Department of Information Technology, State Polytechnic of Malang

Jobsheet-12: PHP – OOP

Web Design and Programming Courses

Web Design and Programming Teaching Team

November 2024

Topic

- Dynamic Web Creation Concepts with OOP

Objectives

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

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.

Encapsulation and Access Modifiers

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.
PHP has three main access modifiers that can be used in classes:

- Public (public): Properties or methods that are declared public can be accessed from outside the class, so they are open to access from anywhere.
- Protected (protected): Properties or methods that are declared protected can only be accessed from within the class itself and from its child classes (inheritance).
- Private (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.

Flexibility and Improved Code Quality

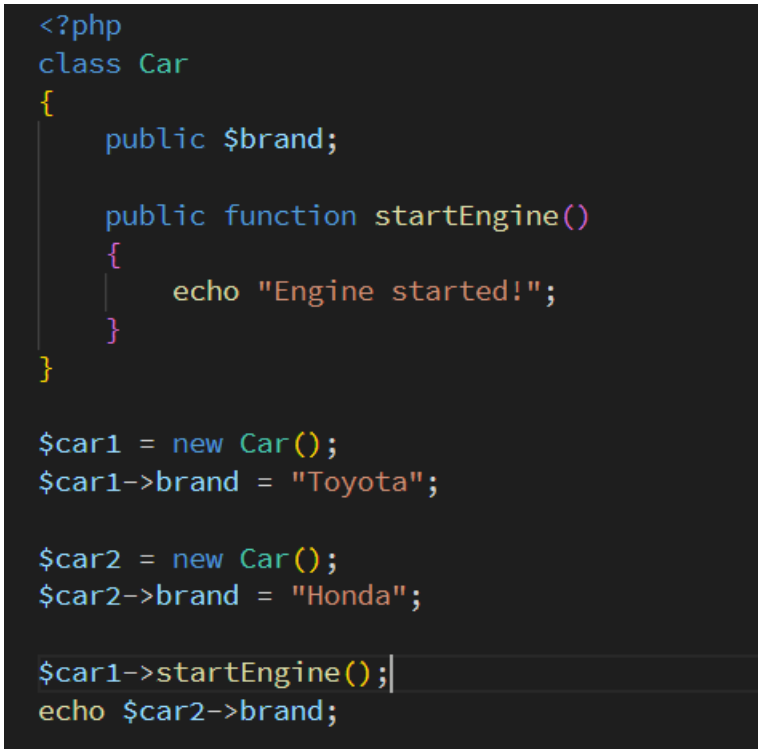
OOP allows you to create high abstractions to manage complexity and define common patterns. It improves code quality because it follows proven principles in software design, such as DRY (Don't Repeat Yourself) and SOLID (Split-Responsibility, Open-Closed, Liskov Substitution, Interface Segregation, Version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

Practical Section 1. Basic OOP

Step	Information
1	A class is a blueprint or blueprint that defines the structure and behavior of an object. Classes contain attributes (data) and methods (functions) that relate to those objects. Objects, 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 for week12, in the foldet create a new file that is `oop.php`.
3	Type into the `oop.php` file the code below.
4	
5	What do you understand from the code above. Record below your understanding. (Question No 1.1)
6	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. The following is a simple example of the implementation of inheritance in PHP:

```php
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>";
    }
}
```

```php
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)

6	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.

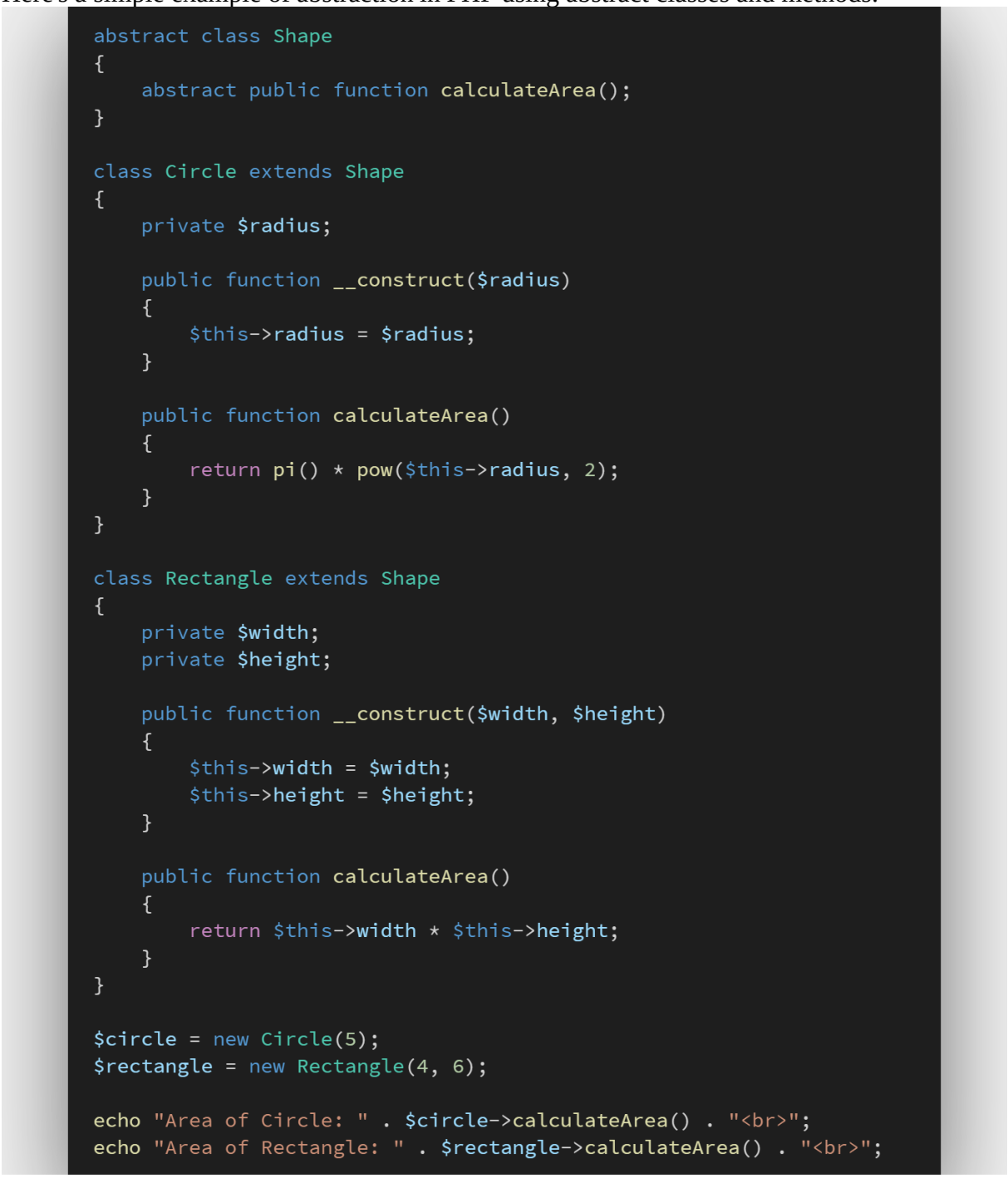
Here is a simple example of using polymorphism in PHP using the interface:

```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);
```

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

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

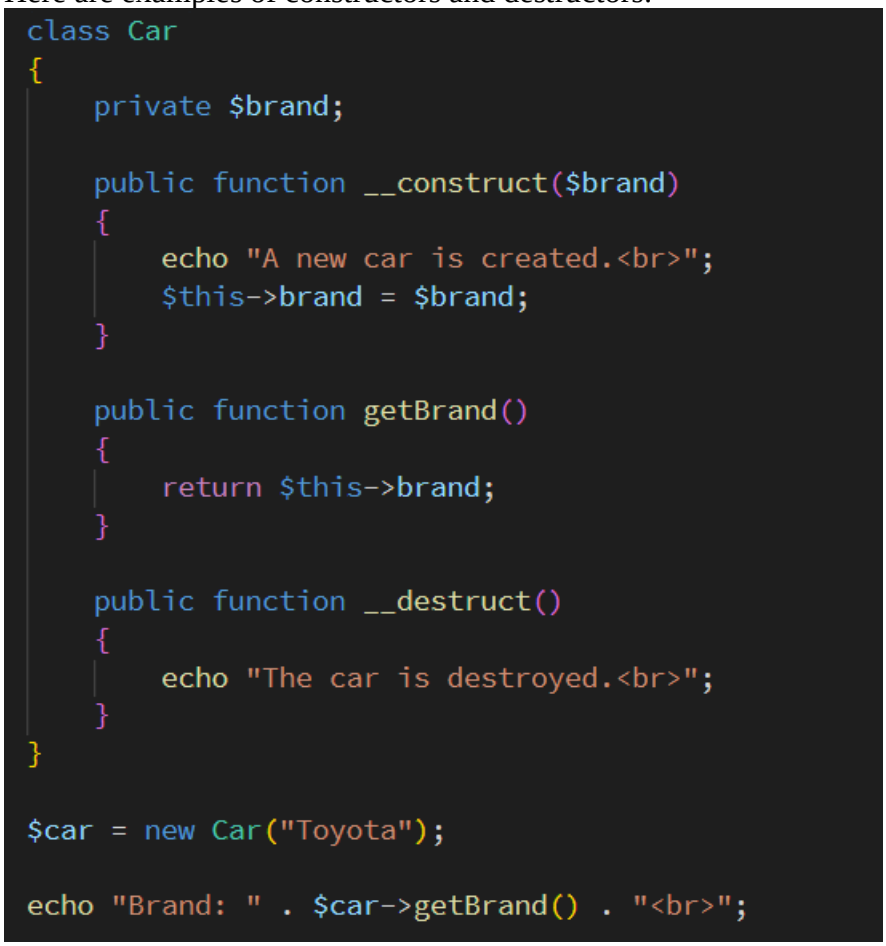
```php
class Car
{
    private $model;
    private $color;

    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>";
```

8	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. Here's a simple example of abstraction in PHP using abstract classes and methods:  What do you understand from the code above. Record below your understanding. (Question No 1.5)

```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>";
```

9	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. Here is an example of using the interface in PHP:  What do you understand from the code above. Record below your understanding. (Question No 1.6)

```php
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>";
```

10	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. Constructor (Metode Pembuat) The constructor uses __construct special names in PHP. This constructor will be called automatically whenever a new object is created from a class that contains that constructor. Destructor (Demolition Method) The destructor uses __destruct special names in PHP. This destructor will be called automatically when the object is deleted or the program finishes executing. Here are examples of constructors and destructors:  What do you understand from the code above. Record below your understanding. (Question No 1.7)
11	Encapsulation is one of the key concepts in object-oriented programming (OOP), and it involves wrapping data (variables) and methods (functions) in a class. Here's an example of using access modifiers for encapsulation in PHP:

```php
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>";
```

```php
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)

Practical Section 2. CRUD with OOP

Step	Information
1	Create a new file on the week12 folder with a new name named `database.php`. Type the code as below.
2	 ```php 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); } } ```
3	Create a new file on the week12 folder with a new name named `crud.php`. Type the code as below.
4	 ```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 = $this->db->conn->query($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; } ```

```php
    // 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);                                    d

        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 on the week12 folder with a new name named `index.php`. Type the code as below.

```php
<?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);
}

$tampil = $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">
        <table class="table">
            <thead>
                <tr>
                    <th>ID</th>
                    <th>Jabatan</th>
                    <th>Keterangan</th>
                    <th>Aksi</th>
                </tr>
            </thead>
            <tbody>
                <?php
                foreach ($tampil 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>
```

```
    <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>
```

6

Create a new file in the week12 folder with a new name named `edit.php`. Type the code as below.

```
<?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">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>
```

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