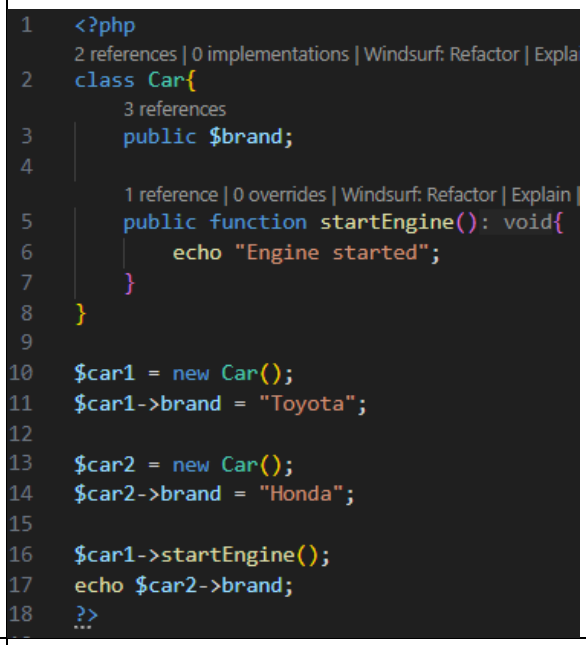
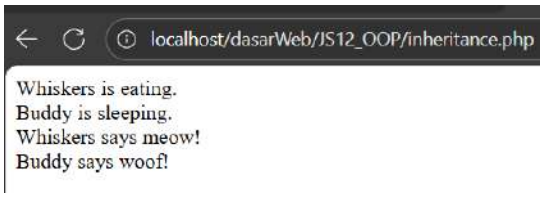
Nama : **Fiza Rahmatus Sholikha**
Kelas : **SIB 2E**
No. Absen : **07**

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 class dan kemudian membuat objek dari kelas tersebut dengan kata kunci new. Berikut adalah contoh sederhana:
2	Buatlah folder `dasarWeb/JS12_OOP` dan file buat baru didalamnya, dengan nama `oop.php`.
3	Ketikkan ke dalam file `oop.php` tersebut kode di bawah ini.
4	
5	Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.1) Pada kode ini terdapat class bernama Car dibuat sebagai blueprint dengan atribut public ($brand) dan satu method public (startEngine). Dua object ($car1 dan $car2) kemudian dibuat sebagai instance dari class Car menggunakan kata kunci new. Objek tersebut dapat memiliki nilai atributnya sendiri, dan method startEngine() dipanggil pada $car1.
6	`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. Berikut ini adalah contoh sederhana penerapan inheritance dalam PHP:

```php
<?php
2 references | 0 implementations | Windsurf: Refactor | Expla
class Car{
    3 references
    public $brand;

    1 reference | 0 overrides | Windsurf: Refactor | Explain |
    public function startEngine(): void{
        echo "Engine started";
    }
}

$car1 = new Car();
$car1->brand = "Toyota";

$car2 = new Car();
$car2->brand = "Honda";

$car1->startEngine();
echo $car2->brand;
?>
```

```php
<?php
2 references | 2 implementations | Windsurf: Refactor | Explain
class Animal{
    5 references
    protected $name;
    0 references | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function __construct($name){
        $this->name = $name;
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function eat(): void{
        echo $this->name . " is eating.<br>";
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function sleep(): void{
        echo $this->name . " is sleeping.<br>";
    }
}

1 reference | 0 implementations | Windsurf: Refactor | Explain
class Cat extends Animal{
    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function meow(): void{
        echo $this->name . " says meow!<br>";
    }
}

1 reference | 0 implementations | Windsurf: Refactor | Explain
class Dog extends Animal{
    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function bark(): void{
        echo $this->name . " says woof!<br>";
    }
}

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

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

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



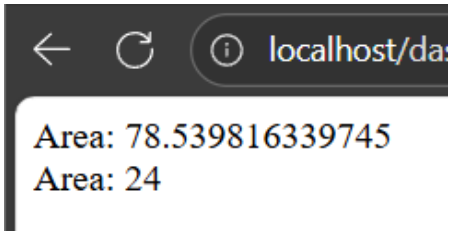
Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.2)
Pada kode ini terdapat class Animal yang merupakan parent class dengan atribut protected ($name) dan method (__construct, eat, sleep). Class Cat dan Dog adalah child class (anak) yang extends (mewarisi) semua properti dan method dari Animal. Hasilnya, objek $cat dan $dog bisa menggunakan method dari Animal (seperti eat() dan sleep()) sekaligus method spesifik mereka sendiri (meow() dan bark())

7

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:

```php
<?php
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);
?>
```



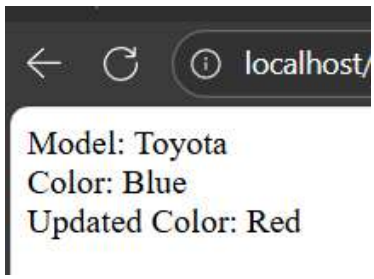
Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.3)
Pada kode ini memiliki interface Shape menetapkan "kontrak" bahwa setiap class yang mengimplementasikannya harus memiliki method calculateArea(). Class Circle dan Rectangle sama-sama mengimplementasikan Shape, tetapi dengan implementasi calculateArea() yang berbeda. Fungsi printArea() dapat menerima objek apapun yang bertipe Shape (baik Circle maupun Rectangle) dan memanggil method calculateArea()

8

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:

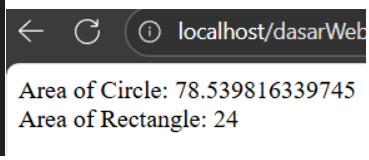
```php
<?php
3 references | 0 implementations | Windsurf: Refactor | Explain
class Car{
    2 references
    private $model;
    3 references
    private $color;

    3 references | 0 overrides | Windsurf: Refactor | Explain | Generate Func
    public function __construct($model, $color){
        $this->model = $model;
        $this->color = $color;
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Functi
    public function getModel(): mixed{
        return $this->model;
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Functi
    public function setColor($color): void{
        $this->color = $color;
    }

    2 references | 0 overrides | Windsurf: Refactor | Explain | Generate Func
    public function getColor(): mixed{
        return $this->color;
    }
}

$car = new Car(model: "Toyota", color: "Blue");
echo "Model: " . $car->getModel() . "<br>";
echo "Color: " . $car->getColor() . "<br>";

$car->setColor(color: "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)
Pada kode ini terdapat atribut $model dan $color pada class Car disetel sebagai private, artinya tidak bisa diakses langsung dari luar class. Untuk mengakses atau mengubah nilainya, method public (disebut getter dan setter) seperti getModel() dan setColor() harus digunakan. Dengan cara ini, data tidak bisa diubah langsung dari luar kelas ($objek->nama tidak bisa diakses langsung), melainkan harus lewat metode yang sudah disediakan. Tujuannya adalah melindungi data, mencegah perubahan yang tidak diinginkan, dan menjaga agar struktur program tetap aman serta konsisten..

9	**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:

```php
<?php
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)
Pada kode ini, class Shape didefinisikan sebagai abstract dan memiliki abstract method calculateArea(). Ini berarti class Shape tidak bisa dibuat objeknya secara langsung dan hanya berfungsi sebagai blueprint dasar. Class anak seperti Circle dan Rectangle yang extends Shape wajib menyediakan implementasi dari method calculateArea().Dengan abstraksi, kita menyembunyikan detail implementasi dan hanya menampilkan kerangka umum agar program lebih mudah dikembangkan tanpa harus tahu bagaimana setiap fungsi bekerja di dalamnya.

10	`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. Berikut adalah contoh penggunaan interface dalam PHP:

```php
<?php
6 references | 5 implementations | Windsurf: Refactor | Explain
interface Shape {
    4 references | 5 overrides
    public function calculateArea();
}

1 reference | 3 implementations | Windsurf: Refactor | Explain
interface Color {
    1 reference | 1 override
    public function getColor();
}

3 references | 0 implementations | Windsurf: Refactor | Explain
class Circle implements Shape, Color {
    6 references
    private $radius;
    2 references
    private $color;

    3 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
    public function __construct($radius, $color){
        $this->radius = $radius;
        $this->color = $color;
    }

    4 references | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
    public function calculateArea(): float|int{
        return pi() * pow(num: $this->radius, exponent: 2);
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Commen
    public function getColor(): mixed{
        return $this->color;
    }
}

$circle = new Circle(radius: 5, color: "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)
Pada kode ini, ada dua interface, Shape dengan method calculateArea dan Color dengan method getColor. Class Circle mengimplementasikan keduanya implements Shape, Color, sehingga class Circle diwajibkan untuk menyediakan implementasi untuk semua method dari kedua interface tersebut. Interface digunakan agar beberapa kelas berbeda dapat memiliki perilaku yang sama tanpa harus saling mewarisi.

10

`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.

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.

Destructor (Metode Penghancur)
Destructor menggunakan nama khusus __destruct dalam PHP. Destructor ini akan dipanggil secara otomatis ketika objek dihapus atau program selesai dieksekusi.

Berikut adalah contoh constructor dan destructor:

```php
<?php
2 references | 0 implementations | Windsurf: Refactor | Explain
class Car{
    2 references
    private $brand;

    2 references | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function __construct($brand){
        $this->brand = $brand;
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function getBrand(): mixed{
        return $this->brand;
    }

    0 references | 0 overrides | Windsurf: Refactor | Explain | Generate
    public function __destruct(){
        echo "The car is being destroyed.<br>";
    }
}

$car = new Car(brand: "Toyota");
echo "Brand: " . $car->getBrand() . "<br>";
?>
```

Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.7)
Pada kode ini, public function __construct() adalah constructor yang otomatis dipanggil saat sebuah objek baru (new Car(...)) dibuat, digunakan di sini untuk menginisialisasi properti $brand. Sebaliknya, public function __destruct() adalah destructor yang otomatis dipanggil saat objek akan dihapus atau di akhir eksekusi skrip, yang di sini digunakan untuk mencetak pesan "The car is being destroyed.".

12

Encapsulation and Access Modifiers

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.

PHP memiliki tiga access modifiers utama yang dapat digunakan dalam class:

`Public`: Properti atau metode yang dideklarasikan sebagai public dapat diakses dari luar class, sehingga mereka bersifat terbuka untuk diakses dari mana saja.

`Protected`: Properti atau metode yang dideklarasikan sebagai protected hanya dapat diakses dari dalam class itu sendiri dan dari class turunannya (inheritance).

`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.

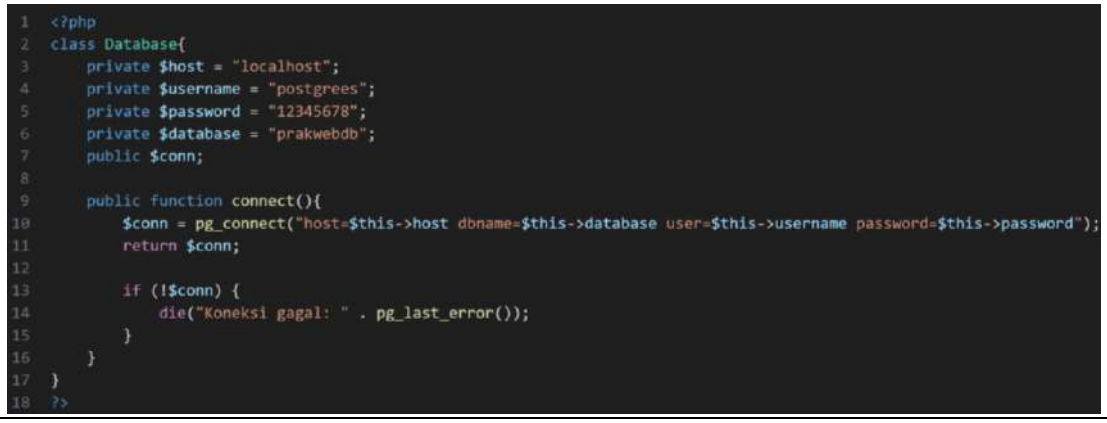
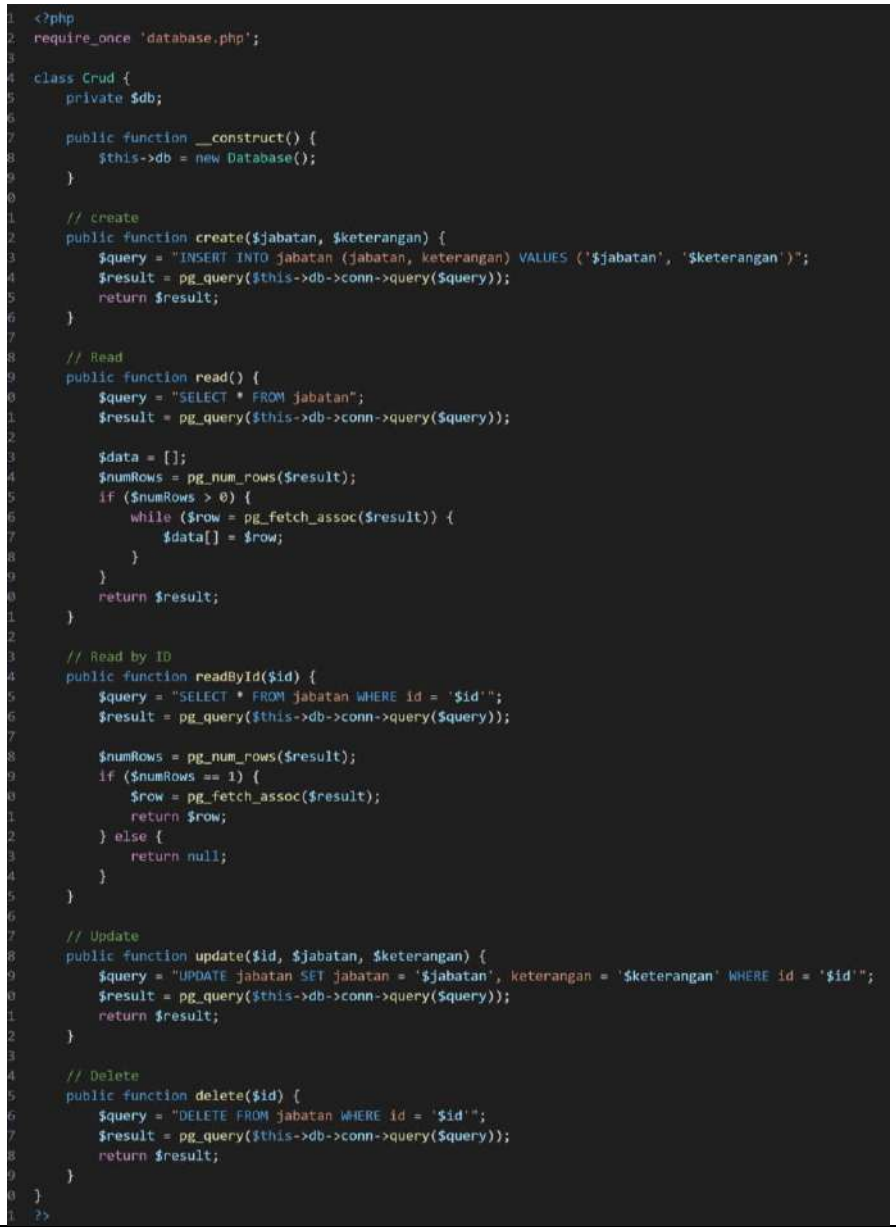
Berikut adalah contoh penggunaan access modifiers dalam PHP:

12

```php
<?php
3 references | 2 implementations | Windsurf: Refactor | Explain
class Animal {
    7 references
    public $name;
    2 references
    protected $age;
    2 references
    private $color;

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
    public function __construct($name, $age, $color){
        $this->name = $name;
        $this->age = $age;
        $this->color = $color;
    }

    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
    public function getName(): mixed{
        return $this->name;
    }
    1 reference | 0 overrides | Windsurf: Refactor | Explain | Generate Function Comme
    protected function getAge(): mixed{
        return $this->age;
    }
    1 reference | Windsurf: Refactor | Explain | Generate Function Comment
    private function getColor(): mixed{
        return $this->color;
    }
}

$animal = new Animal(name: "Dog", age: 3, color: "Brown");
echo "Name: " . $animal->getName() . "<br>";
echo "Age: " . $animal->getAge() . "<br>";
echo "Color: " . $animal->getColor() . "<br>";
?>
```

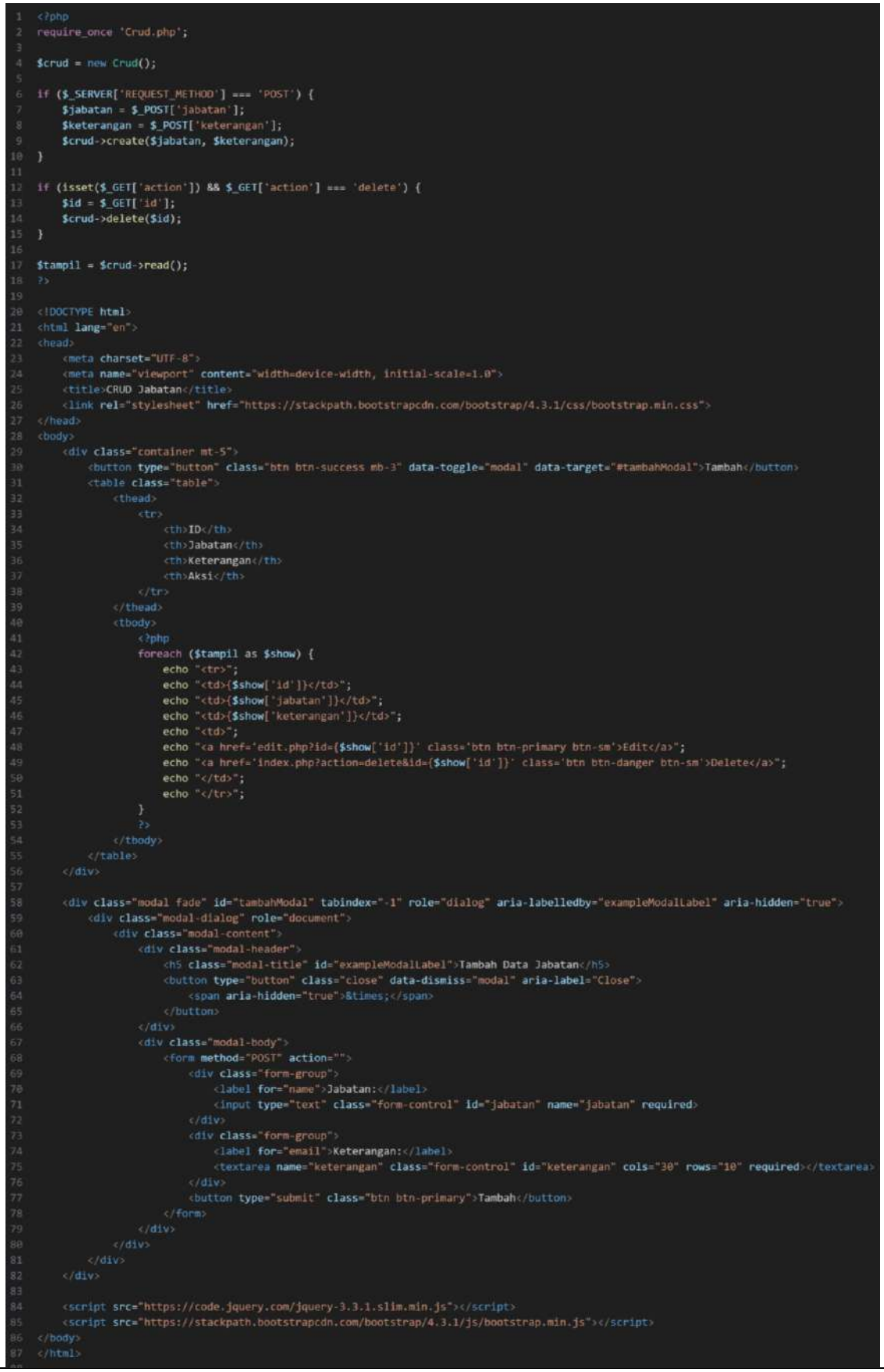
Apa yang anda pahami dari code di atas. Catat di bawah ini pemahaman anda. (soal no 1.8)
Pada kode ini, ethod public (getName) dapat diakses dari mana saja. Method protected (getAge) dan private (getColor) tidak dapat diakses dari luar class. Inilah mengapa kode menghasilkan Fatal Error saat mencoba memanggil getAge() dari global scope, membuktikan bahwa akses terproteksi tidak bisa dipanggil dari luar

Praktikum 2. CRUD dengan OOP

Langkah	Keterangan
1	Buat file baru pada `dasarWeb/JS12_OOP/database.php`. Ketikkan kode seperti di bawah ini.
2	
3	Buat file baru pada `dasarWeb/JS12_OOP/crud.php`. Ketikkan kode seperti di bawah ini.
4	

```php
<?php
class Database{
    private $host = "localhost";
    private $username = "postgrees";
    private $password = "12345678";
    private $database = "prakwebdb";
    public $conn;

    public function connect(){
        $conn = pg_connect("host=$this->host dbname=$this->database user=$this->username password=$this->password");
        return $conn;

        if (!$conn) {
            die("Koneksi gagal: " . pg_last_error());
        }
    }
}
?>
```

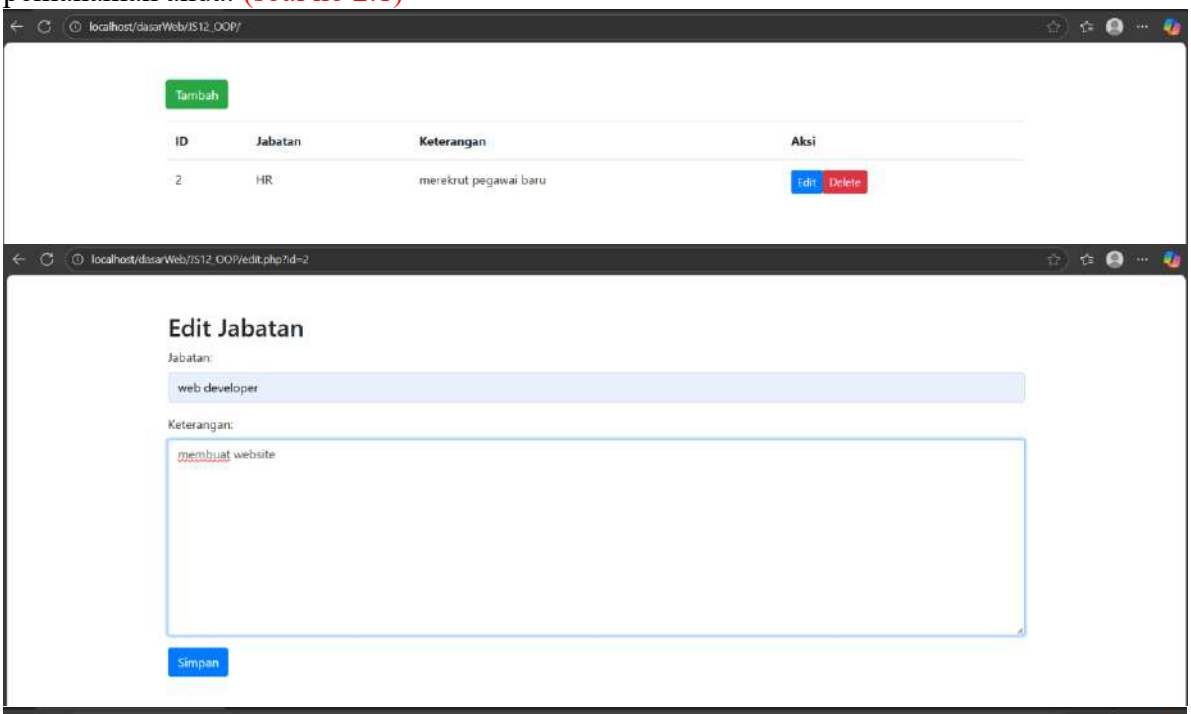
```php
<?php
require_once 'database.php';

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')";
        $result = pg_query($this->db->conn->query($query));
        return $result;
    }

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

        $data = [];
        $numRows = pg_num_rows($result);
        if ($numRows > 0) {
            while ($row = pg_fetch_assoc($result)) {
                $data[] = $row;
            }
        }
        return $result;
    }

    // Read by ID
    public function readById($id) {
        $query = "SELECT * FROM jabatan WHERE id = '$id'";
        $result = pg_query($this->db->conn->query($query));

        $numRows = pg_num_rows($result);
        if ($numRows == 1) {
            $row = pg_fetch_assoc($result);
            return $row;
        } else {
            return null;
        }
    }

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

    // Delete
    public function delete($id) {
        $query = "DELETE FROM jabatan WHERE id = '$id'";
        $result = pg_query($this->db->conn->query($query));
        return $result;
    }
}
?>
```

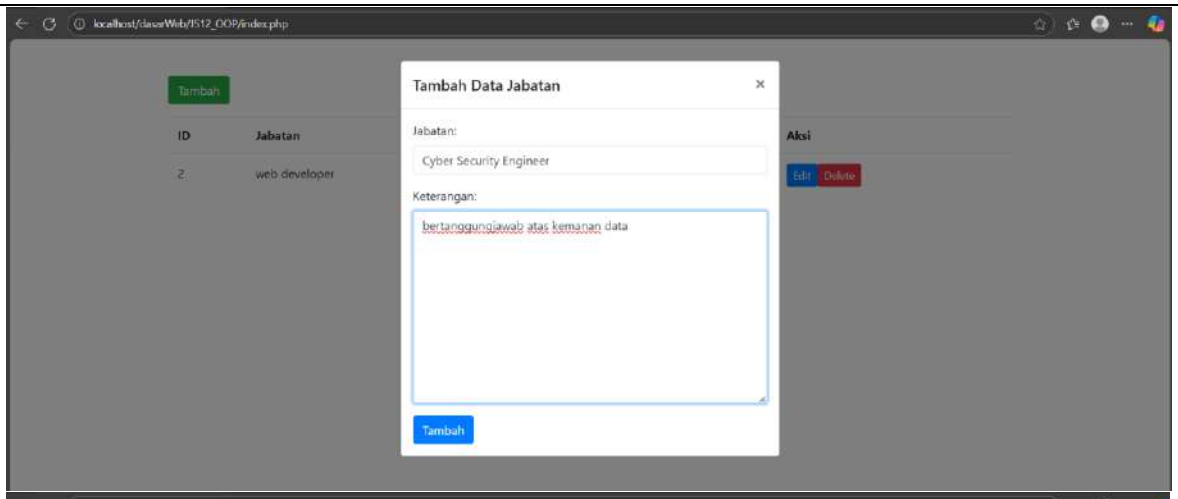
5	Buat file baru pada `dasarWeb/JS12_OOP/index.php`. Ketikkan kode seperti di bawah ini.
6	
7	Buat file baru `dasarWeb/JS12_OOP/edit.php`. Ketikkan kode seperti di bawah ini.

8

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

$crud = new Crud();

$id = $_GET['id'];

$tampil = $crud->readById($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 $tampil['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 $tampil['keterangan']; ?></textarea>
            </div>
            <input type="hidden" name="id" value="<?php echo $tampil['id']; ?>">
            <button type="submit" class="btn btn-primary">Simpan</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)





Penjelasan:

- database.php berisi class Database yang tugasnya hanya mengelola koneksi ke database PostgreSQL
- crud.php berisi class Crud yang menangani semua logika operasi database (INSERT, SELECT, UPDATE, DELETE) sebagai method-nya. Class ini menggunakan class Database untuk mendapatkan koneksi
- index.php adalah halaman utama yang menampilkan data (Read), menangani penambahan data (Create), dan penghapusan data (Delete). Halaman ini menggunakan object dari class Crud untuk berinteraksi dengan database.
- edit.php adalah halaman terpisah untuk mengubah data berdasarkan ID yang dikirim, yang juga menggunakan method dari class Crud