

TEORI PEMROGRAMAN BERORIENTASI OBJEK

21 Mei 2024

ABSTRACT

1. Dasar

a. Abstract

Abstract class adalah class yang mengandung minimal satu buah abstract method, yaitu method yang hanya berupa nama method dan argumennya (jika ada) dan diawali dengan keyword abstract.

Ketika sebuah class dibuat dengan cara menurunkan atau class inheritance dari abstract class maka semua method yang didefinisikan sebagai abstract oleh parent class harus diimplementasikan ulang oleh class tersebut yang sebagai child class atau class turunan dari abstract class.

Contoh :

- Abstract Class

```
2 references | 2 implementations
3  abstract class Vehicle {
4
5  }
```

- Abstract Method

```
abstract class Vehicle {
  2 references | 2 overrides
  abstract public function go();
  2 references | 2 overrides
  abstract public function stop();
}
```

b. Aturan Abstract

Berikut aturan-aturan disertai contoh agar lebih mudah memahami:

1. Cara pembuatan abstract class dan abstract method

Cara membuatnya yaitu harus didahului dengan *keyword* `abstract` sebelum *class* dan *method*.

```
<?php
// diawali keyword abstract
abstract class User
{
    // diawali keyword abstract
    abstract protected function showName();
}
```

2. Abstract class tidak bisa dijadikan object

```
<?php
abstract class User
{
    //
}
```

```
// membuat object dari abstract class
```

```
$thisUser = new User();
```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Uncaught Error: Cannot instantiate abstract class ... karena ini adalah abstract class yang tujuannya sebagai base class atau class acuan bukan untuk digunakan sebagai object.

3. Jika dalam sebuah class terdapat abstract method maka class tersebut harus menjadi abstract class.

```
<?php
// bukan abstract class
class User
{
    // tapi ini abstract method
    abstract protected function showName();
}
```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Class User contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (User::showName) ... karena class User bukan abstract class tetapi memiliki abstract method, ini tidak diperbolehkan.

4. Abstract method hanya boleh signature

Artinya *abstract method* **tidak boleh memiliki body**, yaitu hanya berupa deklarasi saja dan tidak memiliki isi.

```
<?php
abstract class User
{
    // abstract method memiliki body, ditandai
    // dengan disertai {}
    // ini yang salah
    abstract protected function showName(){
        //
    }

    // ini yang benar
    abstract protected function showGreeting();
}
```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Abstract function User::showName() cannot contain body ... karena *abstract method* tidak boleh memiliki *body*.

5. Semua class turunan harus mengimplementasikan semua abstract method dari parent class

```
<?php
abstract class User
{
    abstract protected function showName();
    abstract protected function showGreeting($greeting);

    // regular method
    public class showBio(){
        retudn "this is a Bio";
    }
}

class Admin extends User
{
    public function showName(){
        return "Bagus";
    }

    // tidak ada showGreeting($greeting)
}
```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Class Admin contains 1 abstract method and must therefore be declared abstract or implement the remaining methods (User::showGreeting) ... karena tidak semua abstract method dari parent class diimplementasikan, yaitu showGreeting(\$greeting) tidak ada dalam class Admin.

Ingat ya, hanya abstract method, sedangkan untuk regular method tidak harus diturunkan.

6. Semua method turunan dari abstract method harus didefinisikan dengan tingkat visibilitas yang sama atau lebih rendah

```
<?php
abstract class User
{
    // public
    abstract public function showName();
}

class Admin extends User
{
    // protected
    protected function showName(){
        return "Bagus";
    }
}
```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Access level to Admin::showName() must be public (as in class User) ... karena showName() dalam child class memiliki akses level (tingkat visibilitas) lebih tinggi dari pada showName() yang berada dalam parent class. Urutan tingkatan akses level dari tinggi ke rendah adalah private → protected → public.

7. Abstract class boleh memiliki property dan method regular

```
<?php
// abstract class
abstract class User
{
    // regular property
    protected $address = 'Semarang';

    // abstract method
    abstract protected function showName();
    abstract public function showGreeting($greeting);

    // regular method
    public function showBio(){
```

```

        return "Hi, my name is " . $this->showName() . " from " . $this->address;
    }
}

```

8. Abstract class boleh memiliki static method

```

<?php
// abstract class
abstract class User
{
    // abstract method
    abstract protected function showName();

    // static method
    public static function showHi(){
        return "Hi, this is static method";
    }
}

// panggil static method dari abstract class
echo User::showHi();

```

9. Semua method turunan dari abstract method harus mengikuti signature

Misal dalam signature disertai required argument maka method dalam child class harus memiliki required argument tersebut, contoh:

```

<?php
abstract class User
{
    abstract protected function showName();

    // memiliki argumen $greeting
    abstract public function showGreeting($greeting);
}

class Admin extends User
{
    public function showName(){
        return "Bagus";
    }

    // tidak memiliki argumen $greeting
    public function showGreeting(){
        return "My name is " . $this->showName();
    }
}

```

Jika kode program di atas tetap dijalankan maka akan keluar PHP Fatal error: Declaration of Admin::showGreeting() must be compatible with User::showGreeting(\$greeting) ... karena showGreeting() dalam child class tidak memiliki argument sebagaimana showGreeting() dalam parent class.

Tetapi method dalam child class boleh memberikan opsional argument yang tidak ada dalam signature, contoh:

```
<?php
abstract class User
{
    abstract protected function showName();

    // memiliki required argument: $greeting
    abstract public function showGreeting($greeting);
}

class Admin extends User
{
    public function showName(){
        return "Bagus";
    }

    // memiliki required argument: $greeting
    // dan opsional argument: $address
    public function showGreeting($greeting, $address = 'Banjar'){
        return $greeting . ", my name is " . $this->showName() . " from " . $address;
    }
}

$class = new Admin;

// output: Good morning, my name is Bagus from Banjar
echo $class->showGreeting("Good morning");
```

Demikian aturan-aturan dalam *abstract class* dan *abstract method*.

2. Contoh Penggunaan :

a. Membuat Abstract Class dan Method

```
// diawali keyword abstract
abstract class parentClass
{
    // diawali keyword abstract
    abstract protected function namaMethod();
}
```

```
class childClass extends parentClass {
```

```
    public function namaMethodparentClass() {
        logika
    }
}
```

```
$objek1 = new namachildClass();
```

```
$objek1 = namamethodchildClass();
```


3. PRAKTIK

Membuat Abstract Method Dan Class | Contoh 1

1. Setelah file .php buatlah class dan methodnya dengan awalan Abstract

```
<?php

2 references | 2 implementations
abstract class Vehicle {
    2 references | 2 overrides
    abstract public function go();
    2 references | 2 overrides
    abstract public function stop();
}
```

2. Kemudian buatlah childClass dari Parent class Abstract yang sudah dibuat

```
class Car extends Vehicle {
    2 references | 0 overrides | prototype
    public function go() {
        echo "You drive the car <br>";
    }

    2 references | 0 overrides | prototype
    public function stop() {
        echo "This car is stopped <br><br>";
    }
} <- #8-16 class Car extends Vehicle

1 reference | 0 implementations
class Motorcycle extends Vehicle {
    2 references | 0 overrides | prototype
    public function go() {
        echo "You ride the motorcycle <br>";
    }

    2 references | 0 overrides | prototype
    public function stop() {
        echo "This motorcycle is stopped<br><br>";
    }
} <- #18-26 class Motorcycle extends Vehicle
```

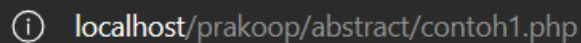
3. Kemudian setelah class dan methodnya dibuat, buatlah instance/objek untuk memanggil childClass dan setelah itu dari objek tersebut dipanggil methodnya

```
// Instantiate the Car and Motorcycle objects
$car = new Car();
$motorcycle = new Motorcycle();

// Call the go and stop methods for both objects
$car->go();
$car->stop();

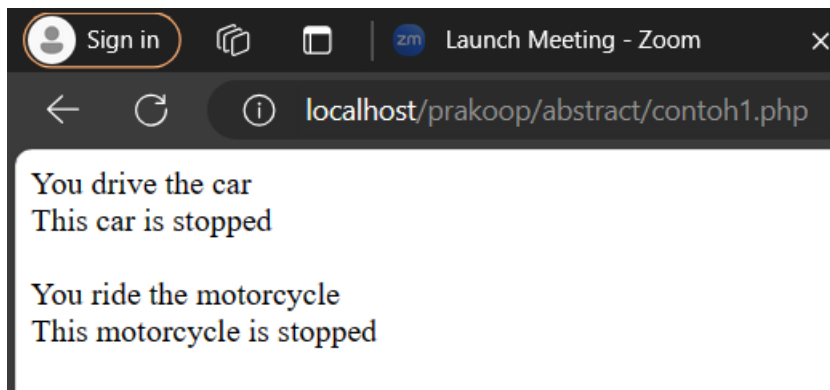
$motorcycle->go();
$motorcycle->stop();
?> The closing ?> tag should be omitted from files c
```

4. Kemudian buka browser dan ketiklah url:
`http:localhost/namafolder(jika langsung htdocs bisa skip)/namafile.php`



localhost/prakoop/abstract/contoh1.php

5. Ketika url tersebut dibuka maka Hasil Akhir seperti ini :



Membuat Abstract Method Dan Class | Contoh 2

1. Setelah file .php buatlah class dan methodnya dengan awalan Abstract

```
<?php
3 references | 3 implementations
abstract class User
{
    5 references | 3 overrides
    abstract protected function showName();

    // memiliki required argument: $greeting
    3 references | 3 overrides
    ⚡ abstract public function showGreeting($greeting);

    // regular method
    3 references | 1 override
    public function showBio(){
        echo "This is my Bio <br/>";
        echo "My name is " . $this->showName();
    }
} <- #3-14 abstract class User
```

2. Kemudian buatlah childClass dari Parent class Abstract yang sudah dibuat

```
1 reference | 0 implementations
class Admin extends User
{
    5 references | 0 overrides | prototype
    public function showName(){
        return "Bagus Admin";
    }

    3 references | 0 overrides | prototype
    public function showGreeting($greeting, $address = 'Banjar'){
        return $greeting . ", my name is " . $this->showName() . " from " . $address;
    }
} <- #17-25 class Admin extends User
```

```

1 reference | 0 implementations
class Editor extends User
{
    5 references | 0 overrides | prototype
    public function showName(){
        |     return "Andre Editor";
    }

    3 references | 0 overrides | prototype
    public function showGreeting($greeting, $address = 'Banjar'){
        |     return $greeting . ", my name is " . $this->showName() . " from " . $address;
    }

    // bukan berasal dari abstract method
    3 references | 0 overrides | prototype
    public function showBio(){
        |     echo "This is Bio from " . $this->showName();
    }
} <- #28-41 class Editor extends User

```

```

1 reference | 0 implementations
class Reporter extends User
{
    5 references | 0 overrides | prototype
    public function showName(){
        |     return "Bambang Reporter";
    }

    3 references | 0 overrides | prototype
    public function showGreeting($greeting, $address = 'Banjar'){
        |     return $greeting . ", my name is " . $this->showName() . " from " . $address;
    }
} <- #44-52 class Reporter extends User

```

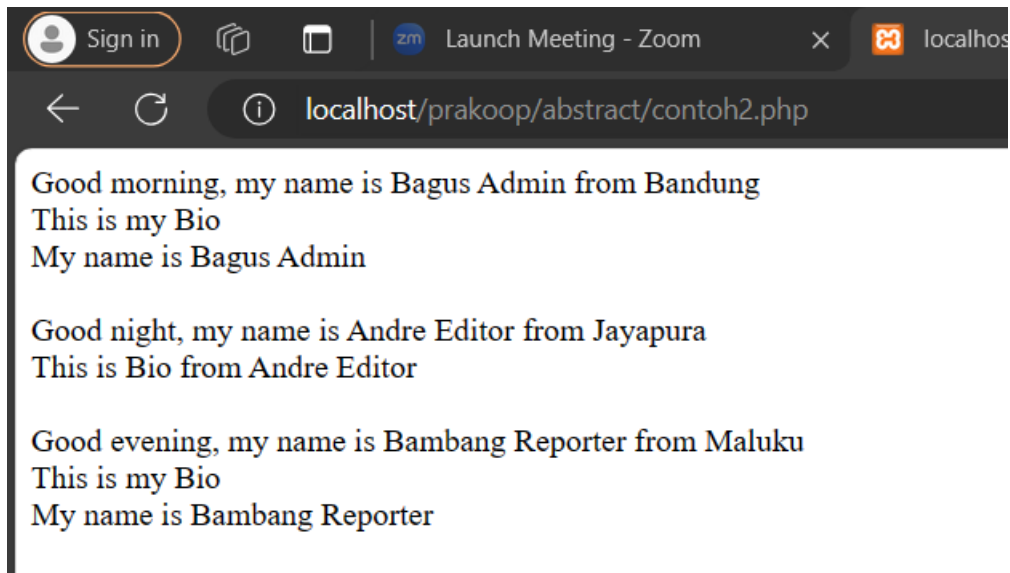
3. Kemudian setelah class dan methodnya dibuat, buatlah instance/objek untuk memanggil childClass dan setelah itu dari objek tersebut dipanggil methodnya

```
$admin = new Admin;  
$editor = new Editor;  
$reporter = new Reporter;  
  
echo $admin->showGreeting("Good morning", "Bandung") . "<br/>";  
echo $admin->showBio() . "<br/><br/>";  
  
echo $editor->showGreeting("Good night", "Jayapura") . "<br/>";  
echo $editor->showBio() . "<br/><br/>";  
  
echo $reporter->showGreeting("Good evening", "Maluku") . "<br/>";  
echo $reporter->showBio();
```

4. Kemudian buka browser dan ketiklah url:
[http://localhost/namafolder\(jika langsung htdocs bisa skip\)/namafile.php](http://localhost/namafolder(jika langsung htdocs bisa skip)/namafile.php)

 localhost/prakoop/abstract/contoh2.php

5. Ketika url tersebut dibuka maka Hasil Akhir seperti ini :



Membuat Abstract Method Dan Class | Contoh 3

1. Setelah file .php buatlah class dan methodnya dengan awalan Abstract

```
<?php

3 references | 3 implementations
abstract class MetodePembayaran {
    3 references | 3 overrides
    abstract public function prosesPembayaran($jumlah);
    3 references | 3 overrides
    abstract public function pengembalian($jumlah);
}
```

2. Kemudian buatlah childClass dari Parent class Abstract yang sudah dibuat

```
class KartuKredit extends MetodePembayaran {
    3 references | 0 overrides | prototype
    public function prosesPembayaran($jumlah) {
        echo "Memproses pembayaran kartu kredit sebesar Rp" . $jumlah . "<br> ";
    }

    3 references | 0 overrides | prototype
    public function pengembalian($jumlah) {
        echo "Mengembalikan Rp" . $jumlah . " ke kartu kredit<br>";
    }
} <- #8-16 class KartuKredit extends MetodePembayaran

1 reference | 0 implementations
class PayPal extends MetodePembayaran {
    3 references | 0 overrides | prototype
    public function prosesPembayaran($jumlah) {
        echo "Memproses pembayaran PayPal sebesar Rp" . $jumlah . "<br>";
    }

    3 references | 0 overrides | prototype
    public function pengembalian($jumlah) {
        echo "Mengembalikan Rp" . $jumlah . " melalui PayPal<br>";
    }
} <- #18-26 class PayPal extends MetodePembayaran
```

```

1 reference | 0 implementations
class TransferBank extends MetodePembayaran {
  3 references | 0 overrides | prototype
  public function prosesPembayaran($jumlah) {
    |   echo "Memproses transfer bank sebesar Rp" . $jumlah . "<br>";
    | }
  }

  3 references | 0 overrides | prototype
  public function pengembalian($jumlah) {
    |   echo "Mengembalikan Rp" . $jumlah . " melalui transfer bank<br>";
    | }
  }
} <- #28-36 class TransferBank extends MetodePembayaran

```

3. Kemudian setelah class dan methodnya dibuat, buatlah instance/objek untuk memanggil childClass dan setelah itu dari objek tersebut dipanggil methodnya

```

// Membuat objek metode pembayaran
$kartuKredit = new KartuKredit();
$payPal = new PayPal();
$transferBank = new TransferBank();

// Memproses pembayaran
$kartuKredit->prosesPembayaran(100000);
$payPal->prosesPembayaran(200000);
$transferBank->prosesPembayaran(300000);

echo "<br>";

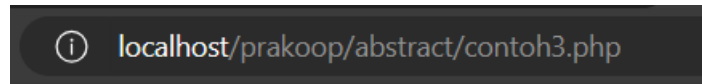
// Memproses pengembalian dana
$kartuKredit->pengembalian(50000);
$payPal->pengembalian(75000);
$transferBank->pengembalian(125000);
?> The closing ?> tag should be omitted from

```

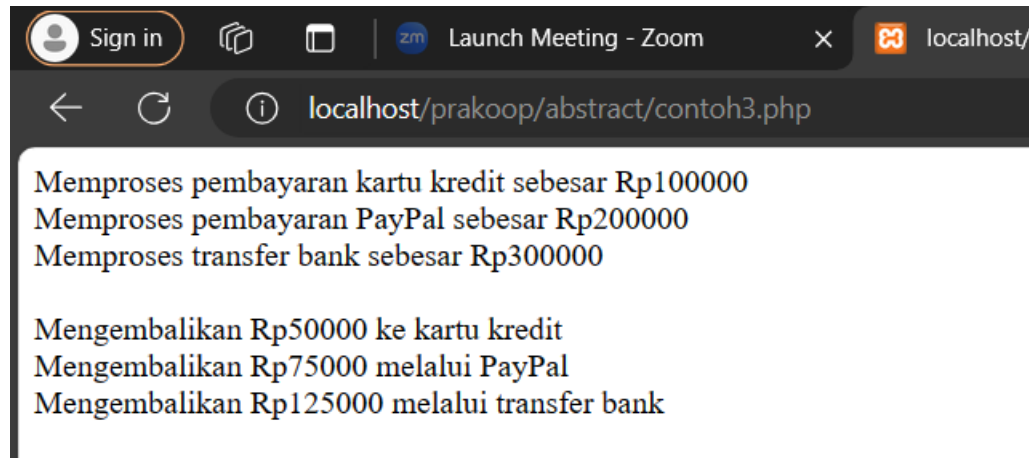
4. Kemudian buka browser dan ketiklah

url:

`http:localhost/namafolder(jika langsung htdocs bisa skip)/namafile.php`



5. Ketika url tersebut dibuka maka Hasil Akhir seperti ini :



3. PERCOBAAN MANDIRI

Lakukanlah Percobaan Praktik Dimulai dari atas, setelah dilakukan buatlah variasi dengan tema yang berbeda dari contoh diatas.

4. TUGAS TAKE HOME

PENGUMPULAN TUGAS DAN DEADLINE

- a. DI Google Drive
- b. Deadline = 21 Mei 2024 | 23.59 WIB

Soal |

1. Buatlah code berdasarkan dari kegiatan sehari hari, yang Dimana harus menggunakan Abstract Class dan memiliki parent class minimal 4
2. Buatlah Penjelasan dari masing masing Outputnya
3. Dikumpulkan file .php dan laporannya di google drive

DAFTAR PUSTAKA

- <https://elektro.um.ac.id/wp-content/uploads/2016/04/Modul-6-Abstract-Class-dan-Interface.pdf>
- https://www.w3schools.com/php/php_oop_classes_abstract.asp
- <https://khoerodin.id/id/abstract-class-dan-abstract-method-dalam-oop-php>
- https://www.youtube.com/watch?v=P002GacKKZA&list=PLFIM0718LjIWvxxll-6wLXrC_16h_Bl_p&index=13&ab_channel=WebProgrammingUNPAS
- https://www.youtube.com/watch?v=Jaa1afOUfS8&list=PLFIM0718LjIWvxxll-6wLXrC_16h_Bl_p&index=14&ab_channel=WebProgrammingUNPAS