



SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with ‘A’ Grade
Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING-IOT Including CS&BCT

**COURSE NAME : 19SB602 FULL STACK DEVELOPMENT FOR NEXT
GENERATION IOT**

III YEAR / VI SEMESTER

Unit III- CORE PHP WITH MODEL–VIEW–CONTROLLER

**Topic :Classes and Objects, Constructor, Access
Modifiers**



PHP OOP

OOP stands for **Object-Oriented Programming**.

Procedural programming is about writing procedures or functions that perform operations on the data, while object-oriented programming is about creating objects that contain both data and functions.

Object-oriented programming has several advantages over procedural programming:



OOP is **faster and easier to execute**

OOP provides a **clear structure for the programs**

OOP helps to keep the **PHP code DRY "Don't Repeat Yourself"**, and makes the **code easier to maintain**, modify and debug

OOP makes it possible to create **full reusable applications** with less code and shorter development time



PHP - What are Classes and Objects?

Classes and objects are the two main aspects of object-oriented programming.

Look at the following illustration to see the difference between class and objects:



class

Fruit

objects

Apple

Banana

Mango

Another example:

class

Car

objects

Volvo

Audi

Toyota

So, a class is a template for objects, and an object is an instance of a class.

Define a Class

A class is defined by using the **class keyword**, followed by the name of the class and a **pair of curly braces ({}).** All its **properties and methods go inside the braces:**

Syntax

```
<?php  
class Fruit {  
    // code goes here...  
}  
?>
```



Below we declare a class named Fruit consisting of two properties (\$name and \$color) and two methods **set_name()** and **get_name()** for setting and getting the \$name property:

```
<?php
class Fruit {
    // Properties
    public $name;
    public $color;
    // Methods
    function set_name($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
?>
```



Define Objects

Classes are nothing without objects! We can create **multiple objects** from a class. Each object has all the properties and methods defined in the class, but they will have different property values.

Objects of a class are created using the new keyword.

In the example below, \$apple and \$banana are instances of the class Fruit:

Example

```
<?php  
class Fruit {
```



// Properties

public **\$name**;

public **\$color**;

// Methods

function **set_name**(\$name) {

 \$this->name = \$name;

}

function **get_name**() {

 return \$this->name;

}

}



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

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



PHP - The `__construct` Function

A constructor allows you to initialize an object's properties upon creation of the object.

If you **create a `__construct()` function**, PHP will **automatically call this function when you create an object from a class**.

Notice that the construct function starts with **two underscores (`__`)!**

We see in the example below, that using a constructor saves us from calling the `set_name()` method which reduces the amount of code:



Example

```
<?php
class Fruit {
    public $name;
    public $color;

    function __construct($name) {
        $this->name = $name;
    }
    function get_name() {
        return $this->name;
    }
}
$apple = new Fruit("Apple");
echo $apple->get_name();
?>
```



PHP - Access Modifiers

Properties and methods can have access modifiers which control where they can be accessed.

There are **three access** modifiers:

public - the property or method can be **accessed from everywhere**.
This is default

protected - the property or method can be **accessed within the class and by classes derived from that class**

private - the property or method can **ONLY be accessed within the class**



Example

```
<?php
class Fruit {
    public $name;
    protected $color;
    private $weight;
}

$mango = new Fruit();
$mango->name = 'Mango'; // OK
$mango->color = 'Yellow'; // ERROR
$mango->weight = '300'; // ERROR
?>
```



Any Query????

Thank you.....