HAWASSA UNIVERSITY

Department of *Computer science*

WEB PROGRAMMING

Group assignment

GROUP MEMBERS: ID:

1 .Ferhan Husein	0573/15
2. Fenan Gadisa	0568/15
3. Tadios Tsegaye	1342/15
4. Abdreshikur Ahmed	0017/15
5. Debela Jobir	0411/15
6. Abduwahid Sultan	0027/15

Submitted: Gizaw

Object-Oriented Programming (OOP) in PHP

- Object-Oriented Programming (OOP) is a programming paradigm that organizes software design around "objects" rather than "functions" and "logic." Objects are self-contained units that combine data (properties) and behavior (methods). OOP aims to increase the flexibility and maintainability of programs, making them easier to understand, debug, and extend. 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. development time The "Don't Repeat Yourself" (DRY) principle is about reducing the repetition of code. You should extract out the codes that are common for the application, and place them at a single place and reuse them instead of repeating it.

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

A. How to Create or Define a Class

In PHP, a class is a blueprint or a template for creating objects. It defines the properties (variables) and methods (functions) that an object will have.

To define a class, you use the **`class`** keyword followed by the class name. Class names typically follow Pascal Case (**e.g., `MyClass`, `UserService`).**

Syntax:

```
<?php
class ClassName {
   // Properties (variables)
   public $propertyName;
   private $anotherProperty;
   // Methods (functions)
   public function methodName() {
      // Method implementation}
   private function anotherMethod() {
      // Method implementation   }}?>
```

B. Creating Objects

An object is an instance of a class. Once you define a class, you can create multiple objects from that class. Each object will have its own set of properties and can call the methods defined in the class.To create an object, you use the `new` keyword followed by the class name.

Syntax:

PHP

```
<?php
$objectName = new ClassName();
?>
```

The $this keyword refers to the current object, and is only available inside methods can we change the value of the object property we can do this in two ways:

1, the class (by adding a set name() method and use $this):

Example :-

```
<?phC
class Fruit {
  public $name;
  function set_name($name) {
    $this->name = $name;}}
$apple = new Fruit();
$apple->set_name("Apple");
 echo $apple->name;?>
```

2,Outside the class (by directly changing the property value):

Example:

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

```
$apple = new Fruit();

$apple->name = "Apple";

echo $apple->name;

?>
```

C. Object-Oriented Programming Principles in PHP

OOP is built upon several core principles that help in designing robust and scalable applications.

1. **Encapsulation:**
    - **Concept:** Encapsulation is the bundling of data (properties) and methods (functions) that operate on the data into a single unit (a class). It also involves restricting direct access to some of an object's components, which is achieved through access modifiers.
    - **In PHP:** PHP uses access modifiers (`public`, `protected`, `private`) to control the visibility of properties and methods.
        - `public`: Accessible from anywhere.
        - `protected`: Accessible only within the class itself and by inheriting classes.
        - `private`: Accessible only within the class that defines it.

      It is use to Protects data integrity by preventing external code from directly manipulating an object

      Exmple:

```
<?php

class Account {

   private $balance = 0; // Encapsulated data

   public function deposit($amount) {

      if ($amount > 0) {

         $this->balance += $amount;

      }   }

   public function getBalance() { // Public method to access private data

      return $this->balance;
```

```
    }}

$myAccount = new Account();

$myAccount->deposit(100);

// $myAccount->balance = 500; // This would cause an error if balance were private and attempted directly

echo "Current Balance: " . $myAccount->getBalance();

?>'s internal state.
```

2. Inheritance:

- **Concept:** Inheritance allows a new class (subclass or child class) to inherit properties and methods from an existing class (superclass or parent class). This promotes code reusability.
- **In PHP:** Achieved using the `extends` keyword. A child class can add new properties/methods or override existing ones. PHP supports single inheritance (a class can only extend one other class).

  It reduces code duplication and establishes a "is-a" relationship between classes (e.g., a `Dog` "is a" `Animal`).

  Example:

```
<?php

class Animal {

    public function eat() {

        echo "Animal is eating.<br>";   }}

class Dog extends Animal { // Dog inherits from Animal

    public function bark() {

        echo "Woof woof!<br>";  }}

$myDog = new Dog();

$myDog->eat();  // Inherited method

$myDog->bark(); // Dog's own method
```

```
?>
```

3.Polymorphism:

- **Concept:** Polymorphism means "many forms." In OOP, it allows objects of different classes to be treated as objects of a common superclass. It enables a single interface to represent different underlying forms or types.
- **In PHP:** Achieved through method overriding (when a subclass provides a specific implementation for a method that is already defined in its superclass) and interfaces.

  Used to increases flexibility and extensibility. Code can be written to work with a general type, and at runtime, the specific implementation appropriate for the actual object type is invoked.

Example:

```
<?php

interface Shape {

    public function calculateArea();}

class Circle implements Shape {

    private $radius;

    public function __construct($radius) {

        $this->radius = $radius;}

    public function calculateArea() {

        return M_PI * $this->radius * $this->radius;}}

class Rectangle implements Shape {

    private $width;

    private $height;

    public function __construct($width, $height) {

        $this->width = $width;
```

```
        $this->height = $height; }


    public function calculateArea() {

        return $this->width * $this->height;

    }}

function printArea(Shape $shape) { // Polymorphic function

    echo "Area: " . $shape->calculateArea() . "<br>";

}

$circle = new Circle(5);

$rectangle = new Rectangle(4, 6);

printArea($circle);    // Calls Circle's calculateArea()

printArea($rectangle); // Calls Rectangle's calculateArea()

?>
```

Abstraction:

- **Concept:** Abstraction focuses on showing only essential information and hiding complex implementation details. It defines what an object does, rather than how it does it.
- **In PHP:** Achieved using abstract classes and interfaces.
    - **Abstract Classes:** Classes that cannot be instantiated on their own and must be extended by other classes. They can contain abstract methods (methods declared without an implementation) and concrete methods.
    - **Interfaces:** Define a set of methods that a class must implement. They provide a contract for classes.
- Simplifies complex systems by breaking them down into manageable, understandable parts. Promotes loose coupling.

Example:

```
 <?php

abstract class Vehicle { // Abstract class

    protected $speed;
```

```
    public function __construct($speed) {
        $this->speed = $speed; }
    abstract public function drive(); // Abstract method - must be implemented by subclasses
    public function getSpeed() { // Concrete method
        return $this->speed;  }}
class Car extends Vehicle {
    public function drive() {
        echo "The car is driving at " . $this->speed . " km/h.<br>";   }}
class Motorcycle extends Vehicle {
    public function drive() {
        echo "The motorcycle is riding at " . $this->speed . " km/h.<br>";
    }}
$myCar = new Car(120);
$myCar->drive();


$myMotorcycle = new Motorcycle(90);
$myMotorcycle->drive();
?>
```

Generally: These four principles – Encapsulation, Inheritance, Polymorphism, and Abstraction – form the foundation of Object-Oriented Programming and are crucial for developing robust, maintainable, and scalable applications in PHP.