



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

DOSEN PENGAJAR : DIMAS WAHYU WIBOWO, S.T., M.T.

NAMA : ODDIS NUR ALIFATRHUR RAZAAQ
KELAS : SIB – 2C
NO. ABSEN : 20
NIM : 2241760015

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> <?php class Car { public \$brand; public function startEngine() { echo "Engine started!"; } } \$car1 = new Car(); \$car1->brand = "Toyota"; \$car2 = new Car(); \$car2->brand = "Honda"; \$car1->startEngine(); echo \$car2->brand; </pre>
5	<p>Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.1)</p> <p>Jawab:</p> <ul style="list-style-type: none"> Kode <div data-bbox="359 1077 1422 1615" data-label="Code-Block"> <pre> oop.php U X Pertemuan12 > oop.php > ... 1 <?php 2 class Car 3 { 4 public \$brand; 5 6 public function startEngine() 7 { 8 echo "Engine started!"; 9 } 10 } 11 \$car1 = new Car(); 12 \$car1->brand = "Toyota"; 13 14 \$car2 = new Car(); 15 \$car2->brand = "Honda"; 16 17 \$car1->startEngine(); 18 echo \$car2->brand; // Memindahkan tanda " " ke dalam baris yang sama dengan perintah echo 19 ?> </pre> </div> Hasil <div data-bbox="359 1675 1453 1832" data-label="Image"> </div> Penjelasan <p>Code di atas mendefinisikan sebuah kelas `Car` dengan properti `brand` dan metode `startEngine()` untuk memulai mesin. Dua objek `Car` dibuat, yaitu `\$car1` dengan merek "Toyota" dan `\$car2` dengan merek "Honda". Kemudian, metode `startEngine()` dipanggil pada objek `\$car1` untuk memulai mesin, dan merek dari objek `\$car2`</p>

	("Honda") dicetak dengan menggunakan perintah `echo`.--
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)

Jawab:

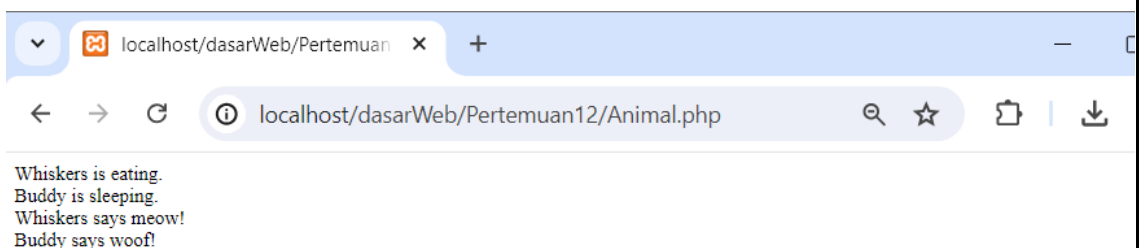
- Kode

```

Animal.php U X
Pertemuan12 > Animal.php > ...
1  <?php
2
3  class Animal
4  {
5      protected $name;
6
7      public function __construct($name)
8      {
9          $this->name = $name;
10     }
11
12     public function eat()
13     {
14         echo $this->name . " is eating.<br>";
15     }
16
17     public function sleep()
18     {
19         echo $this->name . " is sleeping.<br>";
20     }
21 }
22
23 class Cat extends Animal
24 {
25     public function meow()
26     {
27         echo $this->name . " says meow!<br>";
28     }
29 }
30
31 class Dog extends Animal
32 {
33     public function bark()
34     {
35         echo $this->name . " says woof!<br>";
36     }
37 }
38
39 $cat = new Cat("Whiskers");
40 $dog = new Dog("Buddy");
41
42 $cat->eat();
43 $dog->sleep();
44
45 $cat->meow();
46 $dog->bark();
47 ?>

```

- Hasil



Whiskers is eating.
Buddy is sleeping.
Whiskers says meow!
Buddy says woof!

- Penjelasan

Code di atas mendefinisikan tiga kelas dalam PHP: `Animal`, `Cat`, dan `Dog`. Kelas `Animal` memiliki properti proteksi `\$name` dan dua metode, `__construct()` untuk mengatur nama hewan saat objek dibuat, serta metode `eat()` dan `sleep()` untuk mencetak bahwa hewan sedang makan dan tidur, dengan nama hewan yang sesuai.

	Kelas `Cat` dan `Dog` masing-masing merupakan turunan dari `Animal`, menambahkan perilaku khusus seperti metode `meow()` untuk kucing dan `bark()` untuk anjing. Objek `Cat` dan `Dog` kemudian dibuat dan metode-metode dipanggil untuk menampilkan perilaku hewan-hewan tersebut dalam bentuk teks yang sesuai dengan masing-masing jenis hewan.
--	--

6	<p>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.</p> <p>Berikut adalah contoh sederhana penggunaan polymorphism dalam PHP menggunakan antarmuka:</p>
---	--


```

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)

Jawab:

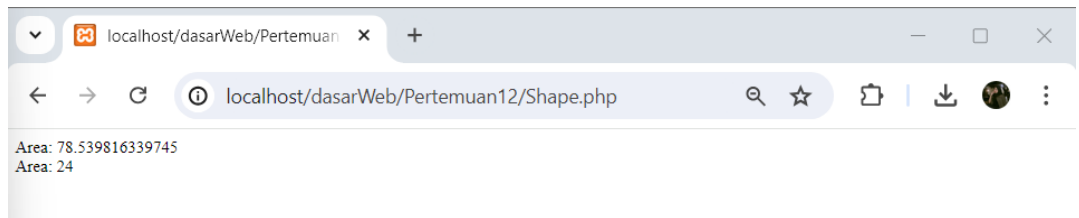
- Kode

```

Shape.php U X
Pertemuan12 > Shape.php > ...
1  <?php
2
3  interface Shape
4  {
5      public function calculateArea();
6  }
7
8  class Circle implements Shape
9  {
10     private $radius;
11
12     public function __construct($radius)
13     {
14         $this->radius = $radius;
15     }
16
17     public function calculateArea()
18     {
19         return pi() * pow($this->radius, 2);
20     }
21 }
22
23 class Rectangle implements Shape
24 {
25     private $width;
26     private $height;
27
28     public function __construct($width, $height)
29     {
30         $this->width = $width;
31         $this->height = $height;
32     }
33
34     public function calculateArea()
35     {
36         return $this->width * $this->height;
37     }
38 }
39
40 function printArea(Shape $shape)
41 {
42     echo "Area: " . $shape->calculateArea() . "<br>";
43 }
44
45 $circle = new Circle(5);
46 $rectangle = new Rectangle(4, 6);
47
48 printArea($circle);
49 printArea($rectangle);
50
51 ?>

```

- Hasil



- **Penjelasan**

Code di atas menggunakan konsep pemrograman berorientasi objek dalam PHP. Dalam code tersebut, terdapat sebuah interface `Shape` yang memiliki satu metode `calculateArea()` yang harus diimplementasikan oleh kelas-kelas yang menggunakannya. Kelas `Circle` dan `Rectangle` merupakan implementasi dari interface `Shape`, di mana kelas `Circle` menghitung luas lingkaran berdasarkan radius yang diberikan, sementara kelas `Rectangle` menghitung luas persegi panjang berdasarkan lebar dan tinggi yang diberikan. Fungsi `printArea()` menerima parameter bertipe `Shape` dan mencetak luas dari objek yang diberikan menggunakan metode `calculateArea()`. Objek `$circle` dan `$rectangle` dibuat sebagai instance dari masing-masing kelas, dan metode `printArea()` dipanggil untuk mencetak luas lingkaran dan persegi panjang tersebut. Dengan menggunakan interface dan pola desain ini, code menjadi lebih modular dan memungkinkan untuk melakukan polimorfisme dengan mengganti objek dengan implementasi yang berbeda dari 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)

Jawab:

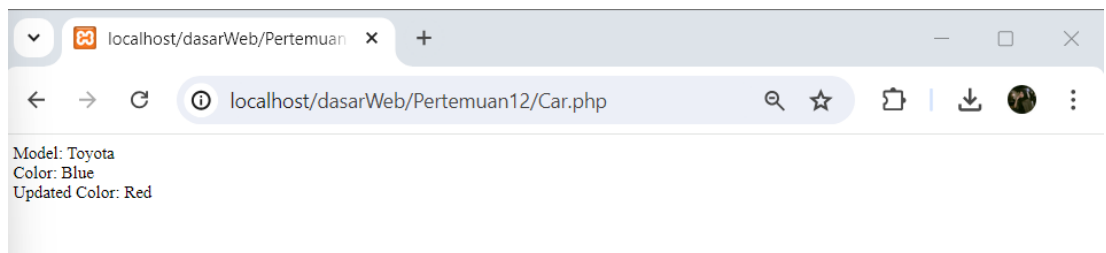
- Kode

```

Car.php U X
Pertemuan12 > Car.php > ...
1  <?php
2
3  class Car
4  {
5      private $model;
6      private $color;
7
8      public function __construct($model, $color)
9      {
10         $this->model = $model;
11         $this->color = $color;
12     }
13
14     public function getModel()
15     {
16         return $this->model;
17     }
18
19     public function setColor($color)
20     {
21         $this->color = $color;
22     }
23
24     public function getColor()
25     {
26         return $this->color;
27     }
28 }
29
30 $car = new Car("Toyota", "Blue");
31
32 echo "Model: " . $car->getModel() . "<br>";
33 echo "Color: " . $car->getColor() . "<br>";
34
35 $car->setColor("Red");
36
37 echo "Updated Color: " . $car->getColor() . "<br>";
38
39 ?>

```

- Hasil



- Penjelasan

Code di atas mendefinisikan sebuah kelas `Car` dalam PHP yang memiliki properti privat `\$model` dan `\$color`, serta metode-metode untuk mengakses dan mengubah nilai properti tersebut. Melalui konstruktor, objek `\$car` dibuat dengan model "Toyota" dan warna "Blue". Selanjutnya, nilai model dan warna ditampilkan menggunakan metode `getModel()` dan `getColor()`. Metode `setColor()` dipanggil untuk mengubah warna mobil menjadi "Red", dan warna yang diperbarui juga ditampilkan kembali menggunakan metode `getColor()`. Dengan cara ini, kode mengilustrasikan konsep dasar dari pemrograman berorientasi objek dalam PHP, termasuk pembuatan objek, pengaturan properti, dan penggunaan metode untuk mengakses dan memanipulasi data.

	objek tersebut.
8	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

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)

Jawab:

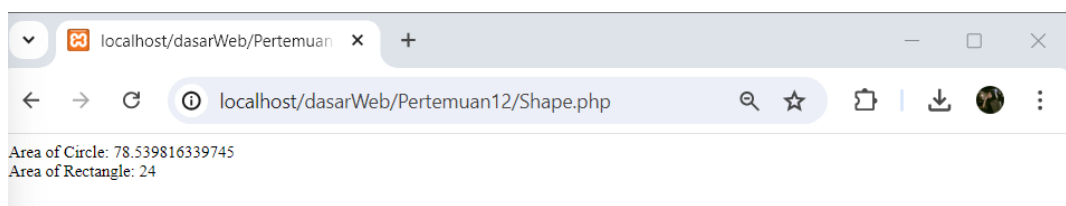
- Kode

```

Shape.php U X
Pertemuan12 > Shape.php > ...
1  <?php
2
3  abstract class Shape
4  {
5      abstract public function calculateArea();
6  }
7
8  class Circle extends Shape
9  {
10     private $radius;
11
12     public function __construct($radius)
13     {
14         $this->radius = $radius;
15     }
16
17     public function calculateArea()
18     {
19         return pi() * pow($this->radius, 2);
20     }
21 }
22
23 class Rectangle extends Shape
24 {
25     private $width;
26     private $height;
27
28     public function __construct($width, $height)
29     {
30         $this->width = $width;
31         $this->height = $height;
32     }
33
34     public function calculateArea()
35     {
36         return $this->width * $this->height;
37     }
38 }
39
40 $circle = new Circle(5);
41 $rectangle = new Rectangle(4, 6);
42
43 echo "Area of Circle: " . $circle->calculateArea() . "<br>";
44 echo "Area of Rectangle: " . $rectangle->calculateArea() . "<br>";
45
46 ?>

```

- Hasil



localhost/dasarWeb/Pertemuan12/Shape.php

Area of Circle: 78.539816339745
Area of Rectangle: 24

- Penjelasan

Code di atas mengilustrasikan penggunaan abstract class dan inheritance dalam PHP. Kelas `Shape` adalah abstract class yang mendefinisikan metode abstrak `calculateArea()` yang harus diimplementasikan oleh subclass-subclassnya. Kelas

	<p>`Circle` dan `Rectangle` masing-masing merupakan subclass dari `Shape`, dimana `Circle` menghitung luas lingkaran berdasarkan radius yang diberikan, sedangkan `Rectangle` menghitung luas persegi panjang berdasarkan lebar dan tinggi yang diberikan. Objek `\$circle` dan `\$rectangle` dibuat sebagai instance dari masing-masing kelas, dan metode `calculateArea()` dipanggil pada keduanya untuk menghitung dan mencetak luas lingkaran dan persegi panjang tersebut. Dengan menggunakan pola desain seperti ini, kode menjadi lebih terstruktur, modular, dan mendukung konsep polimorfisme dalam pemrograman berorientasi objek.</p>
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)

Jawab:

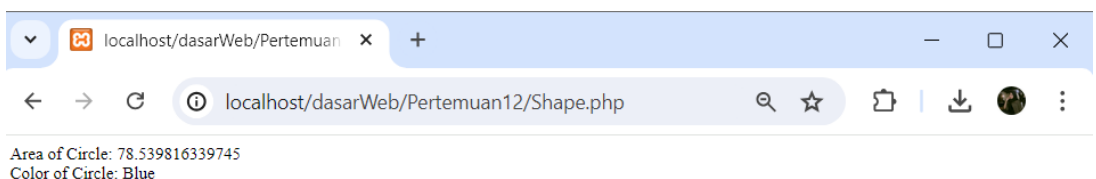
- Kode

```

Shape.php U X
Pertemuan12 > Shape.php > ...
1  <?php
2
3  interface Shape
4  {
5      public function calculateArea();
6  }
7
8  interface Color
9  {
10     public function getColor();
11 }
12
13 class Circle implements Shape, Color
14 {
15     private $radius;
16     private $color;
17
18     public function __construct($radius, $color)
19     {
20         $this->radius = $radius;
21         $this->color = $color;
22     }
23
24     public function calculateArea()
25     {
26         return pi() * pow($this->radius, 2);
27     }
28
29     public function getColor()
30     {
31         return $this->color;
32     }
33 }
34
35 $circle = new Circle(5, "Blue");
36
37 echo "Area of Circle: " . $circle->calculateArea() . "<br>";
38 echo "Color of Circle: " . $circle->getColor() . "<br>";
39
40 ?>

```

- Hasil



Area of Circle: 78.539816339745
Color of Circle: Blue

- Penjelasan

Code di atas mendemonstrasikan penggunaan interface dalam PHP untuk mendefinisikan kontrak perilaku yang harus diimplementasikan oleh kelas-kelas lain. Dalam contoh ini, terdapat dua interface: `Shape` yang memiliki metode `calculateArea()` untuk menghitung luas, dan `Color` yang memiliki metode `getColor()` untuk mengambil warna. Kelas `Circle` mengimplementasikan kedua interface tersebut dengan menghitung luas lingkaran berdasarkan radius yang diberikan dan mengembalikan warna lingkaran. Objek `\$circle` dibuat sebagai instance dari

	<p>`Circle` dengan radius 5 dan warna "Blue". Metode `calculateArea()` dipanggil untuk menghitung dan mencetak luas lingkaran, sedangkan `getColor()` dipanggil untuk mencetak warna lingkaran. Dengan menggunakan interface, kode menjadi lebih fleksibel karena memisahkan deklarasi perilaku dari implementasi spesifik kelas, memungkinkan untuk menyusun kelas-kelas dengan perilaku yang sama tetapi dengan implementasi yang berbeda.</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 <code>_construct</code> 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 <code>_destruct</code> 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)

Jawab:

- Kode

```

Car.php U X
Pertemuan12 > Car.php > ...
1  <?php
2
3  class Car
4  {
5      private $brand;
6
7      public function __construct($brand)
8      {
9          echo "A new car is created.<br>";
10         $this->brand = $brand;
11     }
12
13     public function getBrand()
14     {
15         return $this->brand;
16     }
17
18     public function __destruct()
19     {
20         echo "The car is destroyed.<br>";
21     }
22 }
23
24 $car = new Car("Toyota");
25
26 echo "Brand: " . $car->getBrand() . "<br>";
27
28 ?>

```

- Hasil

localhost/dasarWeb/Pertemuan12/Car.php

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

- Penjelasan

Code di atas mengilustrasikan penggunaan konstruktor (`__construct()`) dan destruktur (`__destruct()`) dalam PHP untuk kelas `Car`. Konstruktor digunakan untuk menginisialisasi objek `Car` dengan nilai brand yang diberikan saat objek dibuat, dan mencetak pesan bahwa mobil baru telah dibuat. Metode `getBrand()` digunakan untuk mengembalikan nilai brand mobil. Destruktor digunakan untuk mencetak pesan bahwa mobil telah dihancurkan saat objek `Car` dihapus dari memori, biasanya pada akhir eksekusi skrip PHP atau jika objek dihapus secara eksplisit dengan `unset()` atau `null`. Dengan demikian, kode ini menunjukkan konsep dasar dari penggunaan konstruktor dan destruktur dalam pengelolaan objek dalam PHP.

11	<p>Encapsulation and Access Modifiers</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>Public (public): Properti atau metode yang dideklarasikan sebagai public dapat diakses dari luar class, sehingga mereka bersifat terbuka untuk diakses dari mana saja.</p> <p>Protected (protected): Properti atau metode yang dideklarasikan sebagai protected hanya dapat diakses dari dalam class itu sendiri dan dari class turunannya (inheritance).</p> <p>Private (private): 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)

Jawab:

- Kode


```

Animal.php 2, U X
Pertemuan12 > Animal.php > ...
1  <?php
2
3  class Animal
4  {
5      public $name;
6      protected $age;
7      private $color;
8
9      public function __construct($name, $age, $color)
10     {
11         $this->name = $name;
12         $this->age = $age;
13         $this->color = $color;
14     }
15
16     public function getName()
17     {
18         return $this->name;
19     }
20
21     protected function getAge()
22     {
23         return $this->age;
24     }
25
26     private function getColor()
27     {
28         return $this->color;
29     }
30 }
31
32 $animal = new Animal("Dog", 3, "Brown");
33
34 echo "Name: " . $animal->name . "<br>";
35 echo "Age: " . $animal->getAge() . "<br>";
36 echo "Color: " . $animal->getColor() . "<br>";
37
38 ?>

```

- Penjelasan

Kode di atas mendefinisikan sebuah kelas `Animal` dengan tiga properti: `\$name` (publik), `\$age` (dilindungi), dan `\$color` (pribadi). Konstruktor `__construct()` digunakan untuk menginisialisasi nilai-nilai properti saat objek `Animal` dibuat. Terdapat metode `getName()` yang mengembalikan nilai properti `\$name` yang bersifat publik, metode `getAge()` yang mengembalikan nilai properti `\$age` yang bersifat dilindungi, dan metode `getColor()` yang mengembalikan nilai properti `\$color` yang bersifat pribadi. Dalam pemanggilan kode, properti `\$name` dapat diakses secara langsung karena bersifat publik, sedangkan metode `getAge()` dipanggil untuk mengambil nilai umur yang bersifat dilindungi. Namun, upaya untuk mengakses metode `getColor()` dari luar kelas akan menghasilkan error karena metode tersebut memiliki aksesibilitas yang pribadi.

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><?php class Database { private \$host = "localhost"; private \$username = "root"; private \$password = ""; private \$database = "prakwebdb"; public \$conn; public function __construct() { \$this->conn = new mysqli(\$this->host, \$this->username, \$this->password, \$this->database); if (\$this->conn->connect_error) { die("Connection failed: " . \$this->conn->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>

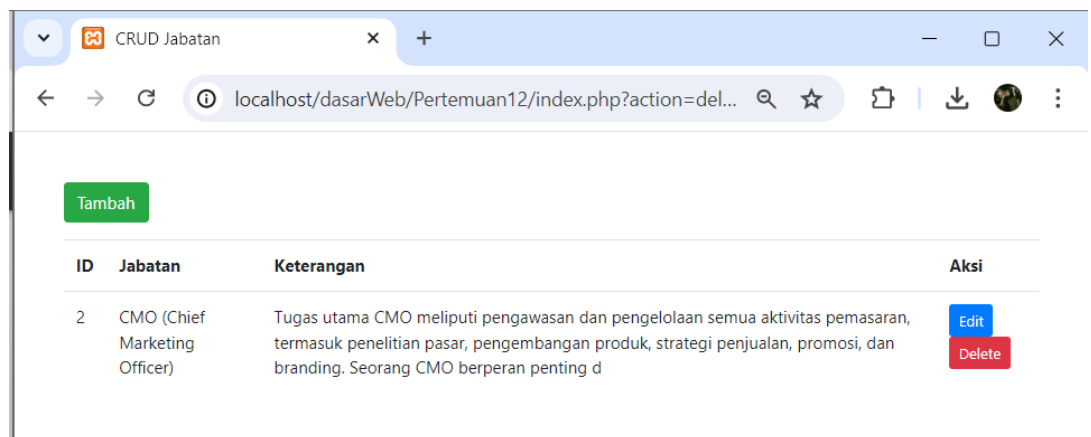
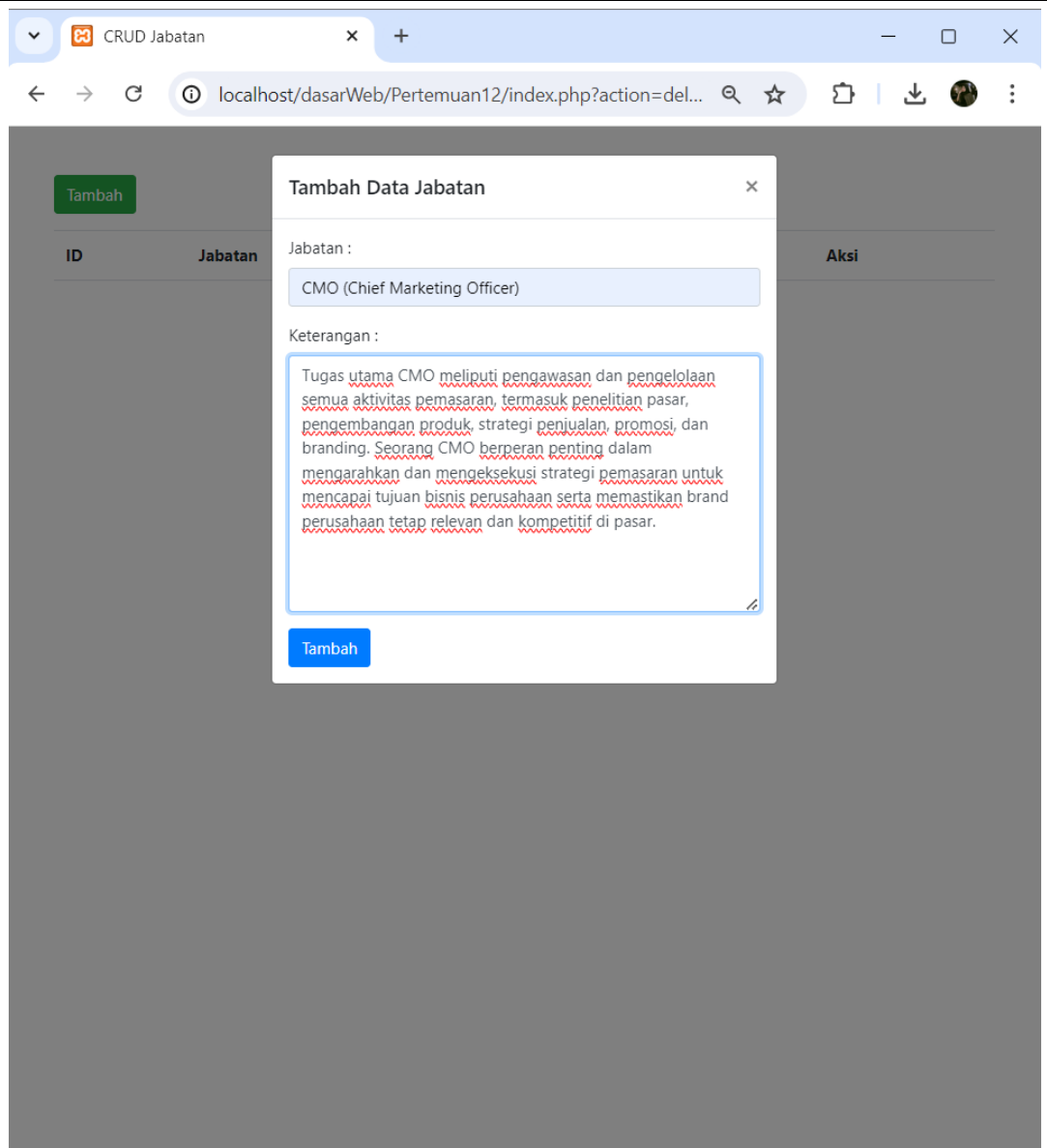
<body>
    <div class="container mt-5">
        <button type="button" class="btn btn-success mb-3" data-toggle="modal" data-target="#tambahModal">Tambah</button>
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Jabatan</th>
                    <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	<div data-bbox="284 271 1473 1366"> <pre> <?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> </pre> </div> <p>Jalankan code pada praktikum 2. Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 2.1)</p> <p>Jawab:</p> <ul style="list-style-type: none"> • Hasil



- **Penjelasan**
 Dalam kode tersebut, mengimplementasikan proyek yang mirip dengan proyek sebelumnya, tetapi menggunakan pendekatan OOP (Object-Oriented Programming). Koneksi database digunakan untuk terhubung ke basis data "praktikumdb", dan fitur "edit" digunakan untuk melakukan pengeditan data.