LATIHAN PRAKTEK OOP MVC

Nama : Mazda Nawallsyah

NIM : G.211.21.0082

Kelas : Teknik Informatika B (Pagi)

Link GitHub : https://github.com/MazdaAzda/Framework2.git

1. Lat4_1

A. Tampilan

B. Setelah menggunakan Method overload getinfo dengan $a

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

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Document</title>
    <link rel="stylesheet" href="   https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/css/bootstrap.min.css">
    <script defer src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.1/dist/js/bootstrap.bundle.min.js"></script>
</head>

<body>
    <?php
    //class mobil
    class Mobil
    {
        public $nama;
        public $merk;
        function getInfo($a)
        {
            echo "Nama mobil: " . $a->nama . "<br/>";
            echo "Merk: " . $a->merk . "<br/>";
        }
    }

    ?>
    <div class="container mt-5 bg-gray-300">
        <div class="card pt-3 px-3 shadow mx-auto" style="width: 400px">
            <div class="card-header text-center text-bg-danger  border-0 rounded">
                <h4>Tugas 2 || Lat4_1.php </h4>
            </div>
            <div class="card-body">
                <p>
                    <?php
                    $a = new Mobil();
                    $a->nama = "Mazda RX-Vision GT3";
                    $a->merk = "Mazda";
                    $a->getInfo($a);
                    ?>
                </p>
            </div>
            <div class="btn-group btn-group-sm" style="width: 40px;">
                <a href="Lat4_2a.php" class="btn btn-danger">
                    < </a>
            </div>
            <p style="font-size: 15px;" class="text-center mt-2">
                Mazda Nawallsyah || G.211.21.0082
            </p>
        </div>
    </div>
</body>

</html>
```

C. Kesimpulan :

1. Cara membuat class pada php
   <?php
   Class nama_class{ }
   ?>
2. Cara penulisan property
   Modifier $nama_properti;
3. Penulisan methodModifier function nama_method(){Isi_method;}
4. Cara inisiasi object$nama_object = new nama_class();
5. Cara mengisi property atau mendefinisikan property$nama_object->properties=”aaa”;
6. Cara memanggil/menjalankan method pada suatu class$nama_object->nama_methode();

2. Lat4_2a dan Lat4_2b

a. menambahkan construct pada Lat4_2a

script 4_2a

```php
<?php
class mahasiswa
{
    public $nama;
    public $nim;
    public $prodi;
    public $semester;
    function __construct($a, $b, $c, $d)
    {
        $this->nama = $a;
        $this->nim = $b;
        $this->prodi = $c;
        $this->semester = $d;

        echo "Kelas telah dibuat<br/><br/>";
    }
    function cetak()
    {
        echo $this->nama . "<br/>" . $this->nim . "
        echo $this->prodi . "<br/>" . $this->semest
    }
    function __destruct()
    {
        echo "Kelas telah dihancurkan";
    }
}
class mahasiswi
{
    public $nama;
    public $nim;
    public $prodi;
    public $semester;
    function __construct($a, $b, $c, $d)
    {
        $this->nama = $a;
        $this->nim = $b;
        $this->prodi = $c;
        $this->semester = $d;

        echo "Kelas telah dibuat<br/><br/>";
    }
    function cetak()
    {
        echo $this->nama . "<br/>" . $this->nim . "
        echo $this->prodi . "<br/>" . $this->semest
    }
    function __destruct()
    {
        echo "Kelas telah dihancurkan";
    }
}
```

script 4_2b

```php
<?php
require_once("lat4_2a.php");
$mhs2 = new mahasiswa("Mazda Nawallsyah", "
21.0082", "F T I K", "Semester 3");
$mhs2->cetak();
```

b. Tampilan

Kelas telah dibuat

Mazda Nawallsyah
G.211.21.0082
F T I K
Semester 3

Kelas telah dihancurkan

c. Kesimpulan : Contructor dalam php oop tidak bisa di override

3. Lat4_3a dan Lat4_3b

a. Ya program tersebut mengalami error, hal itu dikarenakan property yang bermodifier private hanya bisa digunakan pada class mahasiswa sendiri.

b. Modifier protected dan Public
b.1 Protected
b.2 Public

script :

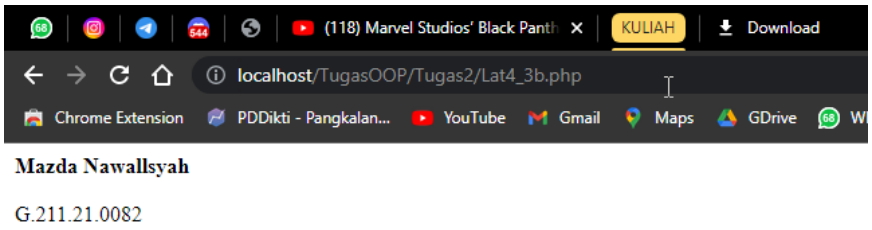
a. 4_3a

```php
<?php
class mahasiswa
{
    public $nama;
    public $nim;
    function __construct()
    {
    }
    function setNama($a)
    {
        $this->nama = $a;
    }
    function setNim($b)
    {
        $this->nim = $b;
    }
    function getNama()
    {
        return $this->nama;
    }
    function getNim()
    {
        return $this->nim;
    }
    function destruct()
    {
    }
}
```

b. 4_3b

```php
<?php
require_once("lat4_3a.php");
$mhs1 = new mahasiswa();
$mhs1->nama = "<b>Mazda Nawallsyah</b></br></br>";
$mhs1->nim = "G.211.21.0082";
echo $mhs1->nama;
echo $mhs1->nim;
```

Hasil :



c. Kesimpulan :

1. Penggunaan modifier
   Modifier Keterangan Public
   Untuk mendefinisikan data atau metode yang akan terlihat dariluar oleh siapapun dan dimanapun.

   Private
   Untuk mendefinisikan data atau metode agar hanya terlihatpada class/object itu sendiri.

   Protected
   Untuk mendefinisikan data atau metode untuk tidak terlihat dariluar (seperti private), tetapi akan dapat diakses oleh "anak" dari class tersebut.

2. Modifier protected dan private, properties bisa dipanggil dengan mengimplementasikan setter-getter.

3. Lat4_4a dan Lat4_4b
   a. Tampilan
   b. Kesimpulan : Pada php oop, class asisten (child) bisa memanggil method dari mahasiswa (parent).

4. Lat4_5a dan Lat4_5b

a. Script

4_5a

```php
<?php
abstract class mahasiswa
{
    abstract protected function getTugasAkhir();
    abstract protected function getProgram($postfix);
    public function tugasAkhir()
    {
        print $this->getTugasAkhir() . "<br>";
    }
}
class sarjana extends mahasiswa
{
    protected function getTugasAkhir()
    {
        return "Skripsi";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S1";
    }
}
class magister extends mahasiswa
{
    public function getTugasAkhir()
    {
        return "Tesis";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S2";
    }
}
```

4_5b

```php
<?php
require_once("lat4_5a.php");
$s = new sarjana;
$s->getProgram('Mahasiswa') . "<br>";
$s->tugasAkhir();
$m = new magister;
$m->getProgram('Mahasiswa') . "<br>";
$m->tugasAkhir();
```

Mahasiswa S1Skripsi
Mahasiswa S2Tesis

b. Setelah menghapus baris ke 29-32

```php
<?php
abstract class mahasiswa
{
    abstract protected function getTugasAkhir();
    abstract protected function getProgram($postfix);
    public function tugasAkhir()
    {
        print $this->getTugasAkhir() . "<br>";
    }
}
class sarjana extends mahasiswa
{
    protected function getTugasAkhir()
    {
        return "Skripsi";
    }
    public function getProgram($postfix)
    {
        print "{$postfix} S1";
    }
}
class magister extends mahasiswa
{

    public function getProgram($postfix)
    {
        print "{$postfix} S2";
    }
}
```

Fatal error: Class magister contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (mahasiswa::getTugasAkhir) in **C:\xampp\htdocs\TugasOOP\Tugas2\Lat4_5a.php** on line **22**

c. Kesimpulan :

class anak yang mewarisi super class harus menuliskan semua method abstrak dari super classnya.

5. Lat4_6

a. Maksud dari program diatas adalah penggunaan object interface

b. Tampilan

```php
<?php
interface a
{
    public function foo();
}
interface b
{
    public function bar();
}
interface c extends a, b
{
    public function baz();
}
class d implements c
{
    public function foo()
    {
    }
    public function bar()
    {
    }
    public function baz()
    {
    }
}
```

Error, karena pada object interface, ketika kita mengimplementasikan object tersebut, seluruh method pada interface harus diimplementasikan seluruhnya. Karena class d mengimplement interface c, maka method-method pada interface c harus diimplementasikan seluruhnya.

Error, method foo dan bar terdapat pada interace a dan b, pada penerapannya sebuah class

c. Setelah menambahkan class baru yaitu class e

```php
<?php
interface a
{
    public function foo();
}
interface b
{
    public function bar();
}
interface c extends a, b
{
    public function baz();
}
class d implements c
{
    public function foo()
    {
    }
    public function bar()
    {
    }
    public function baz()
    {
    }
}
class e
{
    public function foo()
    {
    }
    public function bar()
    {
    }
}
```

Error, method foo dan bar terdapat pada interace a dan b, pada penerapannya sebuah classbaru tidak dapat mengimplementasikan method yang sama pada dua interface.

d. Kesimpulan
   1. Interface didefinisikan dengan “Interface” keyword, mirip dengan deklarasi class biasa, hanya saja definisi atau detail method tidak dituliskan.
   2. Seluruh method yang dideklarasikan pada interface harus memiliki modifier “ public ”.
   3. Untuk mengimplementasikan sebuah interface, kita dapat menggunakan “implement” keyword.
   4. Seluruh method yang ada pada interface harus diimplementasikan seluruhnya. Sebuah class bisa mengimplementasikan lebih dari satu interface.
   5. Class tidak bisa mengimplementasikan dua interface yang mempunyai nama method yang sama.
   6. Interface bisa diwariskan seperti class menggunakan “extends”.
   7. Class yang mengimplementasikan interface harus menggunakan method-method yangada pada interface tersebut dengan nama dan spesifikasi yang sama persis.

6. Lat4_7

a. Tampilan program (program tersebut mengalami error)

b. Setelah memindahkan kata final dari baris 5 ke baris 2 (program tersebut tetap mengalami error),,error pada program tersebut dikarenakan...

c. Kesimpulan

1. final, akan mencegah proses overriding method pada class anak (sub-class)
2. Apabila metode kita berikan status final, maka metode tersebut tidak akan bisa dioverride, begitu juga pada class, apabila kita berikan status “final” pada deklarasi class maka class tersebut tidak bisa diperpanjang (diwariskan).

7. Lat4_8

a. Tampilan

```php
<?php
class One
{
    private static $var = 0;
    function __construct()
    {
    }
    static function disp()
    {
        print self::$var;
    }
    function __destruct()
    {
    }
}
One::disp();
```

b. Kesimpulan :

1. Lat4_8.php mengimplementasikan penggunaan property static dengan modifierprivate.Properti statis dideklarasikan dengan menggunakan katakunci statis sebelum modifier- Sintaks:

   modifier static $nama_property = nilai;

2. Sifat statis dapat diakses tanpa perlu sebuah contoh objek dari kelas, menggunakannama kelas bersama dengan ::- Sintaks:

   ClassName :: $nama_property method_name();

   Perhatikan bahwa properti statis menggunakan tanda dollar ($).Properti statis tidak dapat diakses melalui obyek menggunakan operator panah "->"