**OOP**

**BASICS**

**Introduction**

The 2 most important terms for OOP are class and object:

* A class is a generalised definition of a *thing* – like a blueprint
* An object is a special implementation of that *thing.* – like a house built using the blueprint as a guide

To program using OOP, you design your class and then implement them as objects in your programs when needed.

One of the principles of OOP is modularity: breaking applications into specific subparts/modules, i.e. a class. By separating unrelated (albeit interacting) elements, you can develop code independently, make maintenance and updates less messy, and simplify debugging.

Related to modularity is abstraction**:** classes should be defined broadly, e.g. instead of designing a class for interacting with a MySQL database, you should make one that interacts with a non-specific database. From there, using inheritance and overriding, you would define a more particular class for MySQL.

**Defining a class**

class ClassName{ } //note how the name of a class starts with an uppercase

classes contain properties/attributes and methods; collectively, called *members*. Properties (AKA attributes) must be prefixed with a keyword indicating the variables *visibility* (public, private, protected) – this becomes important when talking about *inheritance*.

Attributes also have to be initialised with a set value (literal), and not the result of an expression.

Public $var = 123; //good

Public $n = $num\*$num; //bad

E.g. class HelloWorld{

//method

Function sayHello($language = “English”){

Echo “<p>”;

Switch($language) {

Case “Iranian”:

Echo “Salam World”;

Break;

Case “French”:

Echo “Bonjour, Monde!”;

Break;

Case “English”:

Default:

Echo “Hello, world!”;

Break;

}

Echo “</p>”;

}//end of method

} //end of class

**Creating an object**

$obj = new ClassName(); // creates an object/instance

$obj -> methodName(); // calls method

$obj -> propName; // calls property

You do not specify the dollar sign ($) when using properties and methods.

To delete an object: unset($obj);

**The *$this* attribute**

The *$this* variable in a class always refers to the current instance (i.e. object involved) of that class.

Within a method, you can refer to the instance of a class and its attributes using the $this->attrName syntax.

E.g. class Rectangle {

Public $width = 0;

Public $height = 0;

//method to set the 2 dimensions

Function setSize($w = 0, $h = 0){

$this->width = $w;

$this->height = $h;

}

Function getArea(){

Return ($this->width \* $this->height);

}

Function getPerimeter(){

Return ( ($this->width + $this->height) \*2);

}

//check if square

Function isSquare(){

If($this->width == $this->height) {

Return true;

}else{

Return false;

}

}

} // end of class

//to use the class

Require(‘rectangle.php’);

$width = 42;

$height = 7;

$r = new Rectangle(); //create obj

$r->setSize($width, $height); //assign values

//call methods

Echo $r->getArea();

Echo $r->getPerimeter();

Echo “This rectangle is ”;

If($r->isSquare()){

Echo “also”;

}else{

Echo “not”;

}

Echo “ a square.”;

Unset($r); // delete obj

Having *get* and *set* methods in a class is a common convention. Set = assign value/s, get = return value/s.

**Creating constructors**

A constructor is a special kind of method that differs from standard ones in 3 ways:

1. Its name is always \_\_construct();
2. It is automatically and immediately called whenever an object of that class is created
3. It cannot have a return statement

A constructor could be used to connect to a database, set cookies, or establish initial values, i.e. whatever should always be done, and done first when an object of this class is made.

A constructor can take arguments, and values for these arguments can be provided when the object is created.

The previous rectangle class can be changed to have a constructor that assigns the dimensions:

//after declaring the attributes, but before any methods, add:

Function \_\_construct($w=0; $h=0){

$this->width = $w;

$this-height = $h;

}

//the object and dimensions can now be set/created in one go:

$r = new Rectangle(12,6);

**Creating destructors**

Whereas a constructor is automatically invoked when an object is created, the destructor is called when the object is destroyed:

Unset($obj); //calls destructor

Destructors cannot take any arguments!

Class ClassName{

//attr. & methods

Function \_\_destruct() {

//code

}

}

E.g. class Demo {

Function \_\_construct() {

Echo “<p>In the constructor</p>”;

}

Function \_\_destruct() {

Echo “<p>In the destructor</p>”;

}

}//end of class

Function test(){

Echo “<p>In the function, creating a new object</p>”;

$f = new Demo();

Echo “<p>Leaving the function</p>”;

}

Echo “<p>Creating a new object globally</p>”;

$o = new Demo();

Echo “<p>Calling the function</p>”;

Test(); //call function

//delete obj

Echo “<p>About to delete the object</p>”;

Unset($o); //delete

Echo “end of script”;

//run the above example to see how the constructor and destructors work and what gets executed first and last etc…

**Unified modelling language (UML)**

UML is a technique used to define a class before coding it, i.e. on paper

It is a box design:

|  |
| --- |
| ClassName |
| Attribute |
| Method() |

<<class name

<< attributes

<<methods

E.g. for the Rectangle class:

|  |
| --- |
| Rectangle |
| Width: *number* = 0  Height: *number* = 0 |
| \_\_construct(width: *number* = 0, height: *number* = 0):void  getSize(width: *number* = 0, height: *number* = 0):void  getArea():number  getPerimeter():number  isSquare(): boolean |

*Italic* = data type, e.g. str

: = what data is returned, this is followed by data type, where void is nothing returned.

Default value is added after = (equals) sign

**Quiz**

1. What is the difference between a class and an object
2. Define what is meant by modularity & abstraction
3. Define a HelloWorld class which says ‘Hello World!’ in different languages
4. Create an instance of the HelloWorld class and call its method/s and properties
5. Create another class which utilises the *$this* variable
6. How do you delete an object
7. What is a constructor and give example
8. What is a destructor and give example
9. Create a class using UML