OOP - Basics

Classes, Properties, Constructors, Objects, Namespaces

greenwich.edu.vn

UNIVERSITY of GREENWICH

Alliance with FPT Education

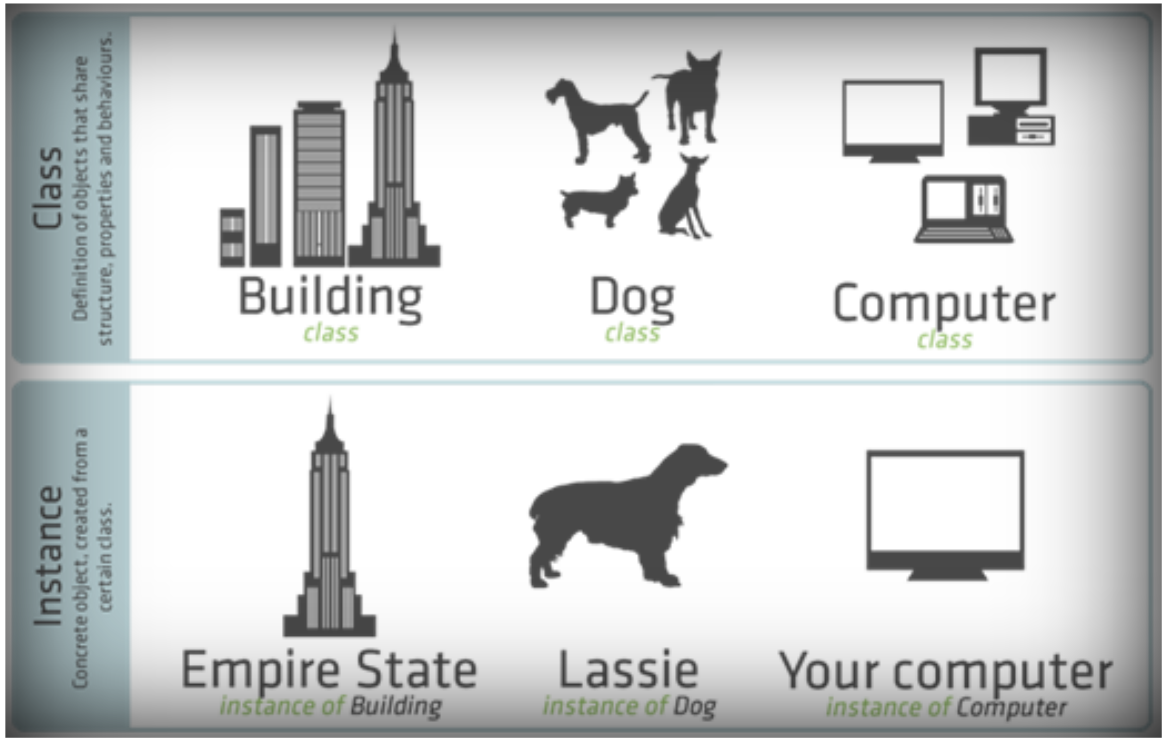
Table of Contents

- Classes and Objects in OOP
- OOP in PHP
  - Define Simple Classes
  - Creating Classes and Objects
  - Using Namespaces

DEFINING SIMPLE CLASSES

Class Versus Instance

- Classes model real-world objects



Classes and Objects

- PHP supports Object-Oriented Programming (OOP)
  - Supports custom classes, objects, interfaces , namespaces, traits
  - Like other OOP languages (C#, Java, C++)

```
class Rock {
    public $height = 12;
    function fall() {
        $this->height--;
    }
}
$myRock = new Rock();
$myRock->fall();
echo $myRock->height; // 11
```

PROPERTIES

Defining and Using Data Properties

Properties

- Properties hold the internal object state
- They have visibility which should be defined at declaration

```
class Dog {
    public $name;
    public $breed;
    public $age;
    public $children;
}
```

Property declarations

CONSTRUCTOR

Defining and Using Class Constructor

Defining Constructor

- "$this" points to the current instance of the class

```
class Person {
  public $name;
  public $age;

   function __construct() {
     $this->name = null;
     $this->age = 0;
   }
}
```

As a rule the constructor should initialize all class properties

Defining Constructor (2)

- The constructor may optionally have parameters

```
class Person {
  public $name;
  public $age;

  function __construct(string $name, $age) {
    $this->name = $name;
    $this->age = $age;
  }
}
```

Constructor with parameters

METHODS

Defining and Using Class Methods

Defining Methods

- Methods are classes own functions and define behavior of the class

```
class Person {
  public $name;
  public $age;

   function printNames(): void {
     echo $this->name . $this->age;
    }
}
```

Simple method with no arguments

"void" return type is available since PHP 7.1

Defining Methods (2)

```
class Person {
  public $name;
  public $age;

  function printNames(string $name): string {
    $this->name = $name;

    return $this->name;
  }
}
```

Method with parameters

Problem: Define a Bird Class

- Create properties
  - age
  - weight
  - flyingSpeed
- Create methods to model bird's behavior
  - breathe()
  - walk()
  - fly()

Solution: Define a Bird Class

```
class Bird {
   private $age;
   private $weight;
   private $flyingSpeed;

   public function __construct($age, $weight,
  $flyingSpeed) {
    $this->age = $age;
    $this->weight = $weight;
    $this->flyingSpeed = $flyingSpeed;
    }
```

Solution: Define a Bird Class

```
    public function walk() {
     echo "Walking" . "\n";
     }
     public function breath() {
     echo "Breathing" . "\n";
     }
     public function fly() {
     echo "Flying" . "\n";
     }
}
```

ANONYMOUS OBJECTS

More on Classes and Objects

Classes and Objects – Example

```
class Student {
    public $name;
    public $age;
    public function __construct($name = null, $age = null) {
        $this->name = $name;
        $this->age = $age;
    }
}

$peter = new Student("Peter", 21);
echo $peter->name;
$peter->age = 25;
print_r($peter); // Student Object ( [name] => Peter [age] => 25 )
$maria = new Student('Maria');
print_r($maria); // Student Object ( [name] => Peter [age] => )
```

Anonymous Objects

- The **stdClass** is an empty generic PHP class for initializing objects of anonymous type (e.g. `$obj = new stdClass;`)
  - It is NOT the base class for objects in PHP
  - Objects can contain their own properties
    - **e.g.** `$obj->prop = value;`

```
$anonCat = new stdClass;
$anonCat->weight = 14;
echo 'My cat weighs ' . $anonCat->weight . ' kg.';
// My cat weighs 14 kg.
```

Anonymous Objects – Example

```
$person = new stdClass;
$person->name = 'Chinese';
$person->age = 43;
$person->weapons = ['AK-47', 'M-16', '9mm-Glock', 'Knife'];
echo json_encode($person);
// {"name":"Chinese","age":43,"weapons":["AK-47","M-
16","9mm-Glock","Knife"]}

$obj = (object)['name' => 'Peter', 'age' => 25];
$obj->twitter = '@peter';
echo json_encode($obj);
// {"name":"Peter","age":25,"twitter":"@peter"}
```

Namespaces

- Namespaces are used to group code (classes, interfaces, functions, etc.) around a particular functionality
  - Better structure for your code, especially in big projects
  - Classes, functions, etc. in a namespace are automatically prefixed with the name of the namespace (e.g. MVC\Models\Lib1\Class1)
- Using a namespace in PHP:

```
use CalculationsManager;
$interest = CalculationsManager\getCurrentInterest();
$mysqli = new \mysqli("localhost", "root", "", "world");
```

Namespaces – Example

```php
<?php
namespace Uni {
    function getTopStudent() {
        return "Pesho";
    }
}

namespace NASA {
    use Uni;
    $topUniStudent = Uni\getTopStudent();
    echo $topUniStudent; // Pesho
}
?>
```

Declares the use of given namespace

Uses a function from the `Uni` namespace

Namespaces – Example (2)

- Define the class in separate file

```
namespace Uni;

class Student {
	public $fname = "Gosho";
	public $lname = "Pesho";

	public function printNames() {
		echo $this->fname . $this->lname . "\n";
	}
}
```

Namespaces – Example (2)

- Include the previous file

```
namespace Uni2;
require_once ‘Uni.php';
use Uni\Student;
function createStudent() {
    $student = new Student('Gosho', 'Petrov');

    return $student;
}
print_r(createStudent());
```

Summary

- Classes define specific structure for objects
  - Objects are particular instances of a class
- Classes define properties, constructor and other members
- Constructor is invoked when creating new class instances and initialize the object's internal state
- PHP supports classes, objects and anonymous objects
- PHP supports namespaces to split program logic into modules