A **Person** class would be define in a file called **Person.php**

**Class properties are declared like this: public $property = ‘value1’;**

A class provides a definition for a user-defined data type. An object is an instance of a class providing value to a class’s properties

The relationship between a class and object is one-to-many. Many objects can be instances of one class

A property or method can have one access modifier:

* public: can be accessed inside or outside
* private: accessible inside the class ONLY
* protected: accessible inside class OR by inherited classes

**Object properties should be declared private to facilitate data encapsulation**

**Object methods should be private if they are only needed by other object methods**

**Class item.php**

**<?php**

**class Item {**

**public $itemID;**

**public $name;**

**public $price;**

**public $description;**

**public $imageFileExt;**

**}**

**?>**

<body>

<?php

**include\_once ‘./classes/Item.php’;**

**$item1 = new Item();**

**$name = $item1->name;**

?>

</body>

$person1 = new Person();

$person1->hello();

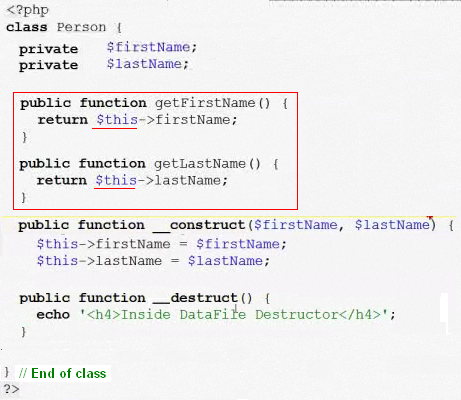
$person1->hello($arg1);

$total = $cart1->calcCartTotal();

$person->setFirstName(‘Joe’);

echo “$person1->getFirstname()”;

**Class person.php**



* Constructors start with **\_\_construct**
* **In PHP it can be only one constructor or none**
* **The destruct is automatically called by PHP when the script ends**
* **The class may be destroyed with $class = NULL; ( $person = NULL ; )**

include\_once ‘./classes/Person.php’;

$person1 = new Person(‘Joe’, “Smith’);

Or

$person1 = new Person(‘’, “’);

$person1->setFirstName(‘Joe”);

$person1->setLastName(‘Smith’);

**GET – SET**

public function \_\_get($name) {

return $this->$name;

}

public function \_\_set( $name , $value ) {

echo “<h4> In \_\_set(): setting<em>\$$name</em></h4>”;

$this->$name = $value;

}

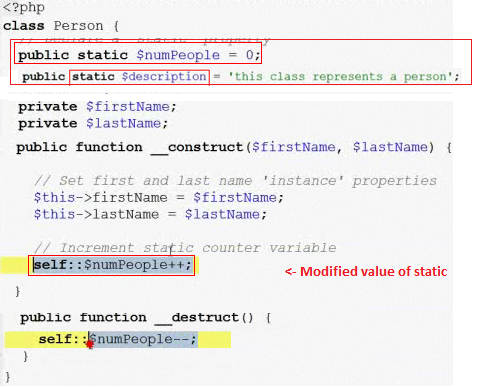
**USING STATIC**

**public static $numPeople = 0;**

* Cannot be accesed using ->
* To access inside the class self::$ echo self::$numPeople;
* To access externally Class::$ echo Person::$numPeople;

**MODIFYING A STATIC (They are not constants)**

**Class person.php**

****

Every time an object is created the constructor increases the $numPeople, and when it is destroyed then the constant is decremented.

|  |  |
| --- | --- |
| **Access** | **Modified** |
| include\_once(‘./classes/Person.php’);  echo Person::$description . “ ‘ <br/> “; | include\_once(‘./classes/Person.php’);  Person::description = “new descrption”; |

**STATIC METHODS**

* Are methods defined in a class that are at the class-level and are not associated with particular instances of a class
* public static function add($opt, $opt2 ) { …
* Because they are not associated with an instance of a class they cannot use the $this variable
* Non-static methods are called instance methods
* Static methods are called using the scope resolution operator (externally using class name & internally using self key

Declared in the class (file Integer.php)

**public static function addIntegers($int1, $int2)**

**{**

**return $int1 + $int2;**

**}**

Called

**<?php**

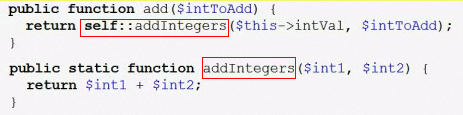
**include\_once(‘./classes/Integer.php’);**

**$int1 = new Integer(1);**

**echo “the sum of 1 and 1 = “ . Integer::addIntegers(1, 1) . ‘ . <br/>’;**

**?>**

Referring a static method inside the same class



**Class constants**

class Math

{

const PI = 3.14;

}

Accessed: without using ‘$’

* Externally: echo Math::PI;
* Internally: echo self::PI;

**Circle.php file**

<?php

class Circle {

const PI = 3.14;

public static function area($radius)

{

return self::PI \* $radius \* radius;

}

?>

<html>

<head>

<title>Class Constant</title>

<style>

Code { font-family: monospace; }

</style>

</head>

<body>

**<?php**

**include\_once(‘./classes/Circle.php’);**

**echo “The value of <code>PI</code> = “ . Circle::PI . ‘</br>”;**

**echo “The area for 2 is “ . Circle::area(2) ;**

**?>**

</body>

</html>

**OUTPUT**

The value of PI = 3.14

The area for 2 is 12.56