



Jurusan Teknologi Informasi Politeknik Negeri Malang

## **Jobsheet-12: PHP – OOP**

### **Mata Kuliah Desain dan Pemrograman Web**

Pengampu: Tim Ajar Desain dan Pemrograman Web

*Oktober 2023*

---

**Nama** : Lentina Abrila Putri  
**Nim** : 2241760120  
**Prodi** : SIB 2C

#### **Topik**

- Konsep Pembuatan Web Dinamis dengan OOP

#### **Tujuan**

Mahasiswa diharapkan dapat:

1. Mahasiswa mampu membuat class dan object, inheritance, polymorphism, encapsulation, abstraction, interfaces, constructors and destructors, dan encapsulation and access modifier
2. Mahasiswa mampu membuat CRUD dengan OOP

#### **Perhatian**

Jobsheet ini harus dikerjakan step-by-step sesuai langkah-langkah praktikum yang sudah diberikan. Soal dapat dijawab langsung di dalam kolom yang disediakan dengan menggunakan PDF Editor.

#### **Pendahuluan**

##### **OOP**

Pemrograman Berorientasi Objek (OOP) adalah paradigma pemrograman yang sangat penting dalam dunia pengembangan perangkat lunak. Ini memungkinkan para pengembang untuk mengorganisasi kode mereka menjadi objek-objek yang memiliki atribut (data) dan metode (fungsi) yang terkait.

##### **Pengenalan Pemrograman Berorientasi Objek (OOP)**

Pemrograman Berorientasi Objek didasarkan pada konsep objek, yang mewakili entitas dalam dunia nyata. Setiap objek memiliki karakteristik yang disebut atribut (properti), dan dapat melakukan tindakan tertentu yang disebut metode (fungsi). OOP membantu dalam memecah kode menjadi bagian-bagian yang lebih kecil dan lebih mudah dikelola.

## **Kenapa OOP Penting?**

Dalam dunia pengembangan website yang semakin kompleks dan dinamis, penggunaan Konsep Pemrograman Berorientasi Objek (OOP) telah menjadi landasan yang esensial. OOP membawa keefektifan, kemudahan pemeliharaan, dan skalabilitas yang tak ternilai harganya untuk proyek-proyek website. Artikel ini akan membahas mengapa OOP begitu penting dalam pengembangan proyek website dan manfaat utamanya.

## **Modularitas dan Pengelolaan Kode yang Lebih Baik**

Salah satu manfaat utama OOP adalah kemampuannya untuk memecah kode menjadi modul atau objek yang independen. Dalam pengembangan website, setiap komponen seperti formulir, tampilan, database, dan lainnya dapat diwakili sebagai objek yang terpisah. Ini memungkinkan tim pengembangan untuk bekerja secara

terpisah pada komponen-komponen ini, mempercepat proses pengembangan dan memungkinkan pemeliharaan yang lebih mudah di masa depan.

## Penggunaan Ulang (Reusability) dan Efisiensi

Dalam OOP, objek-objek dapat digunakan ulang di berbagai bagian proyek. Ini mengurangi jumlah kode yang perlu ditulis, menghemat waktu dan usaha pengembang. Misalnya, jika Anda telah membuat objek "Formulir" yang memiliki metode untuk memvalidasi input, Anda dapat menggunakannya di berbagai halaman website tanpa perlu menulis ulang kode validasi tersebut.

## Pengelolaan Kesalahan yang Lebih Baik

Ketika terjadi kesalahan dalam kode OOP, Anda dapat dengan mudah mengisolasi dan menemukan sumber kesalahan tersebut karena setiap objek memiliki tanggung jawab yang jelas. Ini memungkinkan Anda untuk memperbaiki masalah lebih cepat dan lebih akurat, mengurangi waktu yang dihabiskan untuk debugging.

## Skalabilitas dan Pengembangan Kolaboratif

Proyek website cenderung berkembang seiring waktu. Dengan OOP, Anda dapat dengan mudah menambahkan fitur baru atau memperbarui komponen yang ada tanpa mengganggu fungsi lainnya. Tim pengembangan juga dapat bekerja secara paralel pada berbagai komponen, karena setiap objek berdiri sendiri dan tidak terlalu bergantung pada yang lain.

## Pemeliharaan Lebih Mudah

Ketika proyek website tumbuh, pemeliharaan menjadi sangat penting. OOP membantu dalam memisahkan perubahan yang diperlukan pada suatu komponen tanpa mempengaruhi yang lain. Jika Anda ingin mengubah tampilan halaman tertentu, Anda hanya perlu mengedit objek tampilan tanpa perlu khawatir tentang dampaknya pada komponen lain.

## Enkapsulasi dan Keamanan

Konsep enkapsulasi dalam OOP memungkinkan Anda untuk menyembunyikan detail implementasi dari komponen lainnya. Ini berarti bahwa komponen lain hanya dapat berinteraksi dengan objek melalui antarmuka yang ditentukan, mengurangi potensi kesalahan atau manipulasi yang tidak diinginkan.

## Fleksibilitas dan Peningkatan Kualitas Kode

OOP memungkinkan Anda untuk membuat abstraksi yang tinggi untuk mengelola kerumitan dan mendefinisikan pola umum. Ini meningkatkan kualitas kode karena mengikuti prinsip-prinsip yang terbukti dalam desain perangkat lunak, seperti DRY (Don't Repeat Yourself) dan SOLID (Prinsip-responsibilitas terpisah, Terbuka-Tertutup, Substitusi Liskov, Segregasi Antarmuka, Ketergantungan Inversi).

## Konsep Utama dalam OOP PHP

Dalam PHP, OOP memungkinkan Anda untuk mengorganisir dan mengelompokkan kode menjadi unit-unit yang lebih terstruktur dan mudah dikelola. Berikut adalah konsep-konsep utama OOP dalam PHP:

### Praktikum 1. Basic OOP

Langkah	Keterangan
1	Kelas adalah blueprint atau cetak biru yang mendefinisikan struktur dan perilaku suatu objek. Kelas berisi atribut (data) dan metode (fungsi) yang berkaitan dengan objek tersebut. Objek, di sisi lain, adalah instance konkret dari suatu kelas, memiliki nilai nyata untuk atribut dan mampu menjalankan metode yang didefinisikan dalam kelas. Dalam PHP, Anda dapat membuat kelas dengan kata kunci <code>class</code> dan kemudian membuat objek dari kelas tersebut dengan kata kunci <code>new</code> . Berikut adalah contoh sederhana:
2	Buatlah folder <code>oop</code> dalam folder <code>dasarWeb/</code> dengan file baru yaitu <code>oop.php</code> .

3	Ketikkan ke dalam file oop.php tersebut kode di bawah ini.
4	<pre> &lt;?php class Car {     public \$brand;      public function startEngine()     {         echo "Engine started!";     } }  \$car1 = new Car(); \$car1-&gt;brand = "Toyota";  \$car2 = new Car(); \$car2-&gt;brand = "Honda";  \$car1-&gt;startEngine();  echo \$car2-&gt;brand; </pre>
5	<p>Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.1)</p> <p><b>Hasil:</b></p> <pre>Engine started!Honda</pre> <p><b>Penjelasan:</b></p> <p>Dari hasil diatas ini bahwa php juga dapat dilakukan oleh oop atau objek oriented programming dimana memaksimalkan objek untuk meminimalisirkan baris pada kodeprogram. Potongan kode PHP tersebut merupakan pembuatan kelas "Car" dengan variabel publik "brand" dan metode publik "startEngine()". Selanjutnya, dua objek dibuat dari kelas tersebut, yaitu \$car1 dengan merek "Toyota" dan \$car2 dengan merek "Honda". Metode startEngine() kemudian dipanggil pada objek \$car1 untuk menampilkan pesan "Engine started;", dan merek dari \$car2 ("Honda") ditampilkan menggunakan echo. potongan kode tersebut menciptakan objek mobil dengan merek berbeda dan memanggil metode untuk memulai mesin serta menampilkan merek salah satu mobil.</p>
6	<p>Inheritance adalah salah satu konsep dasar dalam pemrograman berorientasi objek (OOP) yang memungkinkan sebuah class untuk mewarisi properti dan metode dari class lain. Class yang mewarisi disebut subclass atau child class, sedangkan class yang memberikan warisan disebut superclass atau parent class. Konsep ini memungkinkan kita untuk menggunakan kembali kode, memperpanjang fungsionalitas, dan membangun hierarki class.</p> <p>Berikut ini adalah contoh sederhana penerapan inheritance dalam PHP:</p>

```

class Animal
{
    protected $name;

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

    public function eat()
    {
        echo $this->name . " is eating.<br>";
    }

    public function sleep()
    {
        echo $this->name . " is sleeping.<br>";
    }
}

class Cat extends Animal
{
    public function meow()
    {
        echo $this->name . " says meow!<br>";
    }
}

class Dog extends Animal
{
    public function bark()
    {
        echo $this->name . " says woof!<br>";
    }
}

$cat = new Cat("Whiskers");
$dog = new Dog("Buddy");

$cat->eat();
$dog->sleep();

$cat->meow();
$dog->bark();

```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.2)

**Hasil:**

```
Whiskers is eating.  
Buddy is sleeping.  
Whiskers saya meow!  
Buddy saya woof!
```

**Penjelasan:**

Kode diatas mendefinisikan tiga class yaitu 'animal' 'cat' dan 'dog'. Class animal memiliki variable protected 'name' beserta konstruktornya juga method 'eat()' dan method 'sleep()'. Class cat memiliki function method 'meow()' dan class 'dog()' memiliki function/method 'bark()' - objek cat dan dog yang dimana cat dapat mengakses method eat dan meow sedangkan objek dog dapat mengakses method sleep dan bark. Dapat disimpulkan bahwa class Animal sebagai class induk yang dapat mewariskan function/method eat kepada kelas cat dan function/method sleep diwariskan kepada class dog . Karena itu kedua class antara dog dan cat memiliki kata extends. Karena ada pewarisan di antara kelas kelas ini.

6

Polymorphism adalah konsep dalam pemrograman berorientasi objek yang memungkinkan objek dari class yang berbeda untuk merespon pada pemanggilan metode dengan cara yang sama. Ini dapat diwujudkan dalam PHP melalui penggunaan antarmuka (interface) dan penggunaan overriding metode. Dengan polymorphism, Anda dapat memperlakukan objek dari class yang berbeda dengan cara yang seragam.

	Berikut adalah contoh sederhana penggunaan polymorphism dalam PHP menggunakan antarmuka:
--	------------------------------------------------------------------------------------------

```

interface Shape
{
    public function calculateArea();
}

class Circle implements Shape
{
    private $radius;

    public function __construct($radius)
    {
        $this->radius = $radius;
    }

    public function calculateArea()
    {
        return pi() * pow($this->radius, 2);
    }
}

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

$circle = new Circle(5);
$rectangle = new Rectangle(4, 6);

printArea($circle);
printArea($rectangle);

```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.3)

**Hasil:**



Area: 78.539816339745

Area: 24

**Penjelasan:**

Code di atas menggunakan PHP untuk membuat dua kelas, yaitu Circle dan Rectangle, yang memiliki kemampuan menghitung luas masing-masing bentuk. Kedua kelas ini mengikuti peraturan yang ditetapkan oleh antarmuka (interface) bernama Shape, yang memastikan bahwa setiap kelas harus memiliki metode calculateArea(). Ada juga fungsi printArea yang dapat mencetak luas dari berbagai bentuk, karena menerima objek bentuk apapun yang mengikuti interface Shape

Encapsulation adalah salah satu konsep dalam pemrograman berorientasi objek (OOP) yang mengizinkan pembungkusan (encapsulation) properti dan metode dalam sebuah class sehingga akses ke mereka dapat dikontrol. Hal ini dapat membantu dalam menerapkan prinsip-prinsip pengelolaan akses dan memastikan bahwa properti dan metode yang mungkin berubah di kemudian hari tidak merusak integritas class atau program secara keseluruhan.

Berikut adalah contoh sederhana encapsulation dalam PHP:

```
class Car
{
    private $model;
    private $color;

    public function __construct($model, $color)
    {
        $this->model = $model;
        $this->color = $color;
    }

    public function getModel()
    {
        return $this->model;
    }

    public function setColor($color)
    {
        $this->color = $color;
    }

    public function getColor()
    {
        return $this->color;
    }
}

$car = new Car("Toyota", "Blue");

echo "Model: " . $car->getModel() . "<br>";
echo "Color: " . $car->getColor() . "<br>";

$car->setColor("Red");

echo "Updated Color: " . $car->getColor() . "<br>";
```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.4)

**Hasil:**

Model: Toyota  
Color: Blue  
Updated Color: Blue

**Penjelasan:**

	<p>Class Car memiliki properti berupa \$model dan \$color yang bermodifier private. Memiliki konstruktor berparameter serta 3 method, antara lain</p> <ul style="list-style-type: none"> <li>• Method getModel() yang digunakan untuk mendapat nilai dari \$model yang telah diset properti.</li> <li>• Method setColor() digunakan untuk mengeset/mengubah nilai \$color.</li> <li>• Method getColor() digunakan untuk mendapat nilai \$color Kemudian pembuatan objek \$car yang memiliki properti model = "Toyota" dan color = "Blue". Dengan penggunaan method setColor("Red") dalam artian menyetting/mengubah warna \$car menjadi Red. Semua properti ini kemudian di cetak dengan penggunaan methpd getter. Dengan demikian, kode tersebut menggambarkan enkapsulasi dengan menggunakan properti private dan menyediakan metode publik untuk mengakses dan mengubah nilai properti tersebut, menjaga kontrol terhadap data internal kelas Car.</li> </ul>
8	<p>Abstraction adalah salah satu konsep dasar dalam pemrograman berorientasi objek (OOP) yang memungkinkan Anda menyembunyikan detail internal dan hanya mengekspos fungsionalitas yang diperlukan. Ini membantu dalam menciptakan class dan metode yang bersifat umum dan</p>

fleksibel, memungkinkan pengguna untuk berinteraksi dengan objek tanpa perlu mengetahui implementasi internalnya.

Berikut adalah contoh sederhana abstraksi dalam PHP menggunakan abstract class dan method:

```
abstract class Shape
{
    abstract public function calculateArea();
}

class Circle extends Shape
{
    private $radius;

    public function __construct($radius)
    {
        $this->radius = $radius;
    }

    public function calculateArea()
    {
        return pi() * pow($this->radius, 2);
    }
}

class Rectangle extends Shape
{
    private $width;
    private $height;

    public function __construct($width, $height)
    {
        $this->width = $width;
        $this->height = $height;
    }

    public function calculateArea()
    {
        return $this->width * $this->height;
    }
}

$circle = new Circle(5);
$rectangle = new Rectangle(4, 6);

echo "Area of Circle: " . $circle->calculateArea() . "<br>";
echo "Area of Rectangle: " . $rectangle->calculateArea() . "<br>";
```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.5)

**Hasil:**

```
Area of Circle : 78.539816339745
Area of Rectangle : 24
```

**Penjelasan:**

Kode diatas mengilustrasikan konsep abstraksi dalam OOP:

- Kelas abstrak `Shape` menyediakan kerangka dasar dengan metode abstrak `calculateArea()`. Menggambarkan ide bahwa ada suatu entitas yang merupakan bentuk (shape) dan memiliki kemampuan untuk menghitung luasnya, tetapi implementasi

	<p>sebenarnya dari perhitungan ini didelegasikan kepada kelas-kelas turunannya.</p> <ul style="list-style-type: none"> <li>• Kelas <code>`Circle`</code> dan <code>`Rectangle`</code> masing-masing mewarisi dan mengimplementasikan metode <code>`calculateArea()`</code> sesuai dengan jenis bentuknya.</li> <li>• Objek <code>`\$circle`</code> dan <code>`\$rectangle`</code> dibuat, dan metode <code>`calculateArea()`</code> dipanggil untuk menghitung luas lingkaran dan persegi panjang.</li> <li>• Output mencetak hasil perhitungan luas keduanya. Jadi, kode tersebut menggambarkan konsep abstraksi, di mana kelas abstrak Shape memberikan kerangka dasar dan kelas-kelas turunannya memberikan implementasi khusus sesuai dengan jenis bentuk yang diwakilinya.</li> </ul>
9	<p>Interface adalah konsep dalam pemrograman berorientasi objek yang memungkinkan definisi kontrak atau kerangka yang harus diikuti oleh class-class yang mengimplementasikannya. Interface tidak memiliki implementasi sendiri, tetapi hanya menyediakan deklarasi metode dan properti yang harus diimplementasikan oleh class yang menggunakannya. Hal ini memungkinkan untuk mencapai polimorfisme tanpa memerlukan pewarisan tunggal, sehingga sebuah class dapat mengimplementasikan beberapa interface.</p> <p>Berikut adalah contoh penggunaan interface dalam PHP:</p>

```

interface Shape
{
    public function calculateArea();
}

interface Color
{
    public function getColor();
}

class Circle implements Shape, Color
{
    private $radius;
    private $color;

    public function __construct($radius, $color)
    {
        $this->radius = $radius;
        $this->color = $color;
    }

    public function calculateArea()
    {
        return pi() * pow($this->radius, 2);
    }

    public function getColor()
    {
        return $this->color;
    }
}

$circle = new Circle(5, "Blue");

echo "Area of Circle: " . $circle->calculateArea() . "<br>";
echo "Color of Circle: " . $circle->getColor() . "<br>";

```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.6)

**Hasil:**

Area of Circle: 78.539816339745

Color of Circle: Blue

**Penjelasan:**

- Interface untuk membuat kerangka apa saja yang nantinya ingin digunakan lalu pada saat eksekusinya menggunakan 'implementasi' pada class yang ingin menggunakan kerangka yang sudah dibuat pada interface.
- Interface 'Shape' memiliki metode calculateArea() dan Interface 'Color' memiliki method getColor() yang harus diimplementasikan oleh kelas yang mengimplementasikan interface ini. Dalam class Circle dituliskan bahwa kelas ini mengimplementasikan interface Shape dan Color. Pada class ini memiliki properti private \$radius dan \$color, memiliki konstruktor berparameter juga mengimplementasikan method calculateArea() dan getColor() sebagai syarat dari kedua interfacenya. Dari class ini dibuat objek \$circle

	<p>dengan nilai \$radius = 5 dan \$color = blue. Metode calculateArea() dan getColor() dipanggil pada objek \$circle dan hasilnya dicetak. Jadi, interface digunakan untuk menggambarkan kontrak untuk perilaku khusus, dan class Circle memenuhi kontrak tersebut dengan mengimplementasikan kedua interface, sehingga objek dari class tersebut dapat digunakan sesuai dengan interface yang diharapkan.</p>
10	<p>Constructors dan destructors adalah metode khusus dalam pemrograman berorientasi objek (OOP) yang digunakan dalam PHP untuk menginisialisasi dan membersihkan objek. Constructor adalah metode yang dipanggil secara otomatis ketika objek baru dibuat, sedangkan destructor adalah metode yang dipanggil secara otomatis ketika objek dihapus atau tidak lagi digunakan.</p> <p>Constructor (Metode Pembuat)          Constructor menggunakan nama khusus __construct dalam PHP. Constructor ini akan dipanggil secara otomatis setiap kali objek baru dibuat dari class yang mengandung constructor tersebut.</p> <p>Destructor (Metode Penghancur)          Destructor menggunakan nama khusus __destruct dalam PHP. Destructor ini akan dipanggil secara otomatis ketika objek dihapus atau program selesai dieksekusi.</p>

Berikut adalah contoh constructor dan destructor:

```
class Car
{
    private $brand;

    public function __construct($brand)
    {
        echo "A new car is created.<br>";
        $this->brand = $brand;
    }

    public function getBrand()
    {
        return $this->brand;
    }

    public function __destruct()
    {
        echo "The car is destroyed.<br>";
    }
}

$car = new Car("Toyota");

echo "Brand: " . $car->getBrand() . "<br>";
```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.7)

**Hasil:**

```
A new car is created.
Brand: Toyota
The car is destroyed.
```

**Penjelasan:**

Kode diatas menciptakan class 'car' yang memiliki konstruktor untuk inisialisasi metode 'getBrand()' untuk mendapatkan merek dan metode khusus '\_\_destruct()' yang memberikan pesan saat objek dihancurkan. objek '\$car' dibuat dengan merek Toyota dan pesan pesan ditampilkan data pembuatan dan penghacuran objek.



11	<p><b>Encapsulation and Access Modifiers</b></p> <p>Encapsulation adalah salah satu konsep utama dalam pemrograman berorientasi objek (OOP), dan itu melibatkan pembungkusan data (variabel) dan metode (fungsi) dalam sebuah class. Ini membantu dalam menyembunyikan implementasi internal suatu class dan hanya mengekspos fungsionalitas yang diperlukan. Access modifiers adalah bagian dari encapsulation yang memungkinkan Anda mengontrol tingkat akses ke properti dan metode dalam sebuah class.</p> <p>PHP memiliki tiga access modifiers utama yang dapat digunakan dalam class:</p> <p><b>Public (public):</b> Properti atau metode yang dideklarasikan sebagai public dapat diakses dari luar class, sehingga mereka bersifat terbuka untuk diakses dari mana saja.</p> <p><b>Protected (protected):</b> Properti atau metode yang dideklarasikan sebagai protected hanya dapat diakses dari dalam class itu sendiri dan dari class turunannya (inheritance).</p> <p><b>Private (private):</b> Properti atau metode yang dideklarasikan sebagai private hanya dapat diakses dari dalam class itu sendiri. Mereka tidak dapat diakses dari luar class, bahkan oleh class turunannya.</p> <p>Berikut adalah contoh penggunaan access modifiers dalam PHP:</p>
----	---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

```

class Animal
{
    public $name;
    protected $age;
    private $color;

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

    public function getName()
    {
        return $this->name;
    }

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

    private function getColor()
    {
        return $this->color;
    }
}

$animal = new Animal("Dog", 3, "Brown");

echo "Name: " . $animal->name . "<br>";
echo "Age: " . $animal->getAge() . "<br>";
echo "Color: " . $animal->getColor() . "<br>";

```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.8)

**Hasil:**

```

Name: Dog
Age: 3
Color: Brown

```

**Penjelasan:**

- Public (\$name) : property name memiliki akses modifier public, sehingga dapat diakses dari luar kelas dan dari mana saja didalam kode.
- Protected (\$age) akses modifier protected yg hanya dapat diakses dari dalam kelas Animal dan kelas turunannya.
- Private (\$color) akses modifier yang hanya dapat diakses dari dalam kelas Animal .
- Method getter : lain getName(), getAge(), dan getColor() yang berfungsi untuk mengakses/memanggil variabel/properti dengan modifier private.
- Mengcreate objek Animal dengan parameter name=Dog, Age=3 dan Color=Brown. Karena variabel name bermodifler public untuk pemanggilannya cukup memanggil nama variabelnya sedangkan age dan private bermodifler protected dan private sehingga yang bisa mengakses langsung hanya class itu sendiri maka diperlukan method getAge dan getColor.

## Praktikum 2. CRUD dengan OOP

Langkah	Keterangan
1	Buat file baru pada folder oop dengan nama baru bernama database.php. Ketikkan kode seperti di bawah ini.
2	<pre>&lt;?php class Database {     private \$host = "localhost";     private \$username = "root";     private \$password = "";     private \$database = "prakwebdb";     public \$conn;      public function __construct()     {         \$this-&gt;conn = new mysqli(\$this-&gt;host, \$this-&gt;username, \$this-&gt;password, \$this-&gt;database);          if (\$this-&gt;conn-&gt;connect_error) {             die("Connection failed: " . \$this-&gt;conn-&gt;connect_error);         }     } }</pre>
3	Buat file baru pada folder oop dengan nama baru bernama crud.php. Ketikkan kode seperti di bawah ini.

4

```
<?php
require_once 'Database.php';

class Crud
{
    private $db;

    public function __construct()
    {
        $this->db = new Database();
    }

    // Create
    public function create($jabatan, $keterangan)
    {
        $query = "INSERT INTO jabatan (jabatan, keterangan) VALUES ($jabatan, '$keterangan')";
        $result = $this->db->conn->query($query);

        return $result;
    }

    // Read
    public function read()
    {
        $query = "SELECT * FROM jabata ";
        $result = $this->db->conn->query($query);

        $data = [];
        if ($result->num_rows > 0) {
            while ($row = $result->fetch_assoc()) {
                $data[] = $row;
            }
        }

        return $data;
    }

    // Read By Id
    public function readById($id)
    {
        $query = "SELECT * FROM jabatan WHERE i =$id";
        $result = $this->db->conn->query($query);
        if ($result->num_rows == 1) {
            return $result->fetch_assoc();
        } else {
            return null;
        }
    }

    // Update
    public function update($id, $jabatan, $keterangan)
    {
        $query = "UPDATE jabatan SET jabata ='$jabatan', keterangan='$keterangan' WHERE i =$id";
        $result = $this->db->conn->query($query);

        return $result;
    }

    // Delete
    public function delete($id)
    {
        $query = "DELETE FROM jabatan WHERE i =$id";
        $result = $this->db->conn->query($query);

        return $result;
    }
}
```

5

Buat file baru pada folder oop dengan nama baru bernama index.php. Ketikkan kode seperti di bawah ini.

```

<?php
require_once 'Crud.php';

$crud = new Crud();

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $jabatan = $_POST['jabatan'];
    $keterangan = $_POST['keterangan'];
    $crud->create($jabatan, $keterangan);
}

if (isset($_GET['action']) && $_GET['action'] === 'delete') {
    $id = $_GET['id'];
    $crud->delete($id);
}

$stampil = $crud->read();
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>CRUD Jabatan</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<body>
    <div class="container mt-5">
        <button type="button" class="btn btn-success mb-3" data-toggle="modal" data-target="#tambahModal">Tambah</button>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Jabatan</th>
                    <th>Keterangan</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>
                <?php
                foreach ($stampil as $show) {
                    echo "<tr>";
                    echo "<td>" . $show['id'] . "</td>";
                    echo "<td>" . $show['jabatan'] . "</td>";
                    echo "<td>" . $show['keterangan'] . "</td>";
                    echo "<td>";
                    echo "<a href='edit.php?id=" . $show['id'] . "' class='btn btn-primary btn-sm'>Edit</a> ";
                    echo "<a href='index.php?action=delete&id=" . $show['id'] . "' class='btn btn-danger btn-sm'>Delete</a>";
                    echo "</td>";
                    echo "</tr>";
                }
                ?>
            </tbody>
        </table>
        <div class="modal fade" id="tambahModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
            <div class="modal-dialog" role="document">
                <div class="modal-content">
                    <div class="modal-header">
                        <h5 class="modal-title" id="exampleModalLabel">Tambah Data Jabatan</h5>
                        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
                            <span aria-hidden="true">&times;</span>
                        </button>
                    </div>
                    <div class="modal-body">
                        <form method="post" action="">
                            <div class="form-group">
                                <label for="name">Jabatan:</label>
                                <input type="text" class="form-control" id="jabatan" name="jabatan" required>
                            </div>
                            <div class="form-group">
                                <label for="email">Keterangan:</label>
                                <textarea name="keterangan" class="form-control" id="keterangan" cols="30" rows="10" required></textarea>
                            </div>
                            <button type="submit" class="btn btn-primary">Tambah</button>
                        </form>
                    </div>
                </div>
            </div>
        </div>

        <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
        <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
    </body>

</html>

```

7

Buat file baru pada folder oop dengan nama baru bernama edit.php. Ketikkan kode seperti di bawah ini.

8

```
<?php
require_once 'Crud.php';

$crud = new Crud();

$id = $_GET['id'];

$stampil = $crud->readById($id);

if ($_SERVER['REQUEST_METHOD'] === 'POST') {
    $jabatan = $_POST['jabatan'];
    $keterangan = $_POST['keterangan'];

    $crud->update($id, $jabatan, $keterangan);

    header("Location: index.php");
    exit();
}
?>

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Jabatan</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<body>
    <div class="container mt-5">
        <h2>Edit Jabatan</h2>
        <form method="post" action="">
            <div class="form-group">
                <label for="jabatan">Jabatan:</label>
                <input type="text" class="form-control" id="jabatan" name="jabatan" value="<?php echo $stampil['jabatan']; ?>"
required>
            </div>
            <div class="form-group">
                <label for="keterangan">Keterangan:</label>
                <textarea name="keterangan" class="form-control" id="keterangan" cols="30" rows="10" required><?php echo $stampil[
'keterangan']; ?></textarea>
            </div>
            <input type="hidden" name="id" value="<?php echo $stampil['id']; ?>">
            <button type="submit" class="btn btn-primary">Update</button>
        </form>
    </div>

    <script src="https://code.jquery.com/jquery-3.3.1.slim.min.js"></script>
    <script src="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/js/bootstrap.min.js"></script>
</body>

</html>
```

Jalankan code pada praktikum 2. Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 2.1)

**Hasil:**

Tambah

ID	Jabatan	Keterangan	Aksi
----	---------	------------	------

Tambah Data Jabatan

Jabatan:

Manager

Keterangan:

Saya adalah Manager

Tambah

### Tampilan Data:

Tambah

ID	Jabatan	Keterangan	Aksi
1	Manager	Saya adalah Manager	<div>EditDelete</div>

### Aksi Pengeditan:

## Edit Jabatan

Jabatan:

Admin

Keterangan:

Saya adalah Admin

Update

### Hasil yang didapatkan:

Tambah

ID	Jabatan	Keterangan	Aksi
1	Admin	Saya adalah Admin	<div>EditDelete</div>

### Aksi penambahan jabatan:

Tambah

ID	Jabatan	Keterangan	Aksi
1	Admin	Saya adalah Admin	<a href="#">Edit</a> <a href="#">Delete</a>
2	HRD	Saya adalah HRD	<a href="#">Edit</a> <a href="#">Delete</a>

#### Aksi Delete:

Tambah

ID	Jabatan	Keterangan	Aksi
1	Admin	Saya adalah Admin	<a href="#">Edit</a> <a href="#">Delete</a>

#### Penjelasan:

Kode diatas ini merupakan program untuk aplikasi CRUD pada objek jabatan. Dengan memiliki beberapa class seperti:

- Database.php : ini digunakan untuk koneksi databasenya. Ada property private dan konstruktor kelas ini digunakan untuk membuat objek mysqli yang mewakili koneksi ke database. Jika koneksi gagal maka akan ada pesan error.
- Crud.php : ini operasi CRUD. Ada method seperti : create yang digunakan untuk membuat jabatan baru, read untuk membaca atau menampilkan seluruh jabatan yang ada di dalam database, readById digunakan untuk membaca data jabatan berdasarkan ID dari tabel 'jabatan' yang ada didalam database, update digunakan untuk memperbaharui data jabatan berdasarkan ID dengan melakukan operasi UPDATE pada table 'jabatan', delete untuk menghapus jabatan berdasarkan parameter yang didinputkan(id).
- Index.php : ini untuk tampilan dari data dan aksi yang dapat dioperasikan.
- Edit.php : ini untuk mengedit data jabatan Disini kita dapat menambah, membaca, memperbaharui, dan menghapus data jabatan dalam database dengan mudah melalui antarmuka web yang tersedia.