



Department of Information Technology, Malang State Polytechnic

Jobsheet-12: PHP – OOP

Web Design and Programming Courses

Tutor: Web Design and Programming Teaching Team

November 2024

Topic

- Dynamic Web Development Concepts with OOP

Name : Evan Diantha Fafian

Class : SIB 2G

Absent : 09

NIM : 2341760163

Purpose

Students are expected to:

1. Students are able to create classes and objects, inheritance, polymorphism, encapsulation, abstraction, interfaces, constructors and destructors, and encapsulation and access modifiers
2. Students are able to create CRUD with OOP

Attention

This jobsheet must be done step-by-step according to the practicum steps that have been given. Questions can be answered directly in the column provided using the PDF Editor.

Introduction

OOP

Object-Oriented Programming (OOP) is a very important programming paradigm in the world of software development. This allows developers to organize their code into objects that have associated attributes (data) and methods (functions).

Introduction to Object-Oriented Programming (OOP)

Object-Oriented Programming is based on the concept of objects, which represent entities in the real world. Each object has characteristics called attributes (properties), and can perform certain actions called methods (functions). OOP helps in breaking down the code into smaller, more manageable parts.

Why is OOP Important?

In the increasingly complex and dynamic world of website development, the use of Object-Oriented Programming (OOP) Concepts has become an essential foundation. OOP brings invaluable effectiveness, ease of maintenance, and scalability to website projects. This article will discuss why OOP is so important in website project development and its key benefits.

Modularity and Better Code Management

One of the main benefits of OOP is its ability to break code into independent modules or objects. In website development, each component such as forms, views, databases, and more can be represented as separate objects. This allows the development team to work separately on these components, speeding up the development process and allowing for easier maintenance in the future.

Reusability and Efficiency

In OOP, objects can be reused in different parts of the project. This reduces the amount of code that needs to be written, saving developers time and effort. For example, if you've created a "Form" object that has a method for validating input, you can use it on various pages of your website without needing to rewrite that validation code.

Better Error Management

When an error occurs in OOP code, you can easily isolate and find the source of the error because each object has a clear responsibility. This allows you to fix problems faster and more accurately, reducing the time spent on debugging.

Scalability and Collaborative Development

Website projects tend to evolve over time. With OOP, you can easily add new features or update existing components without interfering with other functions. The development team can also work in parallel on various components, as each object stands alone and is less dependent on the other.

Easier Maintenance

As a website project grows, maintenance becomes very important. OOP helps in separating the necessary changes to one component without affecting the others. If you want to change the appearance of a particular page, you just need to edit the view object without worrying about the impact on other components.

Encapsulation and Security

The concept of encapsulation in OOP allows you to hide implementation details from other components. This means that other components can only interact with objects through defined interfaces, reducing the potential for errors or unwanted manipulation.

Flexibility and Improved Code Quality

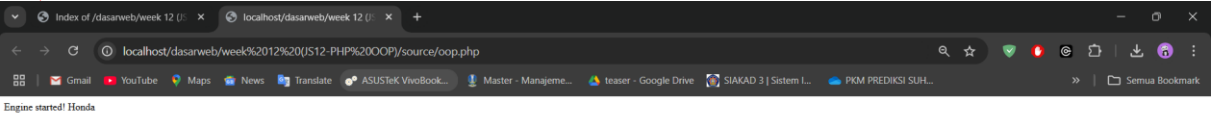
OOP allows you to create high abstractions to manage complexity and define common patterns. It improves code quality because it follows proven principles in software design, such as DRY (Don't Repeat Yourself) and SOLID (Split-Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Version Dependency).

Key Concepts in OOP PHP

In PHP, OOP allows you to organize and group code into more structured, manageable units. Here are the main concepts of OOP in PHP:

Practicum 1. Basic OOP

Step	Information
1	A class is a blueprint that defines the structure and behavior of an object. Class contain attributes (data) and methods (functions) that relate to those objects. Object , on the other hand, are concrete instances of a class, have real values for attributes and are capable of executing methods defined in the class. In PHP, you can create a class with the class keyword and then create an object from that class with the new keyword. Here is a simple example:
2	Create a folder dasarWeb/JS12_OOP and a new file called oop.php inside the folder.
3	Type this code into the oop.php file.

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>What do you understand from the code above. Record below your understanding. (Question No 1.1)</p>  <p>Engine started! Honda</p> <hr/> <ul style="list-style-type: none"> - The Car class defines a template with an attribute brand and a method startEngine. Two objects (\$car1 and \$car2) are created from the class, each with a unique brand value. \$car1 calls startEngine, while \$car2 only displays its brand. This demonstrates how to define and use class attributes and methods in PHP.
6	<p>Inheritance is one of the basic concepts in object-oriented programming (OOP) that allows a class to inherit properties and methods from other classes. An inherited class is called a subclass or child class, while a class that provides inheritance is called a superclass or parent class. This concept allows us to reuse code, extend functionality, and build class hierarchies.</p> <p>The following is a simple example of the implementation of inheritance in 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>";
    }
}

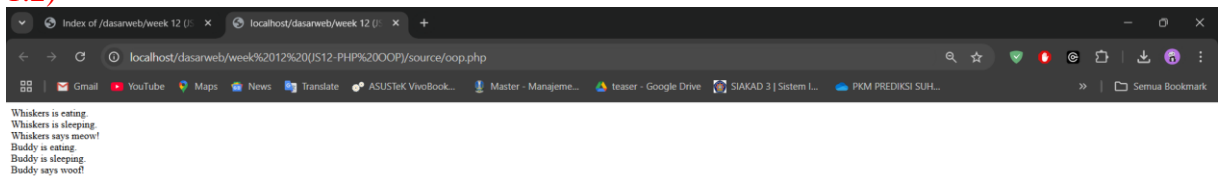
$cat = new Cat("Whiskers");
$dog = new Dog("Buddy");

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

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

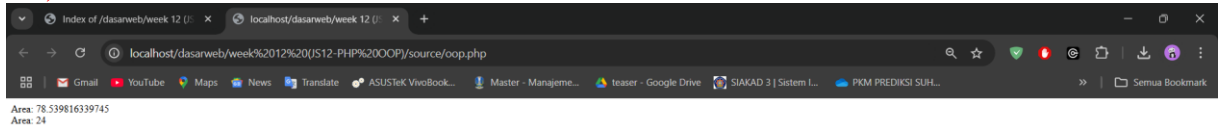
```

What do you understand from the code above. Record below your understanding. (Question No 1.2)



	<ul style="list-style-type: none"> - The code illustrates inheritance in OOP, where Cat and Dog classes inherit properties and methods from the Animal class. The Animal class has methods eat and sleep, while Cat and Dog classes add their unique methods (meow and bark). This demonstrates code reuse and extension via inheritance.
7	<p>Polymorphism is a concept in object-oriented programming that allows objects of different classes to respond to method calls in the same way. This can be realized in PHP through the use of interfaces and the use of overriding methods. With polymorphism, you can treat objects of different classes in a uniform way.</p> <p>Here is a simple example of using polymorphism in PHP using the interface:</p> <pre> 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() . "
"; } \$circle = new Circle(5); \$rectangle = new Rectangle(4, 6); printArea(\$circle); printArea(\$rectangle); </pre>

What do you understand from the code above. Record below your understanding. (Question No 1.3)



- The code demonstrates polymorphism using an interface Shape that has a calculateArea method. Different classes (Circle and Rectangle) implement the interface and provide their specific calculateArea implementations. The printArea function accepts any Shape object, demonstrating polymorphism by treating different shapes uniformly.

8

Encapsulation is one of the concepts in object-oriented programming (OOP) that allows encapsulation of properties and methods in a class so that access to them can be controlled. This can help in applying access management principles and ensure that properties and methods that may change in the future do not compromise the integrity of the class or program as a whole.

Here is a simple example of encapsulation in 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");

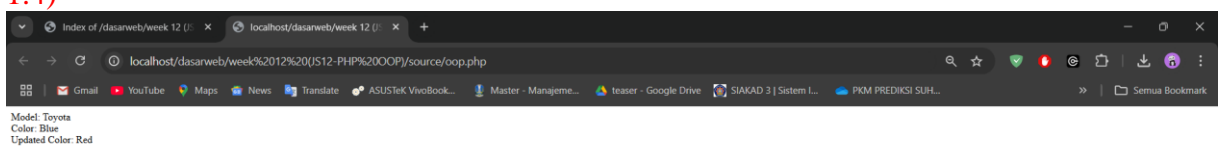
echo "Model: " . $car->getModel() . "<br>";
echo "Color: " . $car->getColor() . "<br>";

$car->setColor("Red");

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

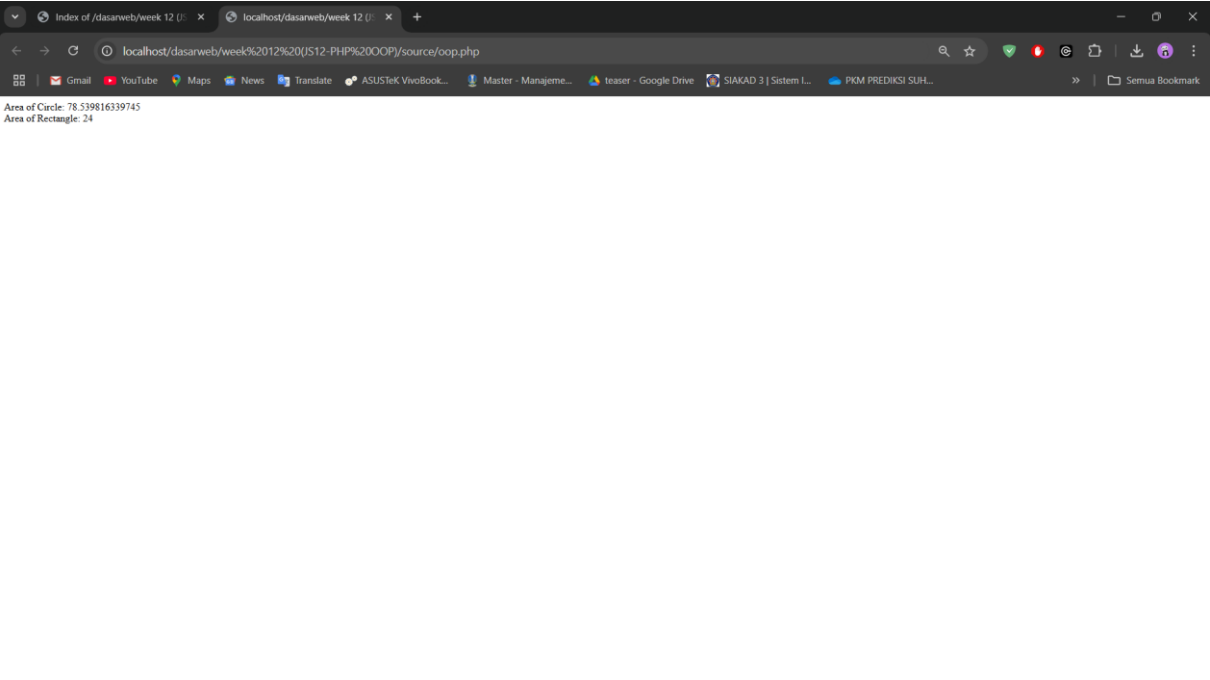
```

What do you understand from the code above. Record below your understanding. (Question No 1.4)



Model: Toyota
Color: Blue
Updated Color: Red

	<ul style="list-style-type: none">- The example shows encapsulation in OOP. The Car class has private properties (model and color) and provides public methods (getModel, getColor, setColor) to access or modify these properties. This approach hides internal details and only exposes necessary functionality.
9	<p>Abstraction is one of the basic concepts in object-oriented programming (OOP) that allows you to hide internal details and expose only the necessary functionality. It helps in creating classes and methods that are general and flexible, allowing users to interact with objects without needing to know their internal implementations.</p> <p>Here's a simple example of abstraction in PHP using abstract classes and methods:</p> <pre>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() . "
"; echo "Area of Rectangle: " . \$rectangle->calculateArea() . "
";</pre> <p>What do you understand from the code above. Record below your understanding. (Question No 1.5)</p>

	 <p>- This code example demonstrates abstraction using abstract classes. The Shape abstract class defines a calculateArea method, and derived classes (Circle and Rectangle) implement this method with specific calculations. The abstract class allows for defining a common interface without knowing the details of each shape's implementation.</p>
10	<p>An interface is a concept in object-oriented programming that allows the definition of a contract or framework that the classes that implement it must follow. Interfaces don't have their own implementations, but only provide a declaration of methods and properties that the classes that use them must implement. This makes it possible to achieve polymorphism without requiring a single inheritance, so that a class can implement multiple interfaces.</p> <p>Here is an example of using the interface in 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);
    }

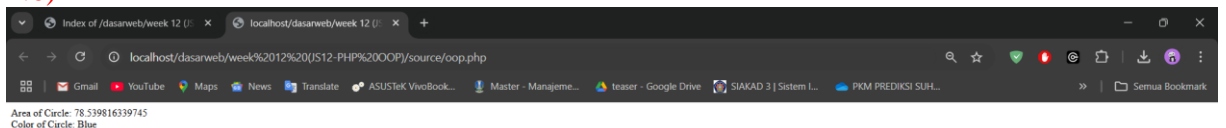
    public function getColor()
    {
        return $this->color;
    }
}

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

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

```

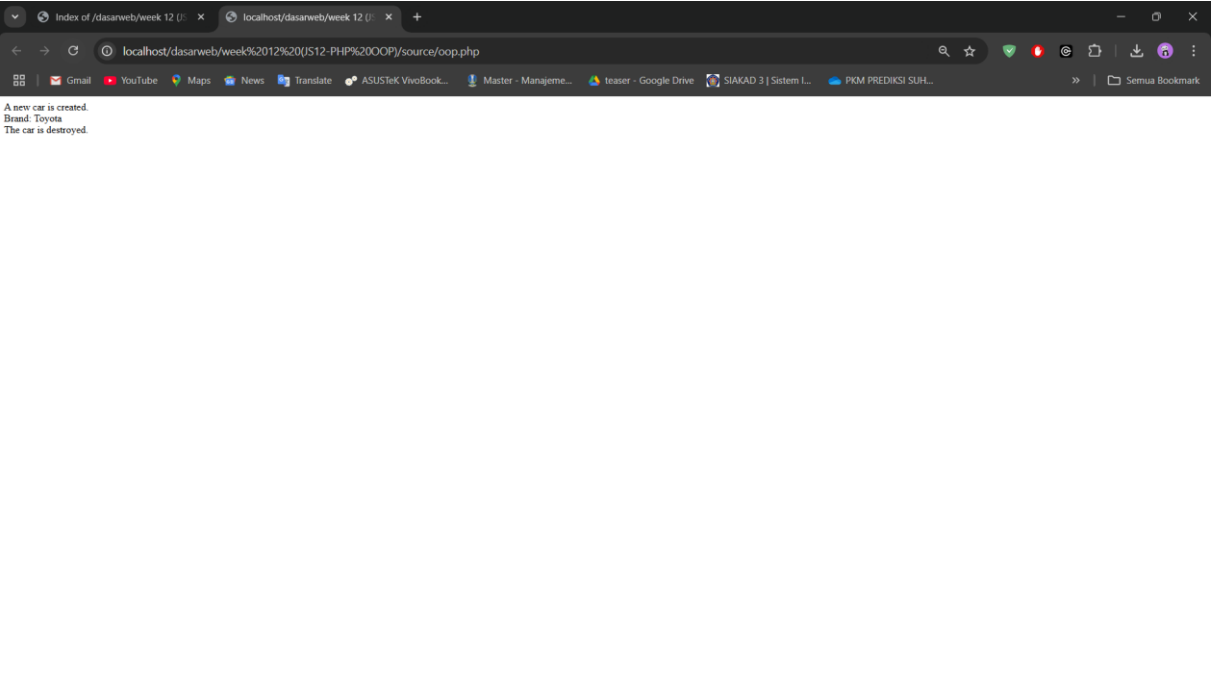
What do you understand from the code above. Record below your understanding. (Question No 1.6)



Area of Circle: 78.539816339745
Color of Circle: Blue

- The code explains **interfaces** in OOP. The Shape and Color interfaces define methods that any implementing class must have. The Circle class implements both interfaces and

	provides the required methods. Interfaces allow multiple inheritance of behaviors without sharing the base class.
11	<p>Constructors and destructors are special methods in object-oriented programming (OOP) used in PHP to initialize and clean objects. A constructor is a method that is called automatically when a new object is created, whereas a destructor is a method that is called automatically when an object is deleted or no longer in use.</p> <p>Constructor The constructor uses <code>__construct</code> special names in PHP. This constructor will be called automatically whenever a new object is created from a class that contains that constructor.</p> <p>Destructor The destructor uses <code>__destruct</code> special names in PHP. This destructor will be called automatically when the object is deleted or the program finishes executing.</p> <p>Here are examples of constructors and destructors:</p> <pre>class Car { private \$brand; public function __construct(\$brand) { echo "A new car is created.
"; \$this->brand = \$brand; } public function getBrand() { return \$this->brand; } public function __destruct() { echo "The car is destroyed.
"; } } \$car = new Car("Toyota"); echo "Brand: " . \$car->getBrand() . "
";</pre> <p>What do you understand from the code above. Record below your understanding. (Question No 1.7)</p>

	 <ul style="list-style-type: none">- This code introduces constructors and destructors in PHP. The constructor (<code>__construct</code>) initializes an object when it is created, and the destructor (<code>__destruct</code>) performs cleanup before the object is destroyed. This helps manage resources effectively, especially when dealing with complex data or external connections.
12	<h2>Encapsulation and Access Modifiers</h2> <p>Encapsulation is one of the key concepts in object-oriented programming (OOP), and it involves wrapping data (variables) and methods (functions) in a class. This helps in hiding the internal implementation of a class and exposing only the necessary functionality. Access modifiers are a subset of encapsulation that allows you to control the level of access to properties and methods in a class.</p> <p>PHP has three main access modifiers that can be used in classes:</p> <p>Public: Properties or methods that are declared public can be accessed from outside the class, so they are open to access from anywhere.</p> <p>Protected: Properties or methods that are declared protected can only be accessed from within the class itself and from its child classes (inheritance).</p> <p>Private: Properties or methods that are declared private can only be accessed from within the class itself. They cannot be accessed from outside the class, not even by its child classes.</p> <p>Here is an example of using access modifiers in 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;
    }

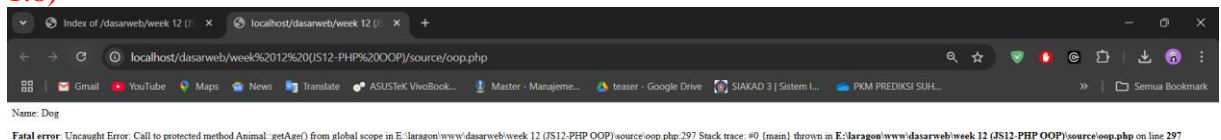
    private function getColor()
    {
        return $this->color;
    }
}

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

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

```

What do you understand from the code above. Record below your understanding. (Question No 1.8)



- This code demonstrates encapsulation with access modifiers (public, protected, and private). Each modifier controls the accessibility of class properties and methods. public properties can be accessed from anywhere, protected within the class and its subclasses,

	and private only within the class itself. This helps control how data is accessed and modified.
--	---

Practicum 2. CRUD with OOP

Step	Information
1	Create a new file <code>dasarWeb/JS12_OOP/database.php</code> . Type the code as below.
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	Create a new file <code>dasarWeb/JS12_OOP/crud.php</code> . Type the code as below.

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

Create a new file [dasarWeb/JS12_OOP/index.php](#). Type the code as below.

```

<?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

Create a new file `dasarWeb/JS12_OOP/edit.php`. Type the code as below.

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">

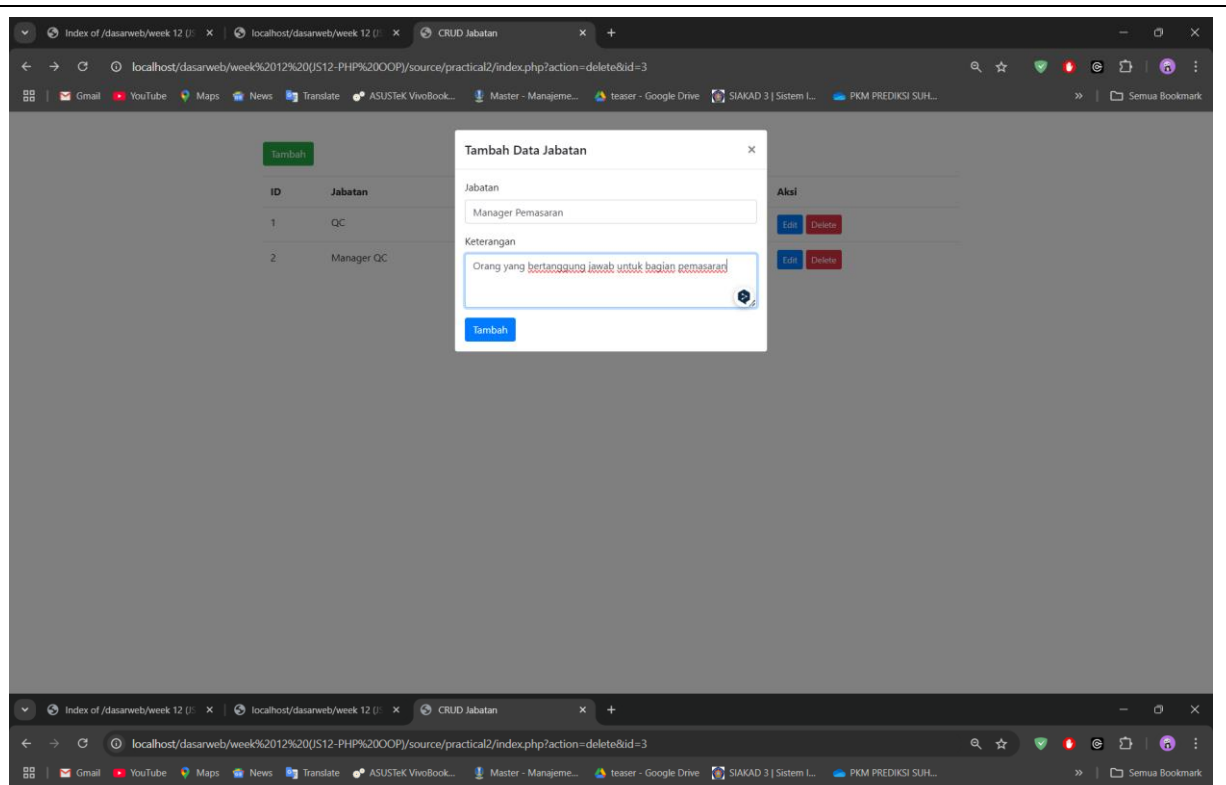
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Edit Jabatan</title>
    <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.3.1/css/bootstrap.min.css">
</head>

<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>
```

Run the code in practicum 2. What do you understand from the code above. Record below your understanding. (Question No 2.1)



Tambah			
ID	Jabatan	Keterangan	Aksi
1	QC	Quality Control	Edit Delete
2	Manager QC	Penanggung Jawab QC	Edit Delete
4	Manager Pemasaran	Orang yang bertanggung jawab untuk bagian pemasaran	Edit Delete

Index of /dasarweb/week 12 | localhost/dasarweb/week 12 | Edit Jabatan

localhost/dasarweb/week%2012%20(JS12-PHP%20OOP)/source/practical2/edit.php?id=4

Gmail YouTube Maps News Translate ASUSTeK VivoBook... Master - Manajeme... teaser - Google Drive SIAKAD 3 | Sistem L... PKM PREDIKSI SUH... Semua Bookmark

Edit Jabatan

Jabatan

Manager Pemasaran

Keterangan

Penanggung Jawab pemasaran

Update

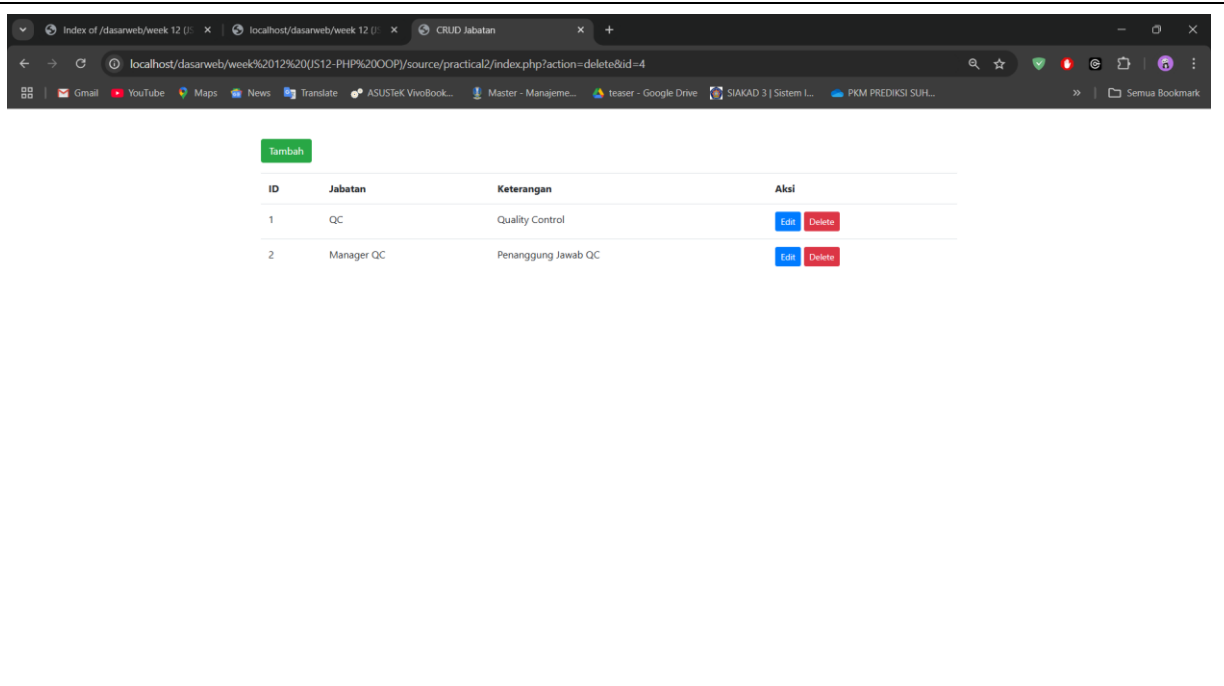
Index of /dasarweb/week 12 | localhost/dasarweb/week 12 | CRUD Jabatan

localhost/dasarweb/week%2012%20(JS12-PHP%20OOP)/source/practical2/index.php

Gmail YouTube Maps News Translate ASUSTeK VivoBook... Master - Manajeme... teaser - Google Drive SIAKAD 3 | Sistem L... PKM PREDIKSI SUH... Semua Bookmark

Tambah

ID	Jabatan	Keterangan	Aksi
1	QC	Quality Control	Edit Delete
2	Manager QC	Penanggung Jawab QC	Edit Delete
4	Manager Pemasaran	Penanggung Jawab pemasaran	Edit Delete



- This code demonstrates a foundational example of integrating PHP, HTML, and SQL to create a simple yet functional CRUD application with a user-friendly interface. It's a practical implementation of database operations within a structured PHP OOP environment.