price = $input_price;
        }
        function __destruct(){
            echo "Model {$this->model} <br> Màu sắc: {$this->color} <br> Giá sản phẩm: {$this->price}";
        }
    }
    $samsung = new Mobile("Samsung Galaxy 3", "Blue", "3.500.000");
    echo "------ Dưới đây là thông tin sản phẩm ------ <br>";
?>
```

# 1.6/ phạm vi truy cập các thuộc tính, phương thức của lớp trong PHP

- Trong quá trình khai báo một lớp, chúng ta cần phải xác định rõ "phạm vi" có thể truy cập vào các thuộc tính & các phương thức của lớp.
- Dưới đây là ba từ khóa dùng để xác định phạm vi truy cập vào các thuộc tính & phương thức của lớp.
  - **public** (mặc định) - Thuộc tính & phương thức có thể được truy cập ở bất cứ đâu.
  - **protected** - Thuộc tính & phương thức chỉ có thể được truy cập bên trong lớp, hoặc bên trong các lớp được kế thừa từ lớp này.
  - **private** - Thuộc tính & phương thức chỉ có thể được truy cập bên trong lớp.

# 1.6/ phạm vi truy cập các thuộc tính, phương thức của lớp trong PHP

- Ví dụ.

```php
<?php
    class Laptop{
        public $model;
        protected $color;
        private $price;
    }
    $asus = new Laptop();
    $asus->model = "ASUS FX503"; //OK
    $asus->color = "Gray"; //ERROR
    $asus->price = "25.000.000"; //ERROR
?>
```

Trong lớp Laptop với ba thuộc tính $model, $color, $price có phạm vi truy cập khác nhau. Trong đó, thuộc tính $model có thể được truy cập ở bất cứ đâu (bởi vì nó là public), còn thuộc tính $color & $price thì không thể truy cập ở bên ngoài lớp, nếu các bạn cố truy cập chúng ở bên ngoài lớp thì sẽ xảy ra lỗi.

# 1.6/ phạm vi truy cập các thuộc tính, phương thức của lớp trong PHP

- Ví dụ.

```php
<?php
    class Laptop{
        public $model;
        public $color;
        public $price;
        function set_model($input_model){ //phương thức không xác định phạm vi, mặc định là “public”
            $this->model = $input_model;
        }
        protected function set_color($input_color){
            $this->color = $input_color;
        }
        private function set_price($input_price){
            $this->price = $input_price;
        }
    }
    $asus = new Laptop();
    $asus->set_model("ASUS FX503"); //OK
    $asus->set_color("Gray"); //ERROR
    $asus->set_price("25.000.000"); //ERROR
?>
```

# Bài Tập Mẫu


PHÉP TÍNH
Số thứ nhất:
1234
Số thứ hai:
24
Cộng
Trừ
Nhân
Chia
Tính
Kết quả sau khi chọn phép tính và nhấn Tính
KẾT QUẢ
1234 * 24 = 29616

# Bài Tập Mẫu

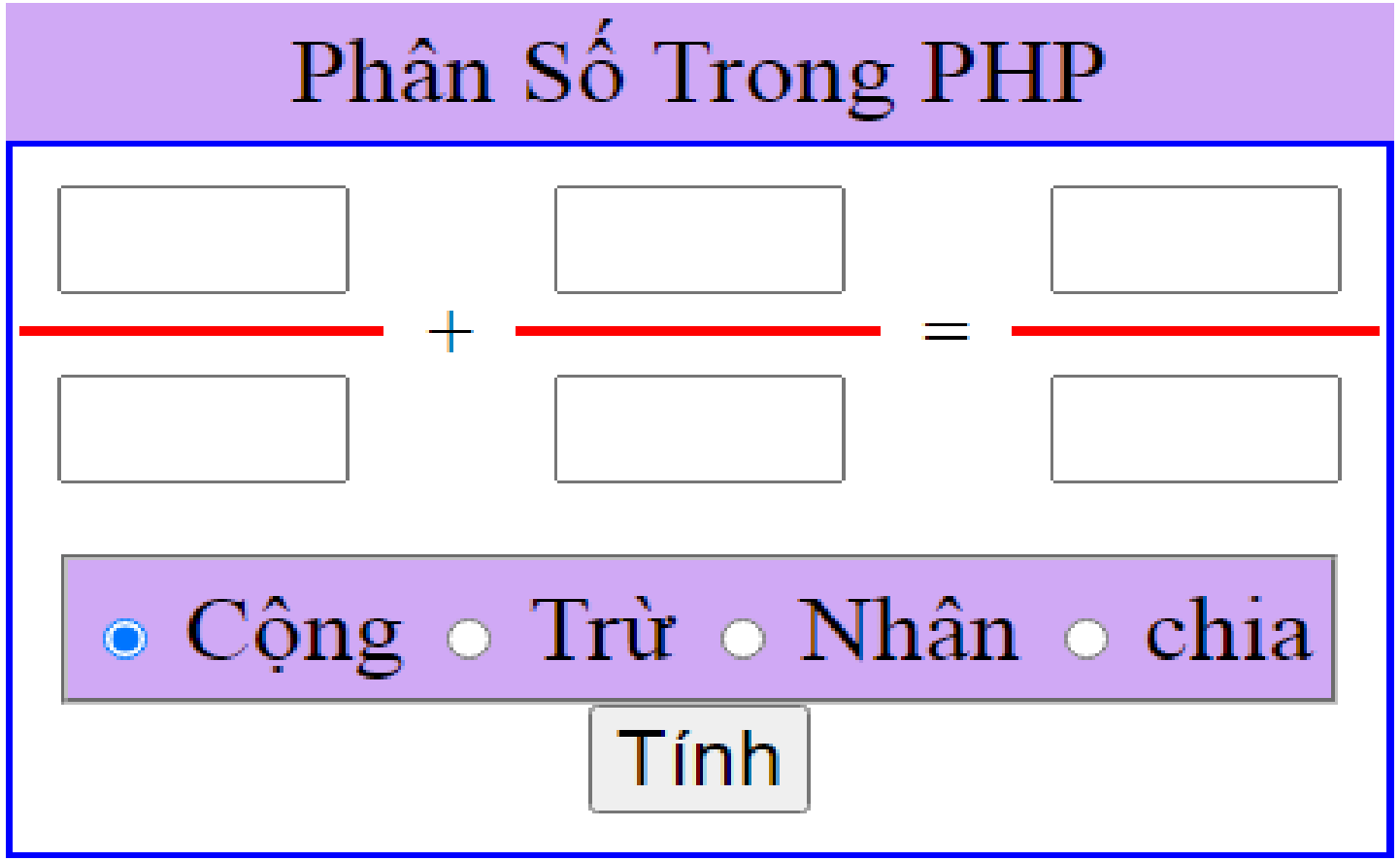
**Yêu cầu thiết kế:**

| Stt | Đối tượng | Yêu cầu | Ghi chú |
|---|---|---|---|
| 1 | Form | – Đặt tên cho Form<br>– Thiết lập phương thức cho Form là post<br>– Và action của Form là tên của trang | |
| 2 | Điều khiển | – Sử dụng điều khiển TextField, Button | |

**Yêu cầu chức năng**

| Stt | Đối tượng | Yêu cầu xử lý chức năng | Ghi chú |
|---|---|---|---|
| 1 | class *phep_tinh* | – Thực hiện: xây dựng class ***phep_tinh*** gồm:<br>o Các thuộc tính: so_thu_nhat và so_thu_hai<br>o Các phương thức gán và lấy giá trị cho các thuộc tính.<br>o Các phương thức tinh_tong(), tinh_hieu(), tinh_tich(), tinh_thuong() của hai số. | |
| 2 | Nút lệnh *Tính* | – Khi chọn, thực hiện:<br>o Kiểm tra giá trị nhập vào và lấy giá trị các số<br>o Gọi sử dụng class phep_tinh<br>o Tính giá trị cho các phép tính cộng, trừ, nhân, chia hai số bằng cách gọi các hàm tương ứng được xây dựng trong class phep_tinh<br>o Tuỳ vào phép tính được chọn mà in ra kết quả như hình trên. | |

# Bài Tập 01




Yêu Cầu: Cho người dùng nhập vào 2 phân số và chọn 1 phép tính và bấm vào nút "Tính" thì tính 2 phân số theo phép tính đã chọn và hiện kết quả.

# Bài Tập 02

**HIỂN THỊ HÌNH ẢNH**

Tiêu đề: Mẫu hoa cưới số 7

Đường dẫn hình: hinh_anh\cuoi_7.jpg

Dòng ghi chú: Happy Wedding!

Chiều rộng: 220 Chiều cao: 240

Đường viền: 1 Canh lề: center

Hiển thị

Kết quả sau khi nhấn ***Hiển thị hình***

**Mẫu hoa cưới số 7**


**Yêu cầu thiết kế:**

| Stt | Đối tượng | Yêu cầu | Ghi chú |
|---|---|---|---|
| 1 | Form | – Đặt tên cho Form<br>– Thiết lập phương thức cho Form là post<br>– Và action của Form là tên của trang | |
| 2 | Điều khiển | – Sử dụng điều khiển TextField, List/Menu, Button | |

**Yêu cầu chức năng**

| Stt | Đối tượng | Yêu cầu xử lý chức năng | Ghi chú |
|---|---|---|---|
| 1 | class *image* | – Thực hiện: xây dựng class ***image*** gồm:<br>o Các thuộc tính: title, src, alt, width, height, border, align<br>o Các phương thức gán và lấy giá trị của các thuộc tính<br>o Phương thức hien_thi_hinh_anh(): dùng để hiển thị hình ảnh với các thuộc tính trên. | |
| 2 | Nút lệnh *Hiển thị hình* | – Khi chọn, thực hiện:<br>o Kiểm tra và lấy giá trị của biến<br>o Gọi sử dụng class image<br>o Gọi hàm hien_thi_hinh_anh() được xây dựng trong class image để hiển thị hình ảnh như trên | |