PHP OOP

OBJECT-ORIENTED PROGRAMMING

This project is funded by the European Union

UNDP

Maharah
Code Your Future

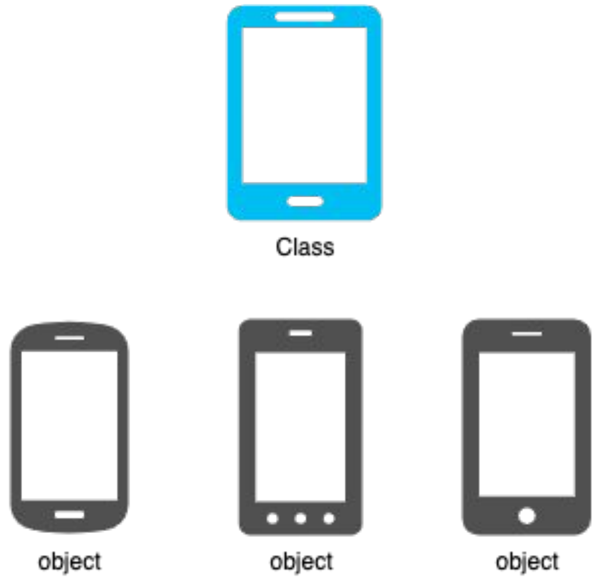
LibyanSpider

Object-Oriented Programming (OOP)

Object-Oriented Programming (OOP) is a programming that organizes code into reusable objects.

- A class is a blueprint for creating objects. It defines the properties (attributes) and behaviors (methods) that the objects of that class will have.
- An object is an instance of a class. It represents a real-world entity and can have its own unique state and behavior.

OOP Example



Properties			
Brand	Samsung	Huawei	Apple
Model	S32 Ultra	Mate 50	iPhone 15
OS	Android	HarmonyOS	iOS

Methods
Make Call
Send Text
Take Photo

Maharah
Code Your Future

Declaring Classes and Instantiating Objects

In PHP, classes are created using the **class** keyword, and objects are created using the **new** keyword.

- Classes can be named using the same rules as variables.
- If you are not passing constructor parameters you can omit the brackets
- Classes should be defined before they are used.

```php
<?php
class Maharah {
   // properties and methods
}
$maharahObject = new Maharah(); //or new Maharah;
```

Maharah
Code Your Future

Visibility or Access Modifiers

The visibility of a method or property can be set by prefixing the declaration with public, protected, or private.

- **Public** class members can be accessed from anywhere.
- **Protected** class members can be accessed from within the class and by its children.
- **Private** class members can only be accessed from within the class itself.

Notes:

- If you don't explicitly specify a visibility then it will default to public.
- Interfaces can only include public methods.
- Any class that implements the interface must match the visibility of the method.

Instance Properties and Methods

Concrete objects that you create from classes are also known as instances.
When you create an object from a class, you are said to instantiate the object.

- Properties
- Methods

Maharah
Code Your Future

Properties

Properties are variables that hold the state of an object, while methods are functions that define the behavior of an object.
Class properties are declared by using one of the visibility modifiers followed by the name of the property. Property names follow the same naming rules as variables.

```php
<?php
class Car {
   public $color; // property
}
```

```php
<?php
class Car {
  public string $color; // property
}
```

Maharah
Code Your Future

Static Properties

Static properties are declared with the static keyword and can be accessed with the scope resolution operator.

```php
<?php
class Foo
{
   public static $message = 'Hello World';
}
echo Foo::$message; // Hello World
```

Maharah
Code Your Future

Methods

Methods are functions within a scope construct. They are declared in a function by using a visibility modifier followed by the function declaration. If you omit a visibility modifier, the method will have public visibility.

```php
<?php
class Car {
   public $color; // property

   public function startEngine() { // method
       // code to start the engine
   }
}
```

Maharah
Code Your Future

Static Methods

Declaring a method or property as static makes it available without needing a concrete implementation of the class.

```php
<?php
class Car
{
   public static $color = 'blue';

   public static function getColor()
   {
       echo "Car color: " . self::$color;
   }
}
Car::getColor();
```

Maharah
Code Your Future

Class Constants

- A constant is a value that is immutable.
- Class constants do not change between instances of the class. All objects created from that class have the same value for the class constant.
- Class constants follow the same naming rules as variables but do not have an $ symbol prefixing them. By convention, constant names are declared in uppercase.

Let's consider an example:

```php
<?php
class MathsHelper
{
   const PI = 3.14;
}
echo MathsHelper::PI; // 3.14
```

Constructors and Destructors

Constructors and destructors are special methods in a class.

- **Constructor**: It is called automatically when an object is created. It is used to initialize object properties or perform any necessary setup.
- **Destructor**: It is called automatically when an object is destroyed. It is used to clean up resources or perform any necessary cleanup actions.

```php
<?php
class Car {
   public $color;

   public function __construct($color) {
       $this->color = $color;
       echo "Car constructed with color: " . $this->color;
   }

   public function __destruct() {
       echo "Car destroyed.";
   }
}
$myCar = new Car("red"); // Output: Car constructed with color: red
unset($myCar); // Output: Car destroyed.
```

Maharah
Code Your Future

Constructor Parameters

If a class constructor takes a parameter, you need to pass it in when instantiating an instance of the class.

```php
<?php
class User
{
   public string $name;
   public function __construct(string $name)
   {
       $this->name = $name;
   }
}
$user = new User('Maharah');
```

Maharah
Code Your Future

Question?

01

Which of these is not a valid php class name?

- exampleClass
- Example_Class
- Example_1_Class
- 1_Example_Class
- They are all valid class names

Maharah
Code Your Future

Benefits of OOP

Using Object-Oriented Programming (OOP) in PHP provides several significant benefits, especially for large, complex, or scalable applications. Here's why OOP is often preferred:

Modularity

OOP allows developers to break down problems into smaller, manageable parts (objects). This modularity makes it easier to maintain and update code.

Reusability

Through inheritance, classes can be extended, allowing new functionalities to be added without duplicating code. This promotes code reusability.

Encapsulation

Encapsulation helps in bundling the data with the methods that operate on the data. It hides the internal state of objects and only exposes what is necessary. This leads to a reduction in system complexity and increases robustness.

Abstraction

OOP provides the ability to create abstract layers over your code, making it more understandable and reducing complexity. You can hide detailed implementation and show only the relevant features of an object.

Polymorphism

Polymorphism allows methods to do different things based on the object they are acting upon. This flexibility can lead to more dynamic and flexible code.

Maintainability

OOP makes code more manageable and easier to debug or update, which is crucial for large applications.

Improved Collaboration

In team environments, OOP's structured approach allows multiple developers to work on different parts of the same project with clearer interfaces and interactions.

Design Pattern Usage

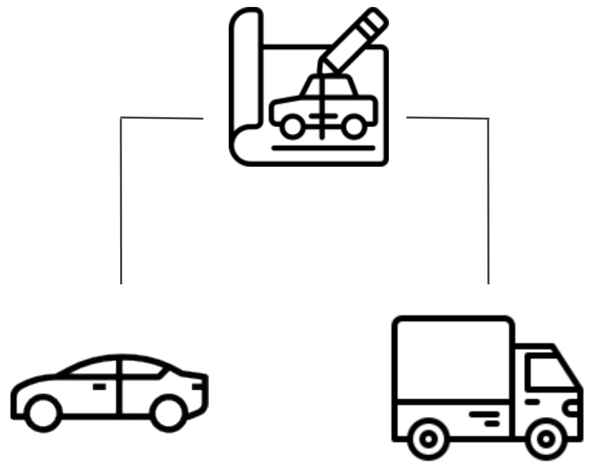
Many modern software design patterns are based around OOP principles. Using OOP enables the implementation of powerful design patterns like Singleton, Factory, Strategy, etc.

Better Data Handling

With OOP, handling and managing data becomes more efficient, especially in cases like database connections and querying.

Maharah
Code Your Future

Inheritance



Vehicle (Parent)	Car (Child)	Truck (Child)
Properties		
numberOfWheels	⇦	⇦
color	⇦	⇦
maxSpeed	⇦	⇦
	seatCount	
	airbags	
		cargoCapacity
		trailerHook
Methods		
start()	⇦	⇦
stop()	⇦	⇦
accelerate()	**accelerate()**	⇦
	openTrunk()	
		attachTrailer()

Maharah
Code Your Future

Inheritance

PHP supports inheritance in its object model. If you extend a class then the child class will inherit all of the non-private properties and methods of the parent class. In other words, the child will have the public and protected elements of the parent class. You can override them in the child class, but they will otherwise have the same functionality.

PHP does not support inheriting from more than one class at a time.

The **extends** keyword is used to create a subclass (child class) that inherits from a parent class.

The subclass inherits all the **public** and **protected** properties and methods of the parent class.

Maharah
Code Your Future

Inheritance (2)

```php
<?php
class User {
  public function __construct(protected string $username) {
  }
   public function login() {
      echo "user {$this->username} is logged in.\n";
  }
}

class Admin extends User {
  public function createUser($username) {
      echo "user {$username} created!\n";
  }
}

class Author extends User {
  public function writeArticle($subject, $content) {
      echo "article {$subject} written!\n";
  }
}
// Usage
$admin = new Admin('admin1');
$admin->login();
$admin->createUser('user3');
```

Maharah
Code Your Future

Overriding

- A child class may declare a method with the same name as the parent class,
- The method parameter signature in the child must be like the parent.
- The child declaration needs to be compatible with the parent:

```php
<?php
class Employee
{
   public function calculate(float $hourlyRate, float $numHoursWorked)
   {
       return $hourlyRate * $numHoursWorked;
   }
}
class Oops extends Employee
{
   public function calculate(float $hourlyRate, float $numHoursWorked)
   {
       return $hourlyRate + $numHoursWorked;
   }
}
```