UNIT – 5
OOP

BCA SEM – 2

WEB PROGRAMMING

Code : CS-09

Presented By : Dhruvita Savaliya

Topics :

- Concept of OOP
  1. Class
  2. Object
  3. Property
  4. Visibility
  5. Constructor
  6. Destructor
  7. Inheritance
  8. Scope Resolution Operator (::)
  9. Autoloading Classes
  10. Class Constants
- Mysql Database handling with oop
  1. Insert
  2. Update
  3. Select
  4. delete

Concept of OOP :

1. Class
2. Object
3. Property
4. Visibility
5. Constructor
6. Destructor
7. Inheritance
8. Scope Resolution Operator (::)
9. Autoloading Classes
10. Class Constants

What is 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:

1. OOP is faster and easier to execute
2. OOP provides a clear structure for the programs
3. OOP helps to keep the PHP code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug
4. OOP makes it possible to create full reusable applications with less code and shorter development time

Object Oriented Concepts :

- **Class** – This is a programmer-defined data type, which includes local functions as well as local data. You can think of a class as a template for making many instances of the same kind (or class) of object.
- **Object** – An individual instance of the data structure defined by a class. You define a class once and then make many objects that belong to it. Objects are also known as instance.
- **Member Variable** – These are the variables defined inside a class. This data will be invisible to the outside of the class and can be accessed via member functions. These variables are called attribute of the object once an object is created.
- **Member function** – These are the function defined inside a class and are used to access object data.
- **Inheritance** – When a class is defined by inheriting existing function of a parent class then it is called inheritance. Here child class will inherit all or few member functions and variables of a parent class.
- **Parent class** – A class that is inherited from by another class. This is also called a base class or super class.
- **Child Class** – A class that inherits from another class. This is also called a subclass or derived class.

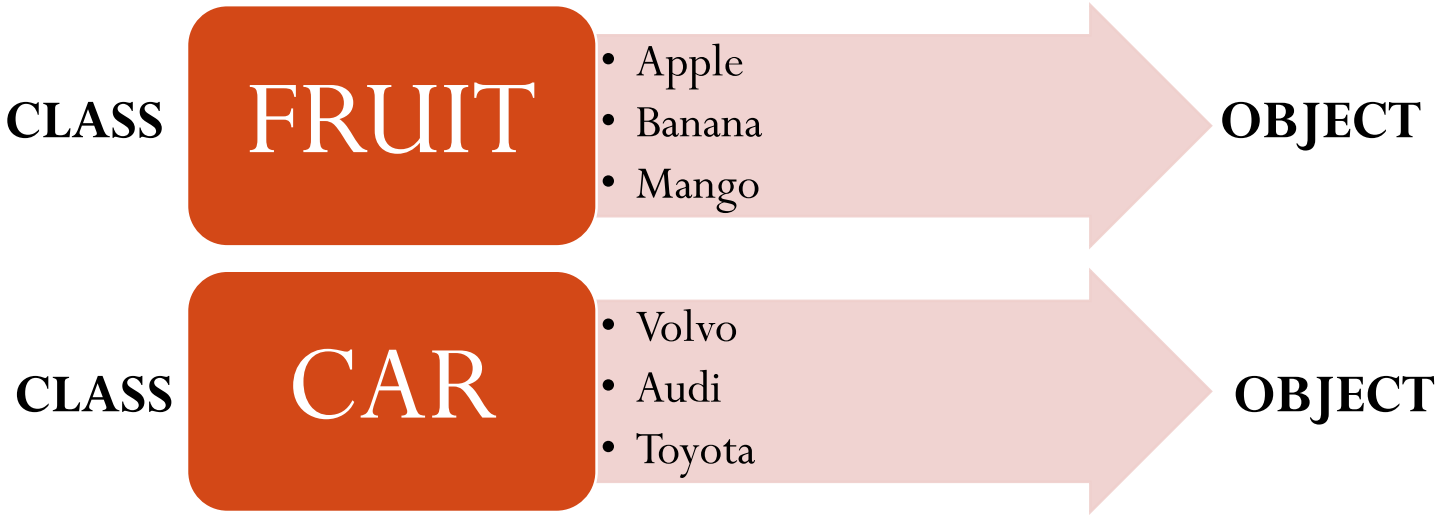
- **Polymorphism** – The word POLY means many. So, the Polymorphism means that the thing is one but the function of that thing is many .
- **Overridden** – There may be possibility that the base and derived class have the function which have same name. It will always call the method of which class object will created.

  To override method the symbol :: is used like –

  Class_name :: function_name();
- **Data Abstraction** – Abstraction means hide. When only the required information is to be given to the user and hide the common information .
- **Encapsulation** – It also means Hide. But here the complex structure of the object is hidden. The whole process is not shown to the user.
- **Constructor** – refers to a special type of function which will be called automatically whenever class object was created.
- **Destructor** – refers to a special type of function which will be called automatically whenever an object is deleted or goes out of scope.

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:



- So, a class is a template for objects, and an object is an instance of a class.
- When the individual objects are created, they inherit all the properties and behaviors from the class, but each object will have different values for the properties.
- Look at the next chapters to learn more about OOP.

- **Class:**
- A class is an entity that determines how an object will behave and what the object will contain. In other words, it is a blueprint or a set of instruction to build a specific type of object.

- In PHP, declare a class using the class keyword, followed by the name of the class and a set of curly braces ({ }).
- This is the blueprint of the construction work that is class, and the houses and apartments made by this blueprint are the objects.

- **Syntax to Create Class in PHP :**

```
<?php
class MyClass
    {
        // Class properties and methods go here
    }
?>
```

- **Object :**
- A class defines an individual instance of the data structure. We define a class once and then make many objects that belong to it.
- Objects are also known as an **instance**.
- We can also said that Object is Blue Print of the class.
- An object is something that can perform a set of related activities.
- Object will create using the **new** keyword.
- **Syntax:**

```
<?php
class MyClass
{
    // Class properties and methods go here
}
$obj = new MyClass;
var_dump($obj);
?>
```

```php
<?php
class demo
{
	private $a= "hello";
	public function display()
	{
			$b="hi";
			echo $this->a;
			echo "<br>".$b;
	}
}
$obj = new demo();
$obj->display();
?>
```

Properties :

- The property of any class is the member of the class.
- There are mainly two members of the class

1. Member variable
2. Member Function

- Note that the member variable or function may be private, public or protected as per requirement.

- Class member variables are called "properties".
- You may also see them referred to using other terms such as "attributes" or "fields", but for the purposes of this reference we will use "properties".
- They are defined by using one of the keywords public, protected, or private, followed by a normal variable declaration.
- This declaration may include an initialization, but this initialization must be a constant value--that is, it must be able to be evaluated at compile time and must not depend on run-time information in order to be evaluated.

- Within class methods non-static properties may be accessed by using -> (Object Operator): $this->property (where property is the name of the property).
- Static properties are accessed by using the :: (Double Colon): self::$property.
- See Static Keyword for more information on the difference between static and non-static properties.
- The variable $this is available inside any class method when that method is called from within an object context.
- $this is a reference to the calling object (usually the object to which the method belongs, but possibly another object, if the method is called statically from the context of a secondary object).

Visibility / Access Modifiers :

- The visibility of a property, a method or (as of PHP 7.1.0) a constant can be defined by prefixing the declaration with the keywords public, protected or private. Class members declared public can be accessed everywhere. Members declared protected can be accessed only within the class itself and by inheriting classes. Members declared as private may only be accessed by the class that defines the member.
- **Property Visibility :**
- Class properties must be defined as public, private, or protected. If declared using var, the property will be defined as public.

Modifier	Class	Sub Class	World
Private	Y	N	N
Protected	Y	Y	N
Public	Y	Y	Y

Constructor :

- A constructor and a destructor are special functions which are **automatically** called when an object is created and destroyed.
- The constructor is the most useful of the two, especially because it allows you to send parameters along when creating a new object, which can then be used to initialize variables on the object.
- the constructor looks just like a regular function, except for the fact that it starts with two underscores.
- In PHP, functions with two underscore characters before the name usually tells you that it's a so-called **magic function**, a function with a specific purpose and extra functionality, in comparison to normal functions.
- So, a function with the exact name "**__construct**", is the constructor function of the class and will be called automatically when the object is created.

- **Example of constructor without using any parameters :**

```
<?php
class Animal
{
        public function __construct()
        {
                echo "I'm alive!<br>";
        }
}
$animal = new Animal();
?>`
```

- **Example of constructor without using any parameters :**

```
<?php
class Animal
{
   public $name;
   public function __construct($name)
   {
         $this->name = $name;
   }
}
$animal = new Animal("Bob the Dog");
echo $animal->name;
?>
```

- Declaring the constructor with parameters is just like declaring a regular function, and passing the parameter(s) is much like calling a regular function, but of course with the "new" keyword that we introduced earlier. A constructor can have as many parameters as you want.

Destructor :

- A destructor is called when the object is destroyed. In some programming languages, you have to manually dispose of objects you created, but in PHP, it's handled by the Garbage Collector, which keeps an eye on your objects and automatically destroys them when they are no longer needed. Like a constructor function you can define a destructor function using function **__destruct()**. You can release all the resources with-in a destructor.

```
<?php
class Animal    {
    public $name="not";
    public function __construct($name) {
            $this->name = $name;
    }
    public function __destruct()            {
            echo "<br>bye bye";
    }
}
$animal = new Animal("Bob the Dog");
echo $animal->name;
?>
```

Inheritance :

- Inheritance is a well-established programming principle, and PHP makes use of this principle in its object model. This principle will affect the way many classes and objects relate to one another.
- For example, when you extend a class, the subclass inherits all of the public and protected methods from the parent class. Unless a class overrides those methods, they will retain their original functionality.
- One of the main advantages of object-oriented programming is the ability to reduce code duplication with inheritance. Code duplication occurs when a programmer writes the same code more than once, a problem that inheritance strives to solve. In inheritance, we have a parent class with its own methods and properties, and a child class (or classes) that can use the code from the parent. By using inheritance, we can create a reusable piece of code that we write only once in the parent class, and use again as much as we need in the child classes.
- we use the **extends** keyword.

```php
<?php
class Foo
{
    public function printItem($string)
    {
        echo 'Foo: ' . $string.'<br>';
    }
    public function printPHP()
    {
        echo 'PHP is great.'.'<br>';
    }
}
```

```php
class Bar extends Foo
{
    public function printItem1($string)
    {
        echo 'Bar: ' . $string.'<br>';
    }
}
$foo = new Foo();
$bar = new Bar();
$foo->printItem('hii');
$foo->printPHP();
$bar->printItem('hello');
$bar->printItem1('hello 1');
$bar->printPHP();
?>
```

Scope Resolution Operator (: :)

- The Scope Resolution Operator (also called Paamayim Nekudotayim) or in simpler terms, the double colon, is a token that allows access to static, constant, and overridden properties or methods of a class.
- When referencing these items from outside the class definition, use the name of the class.
- It is used to refer to blocks or codes in context to classes, objects, etc. An identifier is used with the scope resolution operator.
- The most common example of the application of the scope resolution operator in PHP is to access the properties and methods of the class.
- As of PHP 5.3.0, it's possible to reference the class using a variable.
- The variable's value can not be a keyword (e.g. self, parent and static).

- **Example 1:** This type of definition is used while defining constants within a class.

```
<?php
class democlass {
    const PI = 3.14;
}
echo democlass::PI;
?>
```

- **Example 2:** Three special keywords self, parent, and static are used to access properties or methods from inside the class definition.

```
<?php
// Declaring parent class
class demo{
        public static $bar=10;
        public static function func(){
        echo static::$bar.“<br>";
    }
}
// Declaring child class
class Child extends demo{
        public static $bar=20;
  }
// Calling for demo version of func()
demo::func();
// Calling for child's version of func()
Child::func();
?>
```

Autoloading Classes :

- In order to use a class defined in another PHP script, we can incorporate it with include or require statements.
- However, PHP's autoloading feature does not need such explicit inclusion. Instead, when a class is used (for declaring its object etc.) PHP parser loads it automatically, if it is registered with **spl_autoload_register()** function.
- Any number of classes can thus be registered.
- This way PHP parser gets a last chance to load the class/interface before emitting an error.

- **File 1 :** with name – autoaload.php

```
<?php
spl_autoload_register(function ($class_name) {
 include $class_name . '.php';
});
$obj = new MyClass1();
$obj->bca();
?>
```

- **File 2 :** with name – Myclass1.php

```
<?php
class MyClass1
{
    public function __construct()
    {
            echo "hello";
    }

    public function bca()
    {
            echo "<br>hello bca";
    }
}
?>
```

Class Constants :

- A constant is, just like the name implies, a variable that can never be changed. When you declare a constant, you assign a value to it, and after that, the value will never change. Normally, simple variables are just easier to use, but in certain cases constants are preferable, for instance to signal to other programmers (or your self, in case you forget) that this specific value should not be changed during runtime.
- It is possible to define constant values on a per-class basis remaining the same and unchangeable. Constants differ from normal variables in that you don't use the $ symbol to declare or use them. The default visibility of class constants is public.
- The value must be a constant expression, not (for example) a variable, a property, or a function call.
- It's also possible for interfaces to have constants. Look at the interface documentation for examples.
- As of PHP 5.3.0, it's possible to reference the class using a variable. The variable's value can not be a keyword (e.g. self, parent and static).
- Note that class constants are allocated once per class, and not for each class instance.
- Note that constants does not have a visibility modifier.

- **Accessing Class Constants**
- There are two ways to access class constants.

1. Inside the class:
   The self keyword and Scope Resolution Operator (::) is used to access class constants from the methods in a class.

```
public function greet() {
    echo self::GREET;
}
```

2. Outside the class:
   The class name and constant name is used to access a class constant from outside a class.

```
echo Welcome::GREET;
```

```
<?php
class Welcome {
    const GREET = 'Have Nice Day';
    public function fun() {
        echo self::GREET;
    }
}
$welcome = new Welcome();
$welcome -> fun();

echo "<br>";
echo Welcome::GREET;
?>
```

Mysql Database handling with oop :

1. Insert
2. Update
3. Select
4. delete