PHP Object Oriented Programming

PENGENALAN OOP

Apa itu Object Oriented Programming?

- Object Oriented Programming adalah sudut pandang bahasa pemrograman yang berkonsep “objek”
- Ada banyak sudut pandang bahasa pemrograman, namun OOP adalah yang sangat populer saat ini.
- Ada beberapa istilah yang perlu dimengerti dalam OOP, yaitu: Object dan Class

Apa itu Object?

- Object adalah data yang berisi field / properties / attributes dan method / function / behavior

```
// Definisi kelas
class Mobil {
    // Property
    public $warna;
    public $merk;

    // Method
    public function jalan() {
        echo "Mobil sedang berjalan";
    }
}

// Membuat objek dari kelas Mobil
$mobilSaya = new Mobil();
```

```
// Menetapkan nilai untuk property
$mobilSaya->warna = "Merah";
$mobilSaya->merk = "Toyota";

// Mengakses property
echo "Warna Mobil: " . $mobilSaya->warna . "<br>"; // Output: Wa
echo "Merk Mobil: " . $mobilSaya->merk . "<br>";   // Output: Me

// Memanggil method
$mobilSaya->jalan(); // Output: Mobil sedang berjalan
```

Apa itu Class?

- Class adalah blueprint, prototype atau cetakan untuk membuat Object
- Class berisikan deklarasi semua properties dan functions yang dimiliki oleh Object
- Setiap Object selalu dibuat dari Class
- Dan sebuah Class bisa membuat Object tanpa batas

Membuat Class

- Untuk membuat class, kita bisa menggunakan kata kunci class
- Penamaan class biasa menggunakan format CamelCase

```
<?php
// Membuat class Person
class Person {
    public string $name = "John Doe"; // Default value
    public int $age = 30; // Default value
    public ?string $address = null; // Nullable property
}
?>
```

Membuat Object

- Object adalah hasil instansiasi dari sebuah class
- Untuk membuat object kita bisa menggunakan kata kunci new, dan diikuti dengan nama Class dan kurung ()

```
<?php
// Membuat object dari class Person
$person1 = new Person();
$person2 = new Person();

// Menetapkan nilai ke properties
$person1->name = "Alice Smith";
$person1->age = 25;

$person2->name = "Bob Brown";
$person2->age = 35;
?>
```

Properties

- Fields / Properties / Attributes adalah data yang bisa kita sisipkan di dalam Object
- Namun sebelum kita bisa memasukkan data di fields, kita harus mendeklarasikan data apa aja yang dimiliki object tersebut di dalam deklarasi class-nya
- Membuat field sama seperti membuat variable, namun ditempatkan di block class, namun diawali dengan kata kunci var

Manipulasi Properties

- Fields yang ada di object, bisa kita manipulasi.
- Untuk memanipulasi data field, sama seperti cara pada variable
- Untuk mengakses field, kita butuh kata kunci → setelah nama object dan diikuti nama fields nya

```php
<?php
// Memanipulasi properties
$person1->address = "123 Main St";
?>
```

Properties Type Declaration

- Sama seperti di function, di properties pun, kita bisa membuat type declaration
- Ini membuat PHP otomatis mengecek tipe data yang sesuai dengan type declaration yang telah ditentukan
- Jika kita mencoba mengubah properties dengan type yang berbeda, maka otomatis akan error
- Ingat, bahwa PHP memiliki fitur type juggling, yang secara otomatis bisa mengkonversi ke tipe data lain
- Untuk menambahkan type declaration, kita bisa tambahkan setelah kata kunci var di properties

```php
<?php
// Definisi class dengan type declaration pada properties
class Product {
    public int $id;
    public string $name;
    public float $price;

    // Constructor untuk inisialisasi object
    public function __construct(int $id, string $name, float $pr
        $this->id = $id;
        $this->name = $name;
        $this->price = $price;
    }

    // Method untuk menampilkan informasi produk
```

```
    public function displayInfo() {
        echo "Product ID: {$this->id}, Name: {$this->name}, Pric
    }
}

// Membuat object dari class Product
$product1 = new Product(1, "Keyboard", 49.99);
$product2 = new Product(2, "Mouse", 19.99);

// Memanggil method untuk menampilkan informasi produk
$product1->displayInfo();
$product2->displayInfo();

// Contoh type juggling - mengkonversi float ke int
$product1->price = 50; // Ini akan diubah ke float 50.0 secara o
$product1->displayInfo();
?>
```

Default Properties Value

- Sama seperti variable, di properties juga kita bisa langsung mengisi value nya
- Ini mirip seperti default value, jadi jika tidak diubah di object, maka properties akan memiliki value tersebut

Nullable Properties

- Saat kita menambah type declaration di properties atau di function argument, maka secara otomatis kita tidak bisa mengirim data null ke dalam properties atau function argument tersebut
- Di PHP 7.4 dikenalkan nullable type, jadi kita bisa mengirim data null ke properties atau function arguments
- Caranya sebelum type declaration nya, kita bisa tambahkan tanda ?

FUNCTION

Function

- Selain menambahkan properties, kita juga bisa menambahkan function ke object
- Cara dengan mendeklarasikan function tersebut di dalam block class
- Sama seperti function biasanya, kita juga bisa menambahkan return value dan parameter
- Untuk mengakses function tersebut, kita bisa menggunakan tanda → dan diikuti dengan nama method nya. Sama seperti mengakses properties

This Keyword

- Saat kita membuat kode di dalam function di dalam class, kita bisa menggunakan kata kunci this untuk mengakses object saat ini
- Misal kadang kita butuh mengakses properties atau function lain di class yang sama

Constant

- Properties di class bisa diubah, mirip seperti variable
- Di class juga kita membuat constant, data yang tidak bisa diubah
- Di materi PHP Dasar, kita belajar untuk membuat constant itu perlu menggunakan function define()
- Namun sejak PHP 7.4, kita bisa menggunakan kata kunci const untuk membuat constant, mirip seperti variable, namun tidak menggunakan karakter $

Properties vs Constant

- Saat kita membuat object, properties yang terdapat di class akan secara otomatis dibuat per object, oleh karena itu untuk mengakses properties, kita perlu menggunakan object, atau jika dari dalam object tersebut sendiri, kita perlu menggunakan kata kunci this
- Sedangkan berbeda dengan constant, constant di class tidak akan dibuat per object. Constant itu hidupnya di class, bukan di object, oleh karena itu untuk mengaksesnya kita perlu menggunakan NamaClass::NAMA_CONSTANT

- Secara sederhana, properties akan dibuat satu per instance class (object), sedangkan constant dibuat satu per class

self Keyword

- Jika di dalam class (misal di function) kita ingin mengakses constant, kita perlu mengakses menggunakan NamaClass::NAMA_CONSTANT
- Namun jika di dalam class yang sama, kita bisa menggunakan kata kunci self untuk mempermudah

CONSTRUCTOR & DESTRUCTOR

Constructor

- Saat kita membuat Object, maka kita seperti memanggil sebuah function, karena kita menggunakan kurung ()
- Di dalam class PHP, kita bisa membuat constructor, constructor adalah function yang akan dipanggil saat pertama kali Object dibuat.
- Mirip seperti di function, kita bisa memberi parameter pada constructor
- Nama constructor di PHP haruslah __construct()

Destructor

- Jika constructor adalah function yang akan dipanggil ketika object dibuat
- Destructor adalah function yang akan dipanggil ketika object dihapus dari memory
- Biasanya ketika object tersebut sudah tidak lagi digunakan, atau ketika aplikasi akan mati
- Untuk membuat function destructor, kita bisa menggunakan nama function __destruct()
- Khusus untuk destructor, kita tidak boleh menambahkan function argument
- Dalam penggunaan sehari-hari, ini misal cocok untuk menutup koneksi ke database atau menutup proses menulis ke file, sehingga tidak terjadi memory

leak

INHERITANCE

Inheritance

- Inheritance atau pewarisan adalah kemampuan untuk menurunkan sebuah class ke class lain
- Dalam artian, kita bisa membuat class Parent dan class Child
- Class Child, hanya bisa punya satu class Parent, namun satu class Parent bisa punya banyak class Child
- Saat sebuah class diturunkan, maka semua properties dan function yang ada di class Parent, secara otomatis akan dimiliki oleh class Child
- Untuk melakukan pewarisan, di class Child, kita harus menggunakan kata kunci extends lalu diikuti dengan nama class parent nya.

NAMESPACE

Namespace

- Saat kita membuat aplikasi, bisa dipastikan kita akan banyak sekali membuat class
- Jika class terlalu banyak, kadang akan menyulitkan kita untuk mencari atau mengklasifikasikan jenis-jenis class
- PHP memiliki fitur namespace, dimana kita bisa menyimpan class-class kita di dalam namespace
- Namespace bisa nested, dan jika kita ingin mengakses class yang terdapat di namespace, kita perlu menyebutkan nama namespace nya
- Namespace bagus ketika kita punya beberapa class yang sama, dengan menggunakan namespace nama class sama tidak akan menjadikan error di PHP

Membuat Namespace

- Untuk membuat namespace, kita bisa menggunakan kata kunci namespace
- Jika kita ingin membuat sub namespace, kita cukup gunakan karakter \ setelah namespace sebelumnya

Function dan Constant di Namespace

- Selain class, kita juga menggunakan function dan constant di namespace
- Dan jika kita ingin menggunakan function atau constant tersebut, kita bisa menggunakannya dengan diawali dengan nama namespace nya

Global Namespace

- Secara default saat kita membuat kode di PHP sebenarnya itu disimpan di global namespace
- Global namespace adalah namespace yang tidak memiliki nama namespace

IMPORT

use Keyword

- Sebelumnya kita sudah tahu bahwa untuk menggunakan class, function atau constant di namespace kita perlu menyebutkan nama namespace nya di awal
- Jika terlalu sering menggunakan class, function atau constant yang sama, maka terlalu banyak duplikasi dengan menyebut namespace yang sama berkali-kali
- Hal ini bisa kita hindari dengan cara mengimport class, function atau constant tersebut dengan menggunakan kata kunci use

Alias

- Saat kita menggunakan use, artinya kita tidak perlu lagi menggunakan nama namespace diawal class ketika kita ingin membuat class tersebut
- Namun bagaimana jika kita ternyata nama class nya sama?

- Untungnya PHP memiliki fitur yang namanya alias
- Alias adalah kemampuan membuat nama lain dari class, function atau constant yang ada
- Kita bisa menggunakan kata kunci as setelah melakukan use

Group use Declaration

- Kadang kita butuh melakukan import banyak hal di satu namespace yang sama
- PHP memiliki fitur grup use, dimana kita bisa import beberapa class, function atau constant dalam satu perintah use
- Untuk melakukan itu, kita bisa menggunakan kurung { }

VISIBILITY

Visibility

- Visibility / Access modifier adalah kemampuan properties, function dan constant dapat diakses dari mana saja
- Secara default, properties, function dan constant yang kita buat di dalam class bisa diakses dari mana saja, atau artinya dia adalah public
- Selain public, masih ada beberapa visibility lainnya
- Secara default kata kunci var untuk properties adalah sifatnya public

Access Level

Modifier	Class	Subclass	World
public	Y	Y	Y
protected	Y	Y	N
private	Y	N	N

FUNCTION OVERRIDING

Function Overriding

- Function overriding adalah kemampuan mendeklarasikan ulang function di child class, yang sudah ada di parent class
- Saat kita melakukan proses overriding tersebut, secara otomatis ketika kita membuat object dari class child, function yang di class parent tidak bisa diakses lagi

parent Keyword

- Kadang kita ingin mengakses function yang terdapat di class parent yang sudah terlanjur kita override di class child
- Untuk mengakses function milik class parent, kita bisa menggunakan kata kunci parent
- Sederhananya, parent digunakan untuk mengakses class parent

Constructor Overriding

- Karena constructor sama seperti function, maka constructor-pun bisa kita deklarasikan ulang di class Child nya
- Sebenarnya di PHP, kita bisa meng-override function dengan arguments yang berbeda, namun sangat tidak disarankan
- Jika kita melakukan override function dengan arguments berbeda, maka PHP akan menampilkan WARNING
- Namun berbeda dengan constructor overriding, kita boleh meng-override dengan mengubah arguments nya, namun direkomendasikan untuk memanggil parent constructor

POLYMORPHISM

Polymorphism

- Polymorphism berasal dari bahasa Yunani yang berarti banyak bentuk.
- Dalam OOP, Polymorphism adalah kemampuan sebuah object berubah bentuk menjadi bentuk lain
- Polymorphism erat hubungannya dengan Inheritance

Type Check & Casts

- Sebelumnya kita sudah tau cara melakukan konversi tipe data bukan class
- Khusus untuk tipe data object, kita tidak perlu melakukan konversi secara eksplisit
- Namun agar aman, sebelum melakukan casts, pastikan kita melakukan type check (pengecekan tipe data), dengan menggunakan kata kunci instanceof
- Hasil operator instanceof adalah boolean, true jika tipe data sesuai, false jika tidak sesuai

Abstract Class

- Saat kita membuat class, kita bisa menjadikan sebuah class sebagai abstract class.
- Abstract class artinya, class tersebut tidak bisa dibuat sebagai object secara langsung, hanya bisa diturunkan
- Untuk membuat sebuah class menjadi abstract, kita bisa menggunakan kata kunci abstract sebelum kata kunci class
- Sehingga Abstract Class bisa kita gunakan sebagai kontrak child class

Abstract Function

- Saat kita membuat class yang abstract, kita bisa membuat abstract function juga di dalam class abstract tersebut
- Saat kita membuat sebuah abstract function, kita tidak boleh membuat block function untuk function tersebut
- Artinya, abstract function wajib di override di class child
- Abstract function tidak boleh memiliki access modifier private

ENCAPSULATION

Encapsulation

- Encapsulation artinya memastikan data sensitif sebuah object tersembunyi dari akses luar
- Hal ini bertujuan agar kita bisa menjaga agar data sebuah object tetap baik dan valid
- Untuk mencapai ini, biasanya kita akan membuat semua properties menggunakan access modifier private, sehingga tidak bisa diakses atau diubah dari luar
- Agar bisa diubah, kita akan menyediakan function untuk mengubah dan mendapatkan properties tersebut

Getter dan Setter

- Di PHP, proses encapsulation sudah dibuat standarisasinya, dimana kita bisa menggunakan Getter dan Setter method.
- Getter adalah function yang dibuat untuk mengambil data field
- Setter ada function untuk mengubah data field

Tipe Data	Getter Method	Setter Method
boolean	isXxx(): bool	setXxx(bool value)
lainnya	getXxx(): tipeData	setXxx(tipeData value)

INTERFACE

Interface

- Sebelumnya kita sudah tahu bahwa abstract class bisa kita gunakan sebagai kontrak untuk class child nya.

- Namun sebenarnya yang lebih tepat untuk kontrak adalah Interface
- Jangan salah sangka bahwa Interface disini bukanlah User Interface
- Interface mirip seperti abstract class, yang membedakan adalah di Interface, semua method otomatis abstract, tidak memiliki block
- Di interface kita tidak boleh memiliki properties, kita hanya boleh memiliki constant
- Untuk mewariskan interface, kita tidak menggunakan kata kunci extends, melainkan implements
- Dan berbeda dengan class, kita bisa implements lebih dari satu interface

FINAL CLASS

Final Class

- Kata kunci final bisa digunakan di class, dimana jika kita menggunakan kata kunci final sebelum class, maka kita menandakan bahwa class tersebut tidak bisa diwariskan lagi
- Secara otomatis semua class child nya akan error

Final Function

- Kata kunci final juga bisa digunakan di function
- Jika sebuah function kita tambahkan kata kunci final, maka artinya function tersebut tidak bisa di override lagi di class child nya
- Ini sangat cocok jika kita ingin mengunci implementasi dari sebuah method agar tidak bisa diubah lagi oleh class child nya

ANONYMOUS CLASS

Anonymous Class

- Anonymous class atau class tanpa nama.
- Adalah kemampuan mendeklarasikan class, sekaligus meng-instansiasi object-nya secara langsung
- Anonymous class sangat cocok ketika kita berhadapan dengan kasus membuat implementasi interface atau abstract class sederhana, tanpa harus membuat implementasi class nya

Constructor di Anonymous Class

- Anonymous class juga mendukung constructor
- Jadi kita bisa menambahkan constructor jika kita mau

STATIC KEYWORD

static Keyword

- Kata kunci static adalah keyword yang bisa kita gunakan untuk membuat properties atau function di class bisa diakses secara langsung tanpa menginstansiasi class terlebih dahulu
- Namun ingat, saat kita buat static properties atau function, secara otomatis hal itu tidak akan berhubungan lagi dengan class instance yang kita buat
- Untuk cara mengakses static properties dan function sama seperti mengakses constant, kita bisa menggunakan operator ::
- static function tidak bisa mengakses function biasa, karena function biasa menempel pada class instance sedangkan static function tidak

MAGIC FUNCTION

https://www.php.net/manual/en/language.oop5.magic.php

Magic Function

- Magic function adalah function-function yang sudah ditentukan kegunaannya di PHP
- Kita tidak bisa membuat function tersebut, kecuali memang sudah ditentukan kegunaannya
- Sebelumnya kita sudah membahas beberapa magic function, seperti __construct() sebagai constructor, __destruct() sebagai destructor, dan __clone() sebagai object cloning
- Masih banyak magic function lainnya, kita bisa melihatnya di link berikut : https://www.php.net/manual/en/language.oop5.magic.php

__toString() Function

- __toString() function merupakan salah satu magic function yang digunakan sebagai representasi string sebuah object
- Jika misal kita ingin membuat string dari object kita, kita bisa membuat function __toString()

__invoke() Function

- __invoke() merupakan function yang dieksekusi ketika object yang kita buat dianggap sebagai function
- Misal ketika kita membuat object $student, lalu kita melakukan $student(), maka secara otomatis function __invoke() yang akan dieksekusi

__debugInfo() Function

- Sebelumnya kita sering melakukan debug variable menggunakan function var_dump()
- Function var_dump() sebenarnya memanggil function __debugInfo()
- Jika kita ingin mengubah isi dari debug info, kita bisa membuat function __debugInfo()

EXCEPTION

Exception

- Saat kita membuat aplikasi, kita tidak akan terhindar dengan yang namanya error
- Di PHP, error direpresentasikan dengan istilah exception, dan semua direpresentasikan dalam bentuk class exception
- Kita bisa menggunakan class exception sendiri, atau menggunakan yang sudah disediakan oleh PHP
- Jika kita ingin membuat exception, maka kita harus membuat class yang implement interface Throwable atau turunan-turunannya

Membuat Exception

- Exception biasanya terjadi di function
- Di dalam kode program kita, untuk membuat exception kita cukup menggunakan kata kunci throw, diikuti dengan object exception nya

Try Catch

- Saat kita memanggil sebuah function yang bisa menyebabkan exception, maka kita disarankan menggunakan try-catch expression di PHP
- Ini gunanya agar kita bisa menangkap exception yang terjadi, karena jika tidak ditangkap, lalu terjadi exception, maka secara otomatis program kita akan berhenti
- Cara menggunakan try-catch expression di PHP sangat mudah, di block try, kita tinggal panggil method yang bisa menyebabkan exception, dan di block catch, kita bisa melakukan sesuatu jika terjadi exception

Finally Keyword

- Dalam try-catch, kita bisa menambahkan block finally
- Block finally ini adalah block dimana akan selalu dieksekusi baik terjadi exception ataupun tidak
- Ini sangat cocok ketika kita ingin melakukan sesuatu, tidak peduli sukses ataupun gagal, misal di block try kita ingin membaca file, di block catch kita

akan tangkap jika terjadi error, dan di block finally error ataupun sukses membaca file, kita wajib menutup koneksi ke file tersebut, biar tidak menggantung di memory

Debug Exception

- Exception di PHP memiliki sebuah function bernama getTrace()
- Function getTrace() berisikan informasi dari exception yang terjadi, seperti lokasi file, baris ke berapa, function mana sampai argumenty yang dikirim di function tersebut apa
- Ini sangat cocok untuk kita jika ingin mendebug ketika terjadi exception