Web Development with PHP
Practical - 7
OOP with PHP -1

OOP in PHP

Object-oriented programming (OOP) in PHP can be said as a programming model that is based on the concept of objects, which contain data in the form of attributes or properties, and methods, which are functions that can be performed on the object. PHP is a popular server-side scripting language that supports OOP. In PHP, classes are used to define objects, and objects can be created from classes using the "new" keyword.

Encapsulation, inheritance, and polymorphism are key concepts of OOP that are supported in PHP. Encapsulation provides data hiding and abstraction, allowing for more secure and flexible code. Inheritance allows for the creation of new classes that can inherit properties and methods from existing classes, saving time and effort in code development.

Object-Oriented Concepts and Principles

- Encapsulation: This is the concept of bundling data and behavior within an object and hiding the details from the outside world. In PHP, this is typically achieved through the use of access modifiers such as public, private, and protected.
- Inheritance: This is the ability to create new classes from existing ones, inheriting properties and methods. In PHP, inheritance is implemented using the "extends" keyword.
- Polymorphism: This is the ability to create multiple methods with the same name that behave differently depending on the context. In PHP, polymorphism is typically achieved through the use of interfaces and abstract classes.
- Abstraction: This is the concept of defining a simplified interface for complex functionality. In PHP, this can be achieved through the use of interfaces, abstract classes, and traits.

Classes and Objects

Classes and objects are key components of OOP. A class is a blueprint or template for creating objects, while an object is an instance of a class. In PHP, a class is defined using the "class" keyword, followed by the class name and a block of code containing properties (data) and methods (functions).

For example:

```
class Car {
  public $color;
  public function StartEngine() {
    echo "The engine has started.";
  }
}
```

```
//To create an object from this class, the "new"
keyword is used
$myCar = new Car();

//----------------------------------------

// Define a class
class Person {
  public $name;
  public $age;
}

// Create an object
$person = new Person();

// Set the properties of the object
$person->name = "Mahi Agarwal";
$person->age = 24;

// Access the properties of the object
echo $person->name; // Output: Mahi Agarwal
echo $person->age; // Output: 24
```

Constructor Functions

A constructor function is a special method that is called automatically when an object is created. The constructor function is used to initialize the properties of the object and set its initial state. To define a constructor function in PHP, you use the "__construct()" method.

In PHP, we can pass arguments to the constructor of a class when creating an object. The constructor is a special method that is called when an object is created and is used to initialize the object's properties.

```
// Define a class with a constructor
class Person {
  public $name;
  public $age;

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

```
// Create an object with arguments
$person = new Person("Mahi Agarwal", 24);

// Access the properties of the object
echo $person->name; // Output: Mahi Agarwal
echo $person->age; // Output:24
```

Calling Member Functions

Member functions (also called methods) are functions that are defined within a class and can be called on objects of that class.

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

  public function sayHello() {
     echo "Hello, my name is ". $this->name . " and I
am "  $this->age . " years old.";
  }
}

// Create an object of the Person class
$person = new Person();
$person->name = "John Doe";
$person->age = 30;

// Call the sayHello() method on the object
$person->sayHello(); // Output: Hello, my name is
John Doe and I am 30 years old.
```

Destructor

Destructor function is a special method that is automatically called when an object is no longer needed and is about to be destroyed. The destructor function is used to perform cleanup tasks before the object is removed from memory. To define a destructor function in PHP, you use the "__destruct()" method.

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

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

```
  }

  public function sayHello() {
    echo "Hello, my name is " . $this->name . " and I
am " . $this->age . " years old.";
  }

  public function __destruct() {
    echo "Goodbye, " . $this->name . "!";
  }
}

// Create an object of the Person class using the
constructor
$person = new Person("Mahi Agarwal", 24);

// Call the sayHello() method on the object
$person->sayHello(); // Output: Hello, my name is
Mahi Agarwal and I am 24 years old.

// Destroy the object
unset($person); // Output: Goodbye, Mahi Agarwal!
```

Inheritance

Inheritance is an important concept in PHP that allows you to create new classes based on existing classes. In PHP, inheritance is supported by the "extends" keyword, which allows a new class to inherit the properties and methods of an existing class.

To create a class that inherits from an existing class, you define a new class using the "class" keyword and include the name of the class that you want to inherit from using the "extends" keyword.

```
class Animal {
  public $name;
  public function makeSound() {
    echo "Animal is making a sound.";
  }
}

class Dog extends Animal {
  public function makeSound() {
    echo "Dog is barking.";
  }
}
```

```
// Create an object of the Dog class
$dog = new Dog();
$dog->name = "Fido";

// Call the makeSound() method on the Dog object
$dog->makeSound(); // Output: Dog is barking.
```

Function Overriding

Function overriding is a concept in object-oriented programming that allows a subclass to provide a different implementation of a method that is already defined in its parent class. In PHP, function overriding is achieved by defining a method with the same name and parameters in the subclass as the method in the parent class.

```
class Animal {
  public function makeSound() {
    echo "The animal makes a sound.";
  }
}

class Dog extends Animal {
  public function makeSound() {
    echo "The dog barks.";
  }
}

// Create an object of the Dog class
$dog = new Dog();

// Call the makeSound() method on the Dog object
$dog->makeSound(); // Output: The dog barks.
```

<u>Overriding in PHP:</u>

- Overriding with access modifiers
- Overriding with final keyword
- Final Class Overriding

Public Members

In PHP, public members are properties or methods of a class that can be accessed from outside the class. This means that any code that has access to an instance of the class can read or modify the public members. To define a public member in PHP, you simply declare it using the "public" keyword.

Private members

Private members in PHP are properties and methods that are only accessible within the class in which they are defined. This means that they cannot be accessed or modified from outside the class. To define a private member in PHP, you use the "private" keyword before the property or method name.

Protected members

Protected members in PHP are class members (properties and methods) that can only be accessed within the class itself and its subclasses. To define a protected member in PHP, you use the "protected" keyword before the member name.

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

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

  protected function getAge() {
    return $this->age;
  }
}

class Employee extends Person {
  private $salary;

  public function __construct($name, $age, $salary) {
    parent::__construct($name, $age);
    $this->salary = $salary;
  }

  public function getSalary() {
    return $this->salary;
```

```
  }

  public function getAge() {
     // Call the parent's getAge() method to access
the protected member
    return parent::getAge();
  }
}

// Create an object of the Employee class
$employee = new Employee("Mahi Agarwal", 24, 70000);

// Call the getAge() method on the object (inherited
from the parent class)
echo $employee->getAge(); // Output: 24

// Call the getSalary() method on the object (defined
in the subclass)
echo $employee->getSalary(); // Output: 70000
```

Calling Parent Constructors

In PHP, you can call a parent class's constructor using the parent::__construct() method. This is useful when you have a child class that needs to perform some additional initialization that builds upon the work of the parent constructor.

```
class ParentClass {
  public function __construct() {
    // Parent constructor logic here
  }
}

class ChildClass extends ParentClass {
  public function __construct() {
    parent::__construct(); // Call parent constructor
    // Child constructor logic here
  }
}
```

Exercise

1. Create a HTML form which takes a bank account number, current balance, amount value for deposit or withdrawal operation. It is two button withdrawal and deposit. On click on button it performs a respective operation . Create a bank class having a property account number and amount and methods for deposit, withdrawal and for displaying the account number and current amount. On clicking on the withdrawal or deposit button it creates an object of the bank class and should call the respective method of a class. And display the account no and current available amount
2. Create a class Food. Having a property name, category and price. Create a constructor which assigns the values to the properties. Class should have a method order() which takes the quantity and calculates the total price. Display() will display all food details as well as total price of a food item. getTotalPrice() return the total price of each food item (price*qty) (*qty and total price are not class variables). Give all necessary input by the HTML form. And display output in HTML form. Take the necessary input of three food items from HTML form. [3 objects of a class should be created] . Create a php function which calculates the final sum of all the object's total price and display.
3. Create a class course having properties coursename,noofyear. Create setter and getter public methods for each property. Create a student class which inherits the course class. Student class should have name, passout year, resultclass properties . Student class contain public method setvalue() which set all the properties value. And display() which display all the values. Create four different object of student class and display details of each object.
4. Create a class vehicle having properties prodyear,company name and a protected method for setting and getting values. Create subclass TwoWheeler inherits from vehicle. Properties are nameofvehicle, color. It contains methods setvalues() and getvalues() which set the values of all properties and display it respectively. Create a subclass FourWheeler inherits from vehicle class. Properties are vehiclename, color, price, tolltaxrate . It contains constructor which set all the properties. Display() method for displaying the details. Create an object of TwoWheeler and FourWheeler and display all the details for both the objects.