



Tutorial Belajar OOP PHP Part 1: Pengertian Pemrograman Berbasis Objek

28 Sep 14 | Andre | Tutorial PHP | 62 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

[eBook Pascal Uncover](#)

[eBook HTML Uncover](#)

[eBook CSS Uncover](#)

[eBook PHP Uncover](#)

[eBook MySQL Uncover](#)

[eBook JavaScript Uncover](#)

[eBook Bootstrap Uncover](#)

[eBook OOP PHP Uncover](#)

[eBook Laravel Uncover](#)

Jika anda telah biasa membuat program secara *prosedural*, yakni menulis program-program dari baris pertama sampai dengan baris terakhir secara berurutan, konsep *pemrograman berbasis objek* mungkin sedikit susah dipahami. Dalam tutorial pertama tentang OOP PHP ini, kita akan membahas pengertian [Pengertian Pemrograman Berbasis Objek](#) dalam PHP.

Pengertian Pemrograman Berbasis Objek

Pemrograman Berbasis Objek atau Object Oriented Programming (*OOP*) adalah sebuah tata cara pembuatan program (*programming paradigm*) dengan menggunakan konsep “*objek*” yang memiliki data (*atribut* yang menjelaskan tentang objek) dan prosedur (*function*) yang dikenal dengan *method*. (http://en.wikipedia.org/wiki/Object-oriented_programming)

Dalam pengertian sederhananya, *OOP* adalah konsep pembuatan program dengan memecah permasalahan program dengan menggunakan *objek*. *Objek* dapat diumpamakan dengan ‘fungsi khusus’ yang bisa berdiri sendiri. Untuk membuat sebuah aplikasi, berbagai objek akan saling bertukar data untuk mencapai hasil akhir.

Berbeda dengan konsep fungsi atau ‘*function*’ di dalam pemrograman, sebuah **objek** bisa memiliki data dan *function* tersendiri. Setiap objek ditujukan untuk mengerjakan sebuah tugas, dan menghasilkan nilai akhir untuk selanjutnya dapat ditampilkan atau digunakan oleh **objek** lain.

Fungsi Pemrograman Berbasis Objek dalam PHP

PHP bukan bahasa pemrograman yang '*murni*' berbasis objek seperti **Java**. Bahkan, konsep **OOP** dalam PHP baru hadir dalam PHP versi 4, dan disempurnakan oleh PHP versi 5. Dengan kata lain, **OOP** di PHP merupakan '*fitur tambahan*'. Anda bisa membuat situs web dengan PHP tanpa menggunakan objek sama sekali.

Dalam studi *pemrograman*, pembuatan program dalam PHP tanpa menggunakan objek disebut juga dengan **pemrograman prosedural** atau **pemrograman fungsional**. Dikenal dengan *pemrograman prosedural*, karena kita memecah kode program menjadi bagian-bagian atau fungsi-fungsi kecil, kemudian menyatukannya untuk menghasilkan nilai akhir.

Dengan membuat program secara *prosedural*, aplikasi bisa dibuat dengan cepat dan mudah dipelajari jika dibandingkan dengan *pemrograman berbasis objek* (bagi anda yang pernah mempelajari **Java**, tentu telah '*melewati*' hal ini :)). Keuntungan *pemrograman berbasis objek* baru terasa ketika program tersebut telah '*besar*' atau kita bekerja dengan tim untuk membagi tugas. Konsep '*objek*' untuk memisahkan program menjadi bagian-bagian yang berdiri sendiri akan memudahkan dalam membuat program.

Saya tidak akan panjang lebar menjelaskan tentang keuntungan atau kerugian menggunakan OOP. Sebagai programmer web, **OOP** adalah salah satu *makanan wajib*. Pembuatan website modern saat ini akan lebih mudah jika menggunakan template kode program yang dikenal dengan **framework**. Daripada kita membuat situs mulai dari awal, menggunakan **framework** akan mempercepat proses kerja. Dan, framework PHP hampir semuanya dibuat menggunakan OOP.

Dalam tutorial pertama tentang OOP PHP ini, kita telah mempelajari tentang **pengertian pemrograman berbasis objek**. Dalam tutorial selanjutnya, kita akan mempelajari tentang **pengertian class, object, property dan method** dalam PHP.

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoloading* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

*** Artikel Terkait ***

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array



[Home](#) | [Tutorial PHP](#) | [Tutorial Belajar OOP PHP Part 2: Pengertian Class, Object, Property dan Method](#)

Tutorial Belajar OOP PHP Part 2: Pengertian Class, Object, Property dan Method

29 Sep 14 | Andre | [Tutorial PHP](#) | 81 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Pemrograman berbasis objek tidak hanya berisi ‘object’. Dalam tutorial belajar OOP PHP kali ini kita akan membahas tentang [pengertian class, object, property dan method](#). Keempat ‘keyword’ inilah yang menjadi pondasi dasar dari *Pemrograman Berbasis Objek*. Selain pengertian, kita juga akan mempelajari cara penulisannya dengan PHP.



Untuk memudahkan pemahaman dan agar sejalan dengan istilah aslinya, saya tetap menggunakan istilah *bahasa inggris* untuk kata kunci PHP, seperti: **class, object, property** dan **method**.

Pengertian Class dalam Pemrograman Berbasis Objek

Class adalah ‘cetak biru’ atau ‘blueprint’ dari **object**. Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nantinya adalah hasil cetakan dari class, yakni **object**.

Sebagai analogi, **class** bisa diibaratkan dengan *laptop* atau *notebook*. Kita tahu bahwa *laptop* memiliki ciri-ciri seperti *merk*, memiliki *keyboard*, memiliki *processor*, dan beberapa ciri khas lain yang menyatakan sebuah benda tersebut adalah *laptop*. Selain memiliki ciri-ciri, sebuah laptop juga bisa dikenakan tindakan, seperti: *menghidupkan laptop* atau *mematikan laptop*.



Class dalam analogi ini adalah gambaran umum tentang sebuah benda. Di dalam pemrograman nantinya, contoh class seperti: *koneksi_database* dan *profile_user*.

Di dalam PHP, penulisan **class** diawali dengan *keyword class*, kemudian diikuti dengan *nama dari class*. Aturan penulisan nama **class** sama seperti aturan penulisan *variabel* dalam PHP, yakni diawali dengan huruf atau *underscore* untuk karakter pertama, kemudian boleh diikuti dengan huruf, underscore atau angka untuk karakter kedua dan selanjutnya. Isi dari **class** berada dalam tanda kurung kurawal.

Berikut adalah contoh penulisan **class** dalam PHP:

```
1 <?php
2 class laptop {
3     // isi dari class laptop...
4 }
5 ?>
```

Pengertian Property dalam Pemrograman Berbasis Objek

Property (atau disebut juga dengan *atribut*) adalah data yang terdapat dalam sebuah **class**. Melanjutkan analogi tentang *laptop*, **property** dari laptop bisa berupa *merk*, *warna*, *jenis processor*, *ukuran layar*, dan lain-lain.

Jika anda sudah terbiasa dengan program PHP, **property** ini sebenarnya hanyalah *variabel* yang terletak di dalam **class**. Seluruh aturan dan tipe data yang biasa diinput ke dalam *variabel*, bisa juga diinput kedalam **property**. Aturan tata cara penamaan **property** sama dengan aturan penamaan *variabel*.

Berikut adalah contoh penulisan **class** dengan penambahan **property**:

```
1 <?php
2 class laptop {
3     var $pemilik;
4     var $merk;
5     var $ukuran_layar;
6     // lanjutan isi dari class laptop...
7 }
8 ?>
```

Dari contoh di atas, *\$merk*, *\$ukuran_layar* dan *\$jenis_processor* adalah **property** dari **class laptop**. Seperti yang kita lihat, penulisan property di dalam PHP sama dengan cara penulisan *variabel*, yakni menggunakan tanda *dollar (\$)*. Sebuah **class** tidak harus memiliki **property**.

Pengertian Method dalam Pemrograman Berbasis Objek

Method adalah tindakan yang bisa dilakukan di dalam **class**. Jika menggunakan analogi **class laptop** kita, maka contoh method adalah: *menghidupkan laptop*, *mematikan laptop*, *mengganti cover laptop*, dan berbagai tindakan lain.

Method pada dasarnya adalah **function** yang berada di dalam **class**. Seluruh fungsi dan sifat function bisa diterapkan ke dalam method, seperti argumen/parameter, mengembalikan nilai (dengan keyword *return*), dan lain-lain.

Berikut adalah contoh penulisan **class** dengan penambahan **method**:

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array

```

1 <?php
2 class laptop {
3     function hidupkan_laptop() {
4         //... isi dari method hidupkan_laptop
5     }
6
7     function matikan_laptop() {
8         //... isi dari method matikan_laptop
9     }
10
11     ... //isi dari class laptop
12 }
13 ?>

```

Dari contoh di atas, `function hidupkan_laptop()` dan `function matikan_laptop()` adalah *method* dari class `laptop`. Seperti yang kita lihat, bahwa penulisan method di dalam PHP sama dengan cara penulisan `function`. Sebuah class tidak harus memiliki method.

Pengertian Object dalam Pemrograman Berbasis Objek

Object atau **Objek** adalah *hasil cetak* dari `class`, atau hasil ‘*konkrit*’ dari `class`. Jika menggunakan analogi `class laptop`, maka objek dari `class laptop` bisa berupa: `laptop_andi`, `laptop_anto`, `laptop_duniaIlkom`, dan lain-lain. Objek dari `class laptop` akan memiliki seluruh ciri-ciri `laptop`, yaitu *property* dan *method*-nya.

Proses ‘*mencetak*’ objek dari `class` ini disebut dengan ‘*instansiasi*’ (atau *instantiation* dalam bahasa *inggris*). Pada PHP, proses *instansiasi* dilakukan dengan menggunakan keyword ‘`new`’. Hasil cetakan `class` akan disimpan dalam *variabel* untuk selanjutnya digunakan dalam proses program.

Sebagai contoh, berikut adalah cara membuat **objek** `laptop_andi` dan `laptop_anto` yang dibuat dari `class laptop`:

```

1 <?php
2 class laptop {
3     //... isi dari class laptop
4 }
5
6 $laptop_andi = new laptop();
7 $laptop_anto = new laptop();
8 ?>

```

Dari contoh di atas, `$laptop_andi` dan `$laptop_anto` merupakan **objek** dari `class laptop`. Kedua objek ini akan memiliki seluruh *property* dan *method* yang telah dirancang dari `class laptop`.

Dalam tutorial kali ini kita telah membahas tentang [pengertian dan cara penggunaan class, property, method, dan object](#). Dalam tutorial OOP PHP berikutnya kita akan membahas lebih dalam tentang [cara membuat dan mengakses objek dalam PHP](#).

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾



Tutorial Belajar OOP PHP Part 3: Cara Membuat dan Mengakses Objek dalam PHP

29 Sep 14 | Andre | Tutorial PHP | 46 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Melanjutkan tutorial belajar *Objek Oriented Programming (OOP) PHP*, jika sebelumnya kita telah mempelajari [Pengertian Class, Object, Property dan Method](#), kali ini kita akan membahas [Cara Membuat dan Mengakses Objek dalam PHP](#).

Cara Membuat Objek dalam PHP

Istilah **Objek** dalam *Objek Oriented Programming (OOP)*, sebenarnya terdiri dari **class**, **property**, **method** dan **object**. Keempat istilah ini telah kita pelajari dalam tutorial sebelumnya. Merangkum apa yang telah kita pelajari, berikut adalah contoh cara pembuatan objek di dalam PHP:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     var $pemilik;
7     var $merk;
8     var $ukuran_layar;
9
10    // buat method untuk class laptop
11    function hidupkan_laptop() {
12        return "Hidupkan Laptop";
13    }
14    function matikan_laptop() {
15        return "Matikan Laptop";
16    }
17 }
18
19 // buat objek dari class laptop (instansiasi)
20 $laptop_anto = new laptop();
21 ?>
```

Dalam kode di atas, kita telah membuat sebuah **class** dengan nama ***laptop***, lengkap dengan **property** dan **method**-nya. Kemudian saya membuat 1 buah objek dari class ***laptop*** dengan nama **\$laptop_ant** pada baris terakhir.

Walaupun dalam kode di atas kita telah membuat **objek**, **objek** tersebut belum menampilkan apa-apa, karena **class** ***laptop*** belum berisi data apapun. Kita akan segera mempelajari cara mengakses ‘isi’ dari **class** dalam PHP.



Jika anda bertanya kenapa saya menggunakan contoh **class laptop** yang mungkin ‘jauh’ dari istilah *Web*, jawabannya adalah agar menyederhanakan penjelasan. *Objek Oriented Programming* memiliki aturan dan alur program yang cukup rumit sehingga menggunakan contoh sederhana ini bisa mempermudah dalam memahami konsep OOP.

Dalam buku-buku yang membahas **OOP**, jenis class yang ‘aneh’ ini juga sering digunakan sebagai contoh. Misalnya **class mobil**, **class sapi**, ataupun **class orang_utang**. Sekali lagi, ini hanya untuk menyederhanakan pemahaman.

Cara Mengakses Objek dalam PHP

Cara *mengakses objek* yang saya maksud sebenarnya adalah cara untuk mengakses ‘isi’ dari sebuah **objek**, yakni **property** dan **method**-nya. Agar lebih mudah dipahami, berikut adalah revisi contoh **class laptop** sebelumnya:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     var $pemilik;
7     var $merk;
8     var $ukuran_layar;
9
10    // buat method untuk class laptop
11    function hidupkan_laptop() {
12        return "Hidupkan Laptop";
13    }
14    function matikan_laptop() {
15        return "Matikan Laptop";
16    }
17 }
18
19 // buat objek dari class laptop (instansiasi)
20 $laptop_ant = new laptop();
21
22 // set property
23 $laptop_ant->pemilik="Anto";
24 $laptop_ant->merk="Asus";
25 $laptop_ant->ukuran_layar="15 inchi";
26
27 // tampilkan property
28 echo $laptop_ant->pemilik;
29 echo "<br />";
30 echo $laptop_ant->merk;
31 echo "<br />";
32 echo $laptop_ant->ukuran_layar;
33 echo "<br />";
34
35 // tampilkan method
36 echo $laptop_ant->hidupkan_laptop();
37 echo "<br />";
38 echo $laptop_ant->matikan_laptop();
39 ?>
```

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array

Jika anda menjalankan kode program di atas, berikut adalah hasil yang didapat:

```
1 | Anto
2 | Asus
3 | 15 inchi
4 | Hidupkan Laptop
5 | Matikan Laptop
```

Mari kita bahas kode program di atas:

```
1 | class laptop {
```

Baris awal contoh kode program kita adalah pendefinisan sebuah **class** dengan nama *laptop*. Kurung kurawal menandakan awal dari *class*.

```
1 | var $pemilik;
2 | var $merk;
3 | var $ukuran_layar;
```

Selanjutnya, tiga baris di atas merupakan pendefinisan **variabel class**, atau dikenal dengan **property**. *Property* tidak lain hanya variabel biasa yang berada di dalam *class*. Keyword **var** digunakan untuk deklarasi **variabel** di dalam *class*.



Jika anda pernah mempelajari OOP PHP, mungkin akan langsung komplain tentang penggunaan keyword **var** untuk pendefinisan *property*. Saya menggunakan keyword **var** hanya untuk menyederhanakan penjelasan. Konsep *enkapsulasi* (*public*, *protected* dan *private*) akan dibahas mendalam dalam tutorial berikutnya. Kita akan mengoreksi cara pendefenisian ini nantinya.

```
1 | function hidupkan_laptop() {
2 |     return "Hidupkan Laptop";
3 |
4 | }
5 | function matikan_laptop() {
6 |     return "Matikan Laptop";
7 | }
```

2 buah fungsi di atas adalah *method* dari *class*. Jika anda telah mengenal cara pembuatan **function**, kedua contoh ini hanyalah *fungsi* biasa yang akan mengembalikan nilai berupa *string*. Pembahasan tentang cara penggunaan *function* di dalam PHP pernah kita bahas pada [Tutorial PHP Cara Penulisan dan Pembuatan Fungsi PHP](#).

```
1 | $laptop_ant = new laptop();
```

Baris ini merupakan perintah untuk pembuatan **objek** dari *class laptop* (dikenal dengan proses *instansiasi*). Variabel **\$laptop_ant** saat ini berisi **objek** dari *class laptop*. Kita akan mengakses *property* dan *method* *class laptop* melalui *objek* **\$laptop_ant** ini.

```
1 | $laptop_ant->pemilik="Anto";
2 | $laptop_ant->merk="Asus";
3 | $laptop_ant->ukuran_layar="15 inchi";
```

[18. Pengertian Operand](#)

[19. Fungsi var_dump\(\)](#)

[20. Operator Aritmatika](#)

[21. Operator String](#)

[22. Operator Logika](#)

[23. Perbandingan](#)

[24. Operator Increment](#)

[25. Assignment PHP](#)

[26. Operator Bitwise](#)

[27. Operator Assigment](#)

[28. Type Casting](#)

[29. Struktur Logika IF](#)

[30. Struktur ELSE](#)

[31. Logika ELSE-IF](#)

[32. Struktur Switch](#)

[33. Perulangan For](#)

[34. Perulangan While](#)

[35. Do-While](#)

[36. Perintah Break](#)

[37. Perintah Continue](#)

[38. Perulangan Foreach](#)

[39. Pengertian Function](#)

[40. Penulisan Function](#)

[41. Variabel Scope](#)

[42. Argumen Function](#)

[43. Default Parameter](#)

[44. Variable Parameter](#)

[Tutorial PHP Lanjutan: Form](#)

[Tutorial PHP Lanjutan: Form](#)

[Tutorial PHP Lanjutan: PHP - MySQL](#)

[Tutorial PHP Lanjutan: PHP - MySQL](#)

[Tutorial PHP Lanjutan: OOP PHP](#)

[Tutorial PHP Lanjutan: OOP PHP](#)

Berlangganan Artikel DuniaIlkom

3 baris di atas adalah cara untuk men-set nilai ke dalam *property* dari **objek** `$laptop_anto`. Perhatikan bahwa kita menggunakan *tanda panah* (`->`) untuk mengakses *property* dari **objek**. Tanda panah ini adalah operator khusus objek yang dikenal dengan istilah '*Object Operator*'. Penulisan nama *property* juga dilakukan tanpa menggunakan tanda \$, sehingga *property* `$pemilik`, diakses dengan: `$laptop_anto->pemilik`.

```
1 echo $laptop_anto->pemilik;
2 echo "<br />";
3 echo $laptop_anto->merk;
4 echo "<br />";
5 echo $laptop_anto->ukuran_layar;
6 echo "<br />";
```

Kode program di atas digunakan untuk menampilkan nilai *property* dari objek `$laptop_anto` yang sebelumnya telah di-set nilainya. Sama seperti pada saat men-set nilai *property*, kita juga menggunakan *tanda panah* (`->`), kemudian diikuti nama *property* tanpa tanda \$.

Perintah `echo` ditambahkan agar PHP menampilkan nilai *property* ke dalam web browser. Agar tampilan di web browser lebih rapi, saya menambahkan `
` diantara hasil tampilan *property*.

```
1 echo $laptop_anto->hidupkan_laptop();
2 echo "<br />";
3 echo $laptop_anto->matikan_laptop();
```

Kode terakhir dari contoh kita kali ini adalah cara pemanggilan **method** dari *objek* `$laptop_anto`. Cara pengaksesannya sama dengan cara mengakses *property*, namun karena method adalah *fungsi*, kita menambahkan tanda kurung di akhir pemanggilan fungsi.

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

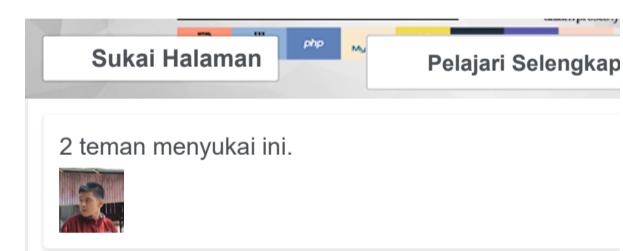
Join 3,082 other subscribers

Subscribe

Objek Sebagai Entitas Terpisah

Dalam contoh kode di atas, saya hanya menggunakan 1 buah **objek** yang berasal dari *class laptop*. Namun pada dasarnya sebuah *class* bisa digunakan untuk membuat berapapun banyak objek. Setiap objek merupakan bagian terpisah, namun tetap memiliki *property* dan *method* yang berasal dari *class laptop*.

Berikut adalah contoh pembuatan beberapa **objek** dari *class laptop*:



```
1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     var $pemilik;
7
8     // buat method untuk class laptop
9     function hidupkan_laptop() {
10         return "Hidupkan Laptop";
11     }
12 }
13
14 // buat objek dari class laptop (instansiasi)
15 $laptop_anto = new laptop();
16 $laptop_andi = new laptop();
17 $laptop_dina = new laptop();
18
19 // set property
20 $laptop_anto->pemilik="Anto";
21 $laptop_andi->pemilik="Andi";
22 $laptop_dina->pemilik="Dina";
23
24 // tampilkan property
25 echo $laptop_anto->pemilik; // Anto
26 echo "<br />";
27 echo $laptop_andi->pemilik; // Andi
28 echo "<br />";
29 echo $laptop_dina->pemilik; // Dina
30 echo "<br />";
31 ?>
```

Class laptop di atas saya sederhanakan agar lebih singkat. Setelah pembuatan class, saya kemudian membuat 3 buah objek dari *class laptop*, yakni *\$laptop_anto*, *\$laptop_andi* dan *\$laptop_dina*. Ketiga objek ini memiliki struktur yang sama (sama-sama berasal dari *class laptop*), namun memiliki isi data yang berbeda-beda.

Agar lebih paham, silahkan anda mencoba menambahkan beberapa *property* dan *method* untuk *class laptop* di atas.

Dalam tutorial kali ini kita telah membahas tentang [Cara Membuat dan Mengakses Objek dalam PHP](#). Sampai disini semoga anda telah memahami konsep *class*, *property*, *method* dan *objek*. Dalam *tutorial belajar OOP PHP* berikutnya, kita akan membahas tentang [pengertian enkapsulasi dalam pemrograman objek](#).

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoload*ing dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

*** Artikel Terkait ***



Home | Tutorial PHP | Tutorial Belajar OOP PHP Part 4: Pengertian Enkapsulasi Objek (Public, Protected dan Private)

Tutorial Belajar OOP PHP Part 4: Pengertian Enkapsulasi Objek (Public, Protected dan Private)

29 Sep 14 | Andre | Tutorial PHP | 29 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Setelah memahami [cara membuat dan mengakses objek dalam PHP](#), dalam tutorial kali ini kita akan membahas salah satu aspek terpenting dalam *Pemrograman Berbasis Objek (OOP)*, yakni **Enkapsulasi** (bahasa Inggris: *Encapsulation*). Proses *enkapsulasi* diterapkan dengan menggunakan 3 jenis hak akses: **Public**, **Protected** dan **Private**.

Kita akan mempelajari [Pengertian Enkapsulasi \(Public, Protected dan Private\) dalam PHP](#).

Pengertian Enkapsulasi (Encapsulation)

Enkapsulasi (encapsulation) adalah sebuah metoda untuk mengatur struktur *class* dengan cara menyembunyikan alur kerja dari *class* tersebut.

Struktur class yang dimaksud adalah **property** dan **method**. Dengan *enkapsulasi*, kita bisa membuat pembatasan akses kepada *property* dan *method*, sehingga hanya *property* dan *method* tertentu saja yang bisa diakses dari luar *class*. *Enkapsulasi* juga dikenal dengan istilah '*information hiding*'.

Dengan *enkapsulasi*, kita bisa memilih *property* dan *method* apa saja yang boleh diakses, dan mana yang tidak boleh diakses. Dengan menghalangi kode program lain untuk mengubah *property* tertentu, *class* menjadi lebih terintegrasi, dan menghindari



kesalahan ketika seseorang '*mencoba*' mengubahnya. Programmer yang merancang *class* bisa menyediakan *property* dan *method* khusus yang memang ditujukan untuk diakses dari luar.

Melanjutkan analogi tentang *class laptop*, perusahaan pembuat laptop telah menyediakan '*method*' khusus untuk menghidupkan laptop, yakni dengan cara menekan tombol **on**. Di dalam laptop sendiri, banyak '*method-method*' lain yang akan dijalankan ketika kita menyalakan laptop, contohnya: mengirim sinyal booting ke processor, mengirim data dari processor ke memory, dan mengirim sinyal listrik ke LED di monitor. Akan tetapi, proses ini adalah *method internal* laptop dimana kita tidak perlu memahaminya untuk menghidupkan laptop.

Enkapsulasi Objek: Public, Protected dan Private

Untuk membatasi hak akses kepada *property* dan *method* di dalam sebuah *class*, *Object Oriented Programming* menyediakan 3 kata kunci, yakni **Public**, **Protected** dan **Private**. Kata kunci ini diletakkan sebelum nama *property* atau sebelum nama *method*. Berikut adalah pembahasannya:

Pengertian Hak Akses: Public

Ketika sebuah *property* atau *method* dinyatakan sebagai **public**, maka *seluruh kode program di luar class bisa mengaksesnya, termasuk class turunan*. Berikut adalah contoh penulisan **public property** dan **public method** dalam PHP:

```

1 <?php
2
3 // buat class laptop
4 class laptop {
5
6     // buat public property
7     public $pemilik;
8
9     // buat public method
10    public function hidupkan_laptop() {
11        return "Hidupkan Laptop";
12    }
13
14
15 // buat objek dari class laptop (instansiasi)
16 $laptop_ant = new laptop();
17
18 // set property
19 $laptop_ant->pemilik="Anto";
20
21 // tampilkan property
22 echo $laptop_ant->pemilik; // Anto
23
24 // tampilkan method
25 echo $laptop_ant->hidupkan_laptop(); // "Hidupkan Laptop"
26 ?>
```

Perhatikan penambahan kata **public** sebelum nama *property* dan nama *method*. Kode di atas pada dasarnya sama dengan contoh *class laptop* kita dalam tutorial sebelum ini.

Jika hak akses *property* dan *method* tidak ditulis, maka PHP menganggapnya sebagai **public**.



Dalam contoh sebelumnya, saya membuat *property class laptop* menggunakan kata kunci **var**, seperti berikut ini:

```
var $pemilik;
```

List Tutorial DuniaIlkom

- [Tutorial Terbaru DuniaIlkom](#)
- [Tutorial HTML](#)
- [Tutorial PHP](#)
- [Tutorial MySQL](#)
- [Tutorial CSS](#)
- [Tutorial JavaScript](#)
- [Tutorial jQuery](#)
- [Tutorial WordPress](#)
- [Tutorial Pascal](#)
- [Tutorial C](#)
- [Tutorial Python](#)
- [Tutorial Java](#)
- [Tutorial Laravel](#)
- [Membuat Web Online](#)
- [Review Jurusan Kuliah](#)
- [Blog DuniaIlkom](#)

Tutorial Dasar PHP

- [Tutorial PHP Dasar](#)
- [1. Pengertian PHP](#)
- [2. Sejarah PHP](#)
- [3. Menginstall XAMPP](#)
- [4. Menjalankan Apache](#)
- [5. Menjalankan File PHP](#)
- [6. Cara Kerja WebServer](#)
- [7. Input PHP ke HTML](#)
- [8. File php.ini](#)
- [9. Dasar Penulisan PHP](#)
- [10. Penulisan Komentar](#)
- [11. Penulisan Variabel](#)
- [12. Penulisan Konstanta](#)
- [13. Tipe Data Integer](#)
- [14. Tipe Data Float](#)
- [15. Tipe Data String](#)
- [16. Tipe Data Boolean](#)
- [17. Tipe Data Array](#)

Kata kunci **var** sebenarnya tidak perlu ditulis, bahkan pada *PHP versi 5.0* sampai dengan *5.1.3*, menggunakan kata **var** di dalam *class* akan menghasilkan *warning*. Untuk *PHP versi 5.1.3* ke atas, menggunakan kata **var** tidak lagi menghasilkan warning (dianggap sebagai **public**).

Agar sejalan dengan konsep *enkapsulasi*, setiap *property* dan *method* dalam *class* harus menggunakan kata kunci seperti **public**, **protected**, atau **private**. Oleh karena itu, membuat *property* dengan *keyword* **var** tidak disarankan lagi .

Pengertian Hak Akses: Protected

Jika sebuah *property* atau *method* dinyatakan sebagai **protected**, berarti *property* atau *method* tersebut *tidak bisa diakses dari luar class, namun bisa diakses oleh class itu sendiri atau turunan class tersebut*.

Apabila kita mencoba mengakses **protected property** atau **protected method** dari luar *class*, akan menghasilkan **error**, seperti contoh berikut ini:

```

1 <?php
2
3 // buat class laptop
4 class laptop {
5
6     // buat protected property
7     protected $pemilik;
8
9     // buat protected method
10    protected function hidupkan_laptop() {
11        return "Hidupkan Laptop";
12    }
13
14
15 // buat objek dari class laptop (instansiasi)
16 $laptop_ant = new laptop();
17
18 // set protected property akan menghasilkan error
19 $laptop_ant->pemilik="Anto";
20 // Fatal error: Cannot access protected property laptop::$pemilik
21
22 // tampilkan protected property akan menghasilkan error
23 echo $laptop_ant->pemilik;
24 // Fatal error: Cannot access protected property laptop::$pemilik
25
26 // jalankan protected method akan menghasilkan error
27 echo $laptop_ant->hidupkan_laptop();
28 // Fatal error: Call to protected method laptop::hidupkan_laptop()
29 // from context
30 ?>
```

Dalam contoh di atas, pemanggilan *property* **\$pemilik** dan *method* **hidupkan_laptop()** dari luar *class* akan menghasilkan **error**.

Walaupun akses level **protected** tidak bisa diakses dari luar *class*, namun *bisa diakses dari dalam class itu sendiri*, berikut adalah contohnya:

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

```

1 <?php
2
3 // buat class laptop
4 class laptop {
5
6     // buat protected property
7     protected $pemilik="Anto";
8
9     public function akses_pemilik() {
10        return $this->pemilik;
11    }
12    protected function hidupkan_laptop() {
13        return "Hidupkan Laptop";
14    }
15    public function paksa_hidup() {
16        return $this->hidupkan_laptop();
17    }
18 }
19
20 // buat objek dari class laptop (instansiasi)
21 $laptop_ant = new laptop();
22
23 // jalankan method akses_pemilik()
24 echo $laptop_ant->akses_pemilik(); // "Anto"
25
26 // jalankan method paksa_hidup()
27 echo $laptop_ant->paksa_hidup(); // "Hidupkan Laptop"
28 ?>

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Hampir sama dengan contoh kita sebelumnya, *property \$pemilik* di deklarasikan sebagai **protected**, sehingga pengaksesan dari luar class akan menghasilkan *error*. Oleh karena itu, saya membuat sebuah *public method* yang akan menampilkan hasil *property \$pemilik*, yakni method *akses_pemilik()*.

Begitu juga dengan *method hidupkan_laptop()* yang tidak bisa diakses secara langsung. Saya menambahkan method *paksa_hidup()* yang secara internal akan mengakses method *hidupkan_laptop()*.



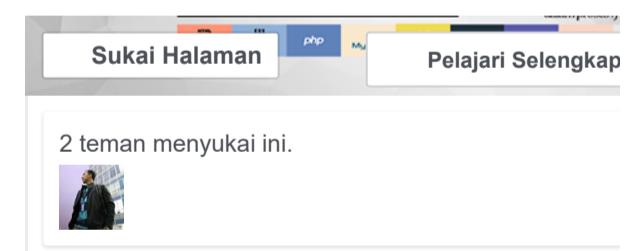
Kata kunci **\$this** merujuk kepada objek saat ini, penjelasan tentang kata kunci **\$this** akan kita bahas dalam tutorial berikutnya. Namun karena satu-satunya cara pengaksesan 'isi' class dari dalam class adalah dengan **\$this**, jika anda tidak memahaminya silahkan lewatkan saja terlebih dahulu.

Selain dari dalam *class* itu sendiri, *property* dan *method* dengan hak akses **protected** juga bisa diakses dari *class turunan* (Kita akan membahas tentang *penurunan class* dalam tutorial lain):

```

1 <?php
2
3 // buat class komputer
4 class komputer{
5
6     // property dengan hak akses protected
7     protected $jenis_processor = "Intel Core i7-4790 3.6Ghz";
8 }
9
10 // buat class laptop
11 class laptop extends komputer{
12     public function tampilkan_processor() {
13         return $this->jenis_processor;
14     }
15 }
16
17 // buat objek dari class laptop (instansiasi)
18 $laptop_baru = new laptop();
19
20 // jalankan method
21 echo $laptop_baru->tampilkan_processor(); // "Intel Core i7-4790 3.
22 ?>

```



Pada kode di atas, walaupun method *\$jenis_processor* di set sebagai **protected** pada *class komputer*, tetapi masih bisa diakses dari *class laptop* yang merupakan turunan dari *class komputer*.

Pengertian Hak Akses: Private

Hak akses terakhir dalam konsep *enkapsulasi* adalah **private**. Jika sebuah *property* atau *method* di-set sebagai **private**, maka satu-satunya yang bisa mengakses adalah *class* itu sendiri. *Class* lain tidak bisa mengaksesnya, termasuk *class turunan*.

Sebagai contoh, berikut adalah hasil yang di dapat jika kita mengakses *property* dan *method* dengan level **private**:

```
1 <?php
2
3 // buat class komputer
4 class komputer {
5
6     // property dengan hak akses protected
7     private $jenis_processor = "Intel Core i7-4790 3.6Ghz";
8
9     public function tampilan_processor() {
10         return $this->jenis_processor;
11     }
12 }
13
14 // buat class laptop
15 class laptop extends komputer{
16
17     public function tampilan_processor() {
18         return $this->jenis_processor;
19     }
20 }
21
22 // buat objek dari class laptop (instansiasi)
23 $komputer_baru = new komputer();
24 $laptop_baru = new laptop();
25
26 // jalankan method dari class komputer
27 echo $komputer_baru->tampilan_processor(); // "Intel Core i7-4790
28
29 // jalankan method dari class laptop (error)
30 echo $laptop_baru->tampilan_processor();
31 // Notice: Undefined property: laptop::$jenis_processor
32 ?>
```

Dalam kode di atas, saya membuat 2 buah class, yakni *class komputer*, dan *class laptop*. *Class laptop* merupakan turunan dari *class komputer*. Di dalam *class komputer* terdapat *property \$jenis_processor* dengan akses level **private**. Di dalam *class komputer* dan *class laptop*, saya membuat *method tampilan_processor()* yang digunakan untuk mengakses *property \$jenis_processor*.

Pengaksesan method *tampilan_processor()* dari **objek \$komputer_baru** sukses ditampilkan karena berada di dalam *satu class* dimana *property \$jenis_processor* berada.

Akan tetapi, jika method *tampilan_processor()* diakses dari objek *\$laptop_baru* yang merupakan turunan dari *class komputer*, PHP akan mengeluarkan error karena *property \$jenis_processor* tidak dikenal.

Akses level **private** sering digunakan untuk menyembunyikan *property* dan *method* agar tidak bisa diakses di luar class.



Home | Tutorial PHP | Tutorial Belajar OOP PHP Part 5: Pengertian dan Fungsi Variabel \$this dalam Pemrograman Objek

Tutorial Belajar OOP PHP Part 5: Pengertian dan Fungsi Variabel \$this dalam Pemrograman Objek

02 Oct 14 | Andre | Tutorial PHP | 40 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Setelah dalam tutorial OOP PHP sebelumnya kita mempelajari tentang [Pengertian Enkapsulasi Objek \(Public, Protected dan Private\)](#), dalam tutorial kali ini kita akan membahas [Pengertian dan Fungsi Variabel \\$this dalam Object Oriented Programming PHP](#).

Pengertian dan Fungsi Variabel \$this dalam OOP

Variabel **\$this** adalah sebuah *variabel khusus* dalam OOP PHP yang digunakan sebagai *penunjuk kepada objek, ketika kita mengaksesnya dari dalam class*. Dalam manual PHP, **\$this** disebut dengan istilah: *pseudo-variable*.



Konsep variabel **\$this** mungkin sedikit susah dipahami, terutama jika anda belum pernah menggunakan *Pemrograman berbasis objek* sebelumnya. Apabila penjelasan dalam tutorial ini membuat bingung, silahkan dipelajari secara perlahan dari awal.

Untuk lebih memudahkan pemahaman, kita akan bahas menggunakan contoh.

Berikut adalah *class laptop* dengan beberapa *property* dan *method*.



```

1 <?php
2
3 // buat class laptop
4 class laptop {
5
6     // buat property untuk class laptop
7     public $pemilik;
8     public $merk;
9     public $ukuran_layar;
10
11    // buat method untuk class laptop
12    public function hidupkan_laptop() {
13        return "Hidupkan Laptop";
14    }
15
16    public function matikan_laptop() {
17        return "Matikan Laptop";
18    }
19 }
20
21 // buat objek dari class laptop (instansiasi)
22 $laptop_ant = new laptop();
23 $laptop_andi = new laptop();
24 ?>

```

Dalam kode di atas, saya membuat **class laptop** dengan 3 *property*, yakni: **\$pemilik**, **\$merk** dan **\$ukuran_layar**. Ketiga *property* ini belum memiliki nilai. Di dalam **class laptop** juga terdapat 2 buah *method*, yakni **hidupkan_laptop()** dan **matikan_laptop()**, kedua *method* ini akan mengembalikan nilai **string**. Seluruh *property* dan *method* dari class laptop memiliki hak akses **public**, sehingga bisa diakses dari luar class.

Setelah membuat *class*, saya kemudian men-*instansiasi*-nya ke dalam 2 buah objek **\$laptop_ant** dan **\$laptop_andi**.

Jika anda menjalankan kode program di atas, di dalam *web browser* belum tampil apa-apa, karena saya belum memanggil *method* atau *property* apapun dari kedua objek.

Selanjutnya, saya ingin menambahkan isi *property* **\$pemilik** kepada kedua objek, berikut adalah kode yang diperlukan:

```

1 $laptop_ant->pemilik = "Anto";
2 $laptop_andi->pemilik = "Andi";

```

Dengan perintah di atas, *property* **\$pemilik** pada masing-masing objek telah berisi nilai. Untuk menampilkan nilai dari objek tersebut, kita tinggal mengaksesnya dengan kode berikut:

```

1 echo $laptop_ant->pemilik; // "Anto"

```

Sampai di sini, kita telah memahami cara mengakses *property* **objek_dari_objek_itu_sendiri**, yakni dengan menggunakan format:

```

1 $nama_objek->nama_property

```

Bagaimana jika *property* tersebut telah di-set nilainya dari dalam **class**? Berikut adalah perubahan **class laptop**:

List Tutorial DuniaIlkom

[Tutorial Terbaru DuniaIlkom](#)

[Tutorial HTML](#)

[Tutorial PHP](#)

[Tutorial MySQL](#)

[Tutorial CSS](#)

[Tutorial JavaScript](#)

[Tutorial jQuery](#)

[Tutorial WordPress](#)

[Tutorial Pascal](#)

[Tutorial C](#)

[Tutorial Python](#)

[Tutorial Java](#)

[Tutorial Laravel](#)

[Membuat Web Online](#)

[Review Jurusan Kuliah](#)

[Blog DuniaIlkom](#)

Tutorial Dasar PHP

[Tutorial PHP Dasar](#)

[1. Pengertian PHP](#)

[2. Sejarah PHP](#)

[3. Menginstall XAMPP](#)

[4. Menjalankan Apache](#)

[5. Menjalankan File PHP](#)

[6. Cara Kerja WebServer](#)

[7. Input PHP ke HTML](#)

[8. File php.ini](#)

[9. Dasar Penulisan PHP](#)

[10. Penulisan Komentar](#)

[11. Penulisan Variabel](#)

[12. Penulisan Konstanta](#)

[13. Tipe Data Integer](#)

[14. Tipe Data Float](#)

[15. Tipe Data String](#)

[16. Tipe Data Boolean](#)

[17. Tipe Data Array](#)

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     public $pemilik="Andi";
7
8     // buat method untuk class laptop
9     public function hidupkan_laptop() {
10        return "Hidupkan Laptop";
11    }
12 }
13
14 // buat objek dari class laptop (instansiasi)
15 $laptop_baru = new laptop();
16 $laptop_lama = new laptop();
17 ?>

```

Class *laptop* di atas saya revisi sehingga hanya memiliki 1 *property* dan 1 *method*. Perhatikan bahwa property **\$pemilik** telah diisi pada **level class**, sehingga pada saat pembuatan objek (*instansiasi*), seluruh objek akan memiliki nilai ini.

Objek *\$laptop_baru* dan *\$laptop_lama* sama-sama berasal dari class *laptop*, dan nilai *property \$pemilik* sama-sama berisi "Andi".

```

1 echo $laptop_baru->pemilik; // Andi
2 echo $laptop_lama->pemilik; // Andi

```

Lebih jauh lagi, pemanggilan method juga akan menghasilkan nilai yang sama:

```

1 echo $laptop_baru->hidupkan_laptop(); // "Hidupkan Laptop"
2 echo $laptop_lama->hidupkan_laptop(); // "Hidupkan Laptop"

```

Kita akan masuk ke dalam bagian terpenting. *Bagaimana jika saya ingin ketika method hidupkan_laptop() dipanggil, yang akan ditampilkan adalah : "Hidupkan Laptop Andi"?*

Apabila anda telah mempelajari dasar-dasar PHP, tentunya cara paling jelas adalah dengan mengubah *return* string di dalam method *hidupkan_laptop()* sebagai berikut:

```

1 public function hidupkan_laptop() {
2     return "Hidupkan Laptop Andi";
3 }

```

Cara tersebut tidak salah, dan akan menghasilkan nilai "Hidupkan Laptop Andi" pada saat pemanggilan method *hidupkan_laptop()*. Tetapi, bukankah kita telah memiliki property *\$pemilik* dengan nilai "Andi"? kita bisa menggunakan *property* ini dan menambahkannya ke dalam *method hidupkan_laptop*.

Untuk mencoba ide tersebut, silahkan ubah contoh *class laptop* sebelumnya menjadi berikut:

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assignment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     public $pemilik="Andi";
7     public $merk;
8
9     // buat method untuk class laptop
10    public function hidupkan_laptop() {
11        return "Hidupkan Laptop $pemilik";
12    }
13 }
14
15 // buat objek dari class laptop (instansiasi)
16 $laptop_baru = new laptop();
17
18 echo $laptop_baru->hidupkan_laptop(); // "Hidupkan Laptop"
19 ?>

```

Jika anda menjalankan kode di atas, hasil yang didapat adalah:

```

1 Notice: Undefined variable: pemilik in
2 D:\xampp\htdocs\oop_php\belajar_objek.php on line 11 Hidupkan Laptop

```

Dari penjelasan *error PHP*, dinyatakan bahwa “*variabel pemilik tidak terdefinisi pada baris 11*”. Baris tersebut adalah baris dimana kita menampilkan “*Hidupkan Laptop \$pemilik*” dengan *method hidupkan_laptop()*.

Apa yang terjadi? Bukankah *\$pemilik="Andi"* telah terdefinisi pada *property class*?

Inilah yang dimaksud dalam awal tutorial kali ini, kita sedang mencoba **mengakses property objek dari dalam class**. Jika anda bingung dengan pengertian ini, anda tidak sendiri. Saya juga kesulitan dalam memahami konsep ini pada pertama kali belajar *OOP*.

Untuk memahaminya, kita harus ingat bahwa class hanyalah sebuah “*blue print*” atau *kerangka*. Seluruh *property* dan *method* nantinya akan diakses dari dalam *objek*, *bukan dari dalam class*. Pada saat proses *instansiasi class*, seluruh *property* dan *method* akan “*dicopy*” ke dalam objek.

Perintah: *\$laptop_baru = new laptop()* akan membuat *objek \$laptop_baru* memiliki *property \$pemilik* dan *method hidupkan_laptop()* (sesuai dengan kerangka dari *class laptop*). Seluruh perintah lanjutan yang kita lakukan berada di dalam objek *\$laptop_baru*, bukan di dalam class laptop.

Ketika kita menjalankan *method hidupkan_laptop()*, *objek \$laptop_baru* akan menemui baris perintah:

```

1 "Hidupkan Laptop $pemilik"

```

variabel \$pemilik di sini *tidak terdefinisi* karena *\$pemilik* berada di dalam konteks *class, bukan objek*.

Untuk mengatasi masalah ini, kita harus mengubah definisi *method hidupkan_laptop()* menjadi:

```

1 "Hidupkan Laptop $this->pemilik";

```

Variabel *\$this* merujuk kepada *objek yang sedang menginstansiasi class*.

Jika di dalam class laptop kita membuat:

```

1 "Hidupkan Laptop $this->pemilik";

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Subscribe



Maka di dalam objek \$laptop_baru, akan menjadi:

```
1 "Hidupkan Laptop $laptop_baru->pemilik";
```

Jika kita memiliki objek lain misalkan \$laptop_lama, maka perintah tersebut akan dijalankan menjadi:

```
1 "Hidupkan Laptop $laptop_lama->pemilik";
```

Perhatikan bahwa pada level objek, variabel *\$this* 'dijalankan' menjadi objek saat ini.

Untuk menguji pemahaman kita, silahkan pelajari kode berikut ini:

```
1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     public $pemilik="Andi";
7
8     // buat method untuk class laptop
9     public function hidupkan_laptop() {
10         return "Hidupkan Laptop $this->pemilik";
11     }
12 }
13
14 // buat objek dari class laptop (instansiasi)
15 $laptop_baru = new laptop();
16
17 echo $laptop_baru->hidupkan_laptop(); // "Hidupkan Laptop Andi";
18
19 //ubah isi property $pemilik pada objek $laptop_baru
20 $laptop_baru->pemilik="Arie";
21
22 echo $laptop_baru->hidupkan_laptop(); // "Hidupkan Laptop Arie";
23
24 // buat objek baru dari class laptop dan panggil hidupkan_laptop()
25 $laptop_lama = new laptop();
26 echo $laptop_lama->hidupkan_laptop();
27 ?>
```

Tanpa menjalani kode program tersebut, silahkan tebak apa hasil yang akan ditampilkan dari perintah: `echo $laptop_lama->hidupkan_laptop()` yang berada pada baris terakhir, dan kenapa hasilnya bukan "*Hidupkan Laptop Arie*"?

Jika anda bisa menebak hasil akhir dengan benar dan bisa menjelaskan kenapa hasilnya bukan "*Hidupkan Laptop Arie*", maka anda telah memahami salah satu konsep terpenting dalam Pemrograman berbasis objek.

Jika di dalam tutorial kali ini kita membahas [pengertian dan fungsi variabel \\$this](#), dalam tutorial berikutnya kita akan membahas variabel \$this secara lebih jauh, dan melihat [cara penggunaan pseudo-variable \\$this dalam pemrograman objek PHP](#).

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoload* dan *exception*. Di akhir buku juga



[Home](#) | [Tutorial PHP](#) | Tutorial Belajar OOP PHP Part 6: Cara Penggunaan Pseudo-Variable \$this dalam Objek PHP

Tutorial Belajar OOP PHP Part 6: Cara Penggunaan Pseudo-Variable \$this dalam Objek PHP

02 Oct 14 | Andre | [Tutorial PHP](#) | 33 Comments

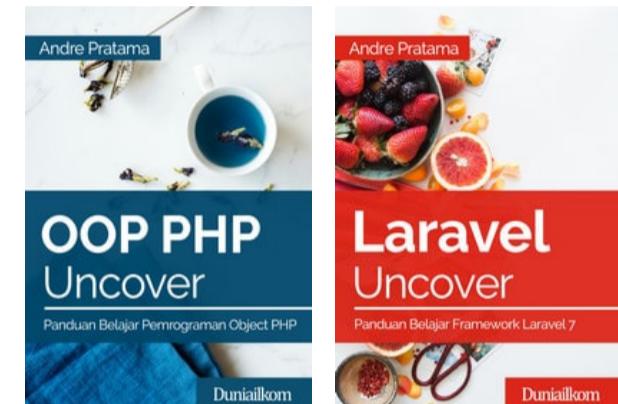
Jika dalam tutorial sebelumnya kita telah memahami [pengertian dan fungsi dari variabel \\$this](#), dalam tutorial kali ini saya akan membahas [cara penggunaan pseudo-variable \\$this dalam pemrograman Objek PHP](#). Tutorial kali ini hanya untuk memperjelas konsep variabel \$this yang telah kita pelajari sebelumnya.

Arti Pseudo-Variable \$this dalam Pemrograman Objek PHP

Melanjutkan tutorial mengenai variabel \$this, kali ini kita akan melihat cara penggunaannya melalui contoh program PHP yang lebih lengkap.

Dalam contoh berikut, saya membuat **class laptop** dengan **method** yang saling memanggil **method** lain menggunakan variabel \$this, silahkan anda pahami alur kerja dari **class** dan **objek** dibawah ini:

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

List Tutorial DuniaIlkom

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat property untuk class laptop
6     public $pemilik;
7     public $merk;
8
9     // buat method untuk class laptop
10    public function hidupkan_laptop() {
11        return "Hidupkan Laptop $this->merk punya $this->pemilik";
12    }
13
14    public function matikan_laptop() {
15        return "Matikan Laptop $this->merk punya $this->pemilik";
16    }
17
18    public function restart_laptop() {
19        $matikan=$this->matikan_laptop();
20        $hidupkan= $this->hidupkan_laptop();
21        $restart=$matikan."<br />".$hidupkan;
22        return $restart;
23    }
24 }
25
26 // buat objek dari class laptop (instansiasi)
27 $laptop_anto = new laptop();
28
29 // isi property objek
30 $laptop_anto->pemilik="Anto";
31 $laptop_anto->merk="Asus";
32
33
34 echo $laptop_anto->hidupkan_laptop();
35 // hasil: "Hidupkan Laptop Asus punya Anto";
36
37 echo "<br />";
38
39 echo $laptop_anto->matikan_laptop();
40 // hasil: "Matikan Laptop Asus punya Anto";
41
42 echo "<br />";
43
44 echo $laptop_anto->restart_laptop();
45 // hasil:
46 // "Matikan Laptop Asus punya Anto";
47 // "Hidupkan Laptop Asus punya Anto";
48 ?>
```

Contoh `class laptop` diatas, mirip dengan contoh-contoh kita sebelumnya, dengan beberapa modifikasi. Saya membuat 2 **property**: `$pemilik` dan `$merk`, kemudian membuat 3 method: `hidupkan_laptop()`, `matikan_laptop()`, dan `restart_laptop()`.

Dalam **method** `hidupkan_laptop()`, dan saya memanggil **property** `$pemilik` dan `$merk`. Karena property ini nantinya akan dipanggil dari **objek**, maka kita harus menggunakan variabel `$this`:

```
1 "Hidupkan Laptop $this->merk punya $this->pemilik";
```

Variabel `$this` nantinya akan ‘*merujuk*’ kepada **objek** yang memanggil **method**. Misalkan kita memiliki **objek** `$laptop_anto`, maka hasil yang dijalankan adalah:

```
1 "Hidupkan Laptop $laptop_anto ->merk punya $laptop_anto->pemilik";
```

Method `matikan_laptop()` juga menggunakan pola perintah yang sama.

Untuk **method** `restart_laptop()`, di dalam struktur **method** ini saya memanggil **method** `matikan_laptop()` dan `hidupkan_laptop()`. Karena alasan yang sama dengan **property** `$pemilik` dan `$merk`, saya juga menggunakan variabel `$this` untuk memanggil **method**. Hasil pemanggilan kedua **method**, kemudian disambung dan disimpan kedalam variabel `$restart`.

[Tutorial Terbaru DuniaIlkom](#)
[Tutorial HTML](#)
[Tutorial PHP](#)
[Tutorial MySQL](#)
[Tutorial CSS](#)
[Tutorial JavaScript](#)
[Tutorial jQuery](#)
[Tutorial WordPress](#)
[Tutorial Pascal](#)
[Tutorial C](#)
[Tutorial Python](#)
[Tutorial Java](#)
[Tutorial Laravel](#)
[Membuat Web Online](#)
[Review Jurusan Kuliah](#)
[Blog DuniaIlkom](#)

Tutorial Dasar PHP

[Tutorial PHP Dasar](#)
[1. Pengertian PHP](#)
[2. Sejarah PHP](#)
[3. Menginstall XAMPP](#)
[4. Menjalankan Apache](#)
[5. Menjalankan File PHP](#)
[6. Cara Kerja WebServer](#)
[7. Input PHP ke HTML](#)
[8. File php.ini](#)
[9. Dasar Penulisan PHP](#)
[10. Penulisan Komentar](#)
[11. Penulisan Variabel](#)
[12. Penulisan Konstanta](#)
[13. Tipe Data Integer](#)
[14. Tipe Data Float](#)
[15. Tipe Data String](#)
[16. Tipe Data Boolean](#)
[17. Tipe Data Array](#)
[18. Pengertian Operand](#)

```

1 public function restart_laptop() {
2     $matikan=$this->matikan_laptop();
3     $hidupkan= $this->hidupkan_laptop();
4     $restart=$matikan."<br />".$hidupkan;
5     return $restart;
6 }

```

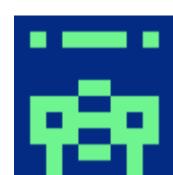
Tutorial kali ini ditujukan untuk melengkapi pemahaman variabel \$this dari tutorial sebelumnya. Dalam tutorial OOP PHP selanjutnya, kita akan membahas [cara membuat method dalam pemrograman objek PHP](#).

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class, object, property, hingga trait, namespace, autoloading dan exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

Tags: [Object Oriented Programming](#), [OOP PHP](#), [Pemrograman Berbasis Objek](#), [pengertian variabel this](#), [Pseudo-Variable this PHP](#), [tutorial belajar PHP](#), [Tutorial OOP PHP](#)

33 COMMENTS



mudiono

13 Nov 15

terimakasih telah memberikan pencerahan terhadap materi oop , untuk php

[Reply](#)

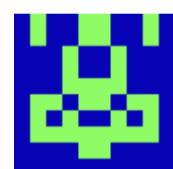


Andre Author

15 Nov 15

Sama2 mas, semoga bermanfaat :)

[Reply](#)



jaka

17 Dec 15

Makasi banyak nih, duniaIlkom atas penjelasan yg sangat mudah, ,sya baru setahun belajar program trutama html dan php ,di hari ini sya baru paham apa itu oop, sekali lagi makasi, semoga menjadi amal jariyah,

[Reply](#)



Andre Author

17 Dec 15

Amiin,... sama2 mas, saya juga senang tutorialnya bisa membantu untuk memahami konsep OOP :)

[Reply](#)

Eko Okda

[19. Fungsi var_dump\(\)](#)

[20. Operator Aritmatika](#)

[21. Operator String](#)

[22. Operator Logika](#)

[23. Perbandingan](#)

[24. Operator Increment](#)

[25. Assignment PHP](#)

[26. Operator Bitwise](#)

[27. Operator Assigment](#)

[28. Type Casting](#)

[29. Struktur Logika IF](#)

[30. Struktur ELSE](#)

[31. Logika ELSE-IF](#)

[32. Struktur Switch](#)

[33. Perulangan For](#)

[34. Perulangan While](#)

[35. Do-While](#)

[36. Perintah Break](#)

[37. Perintah Continue](#)

[38. Perulangan Foreach](#)

[39. Pengertian Function](#)

[40. Penulisan Function](#)

[41. Variabel Scope](#)

[42. Argumen Function](#)

[43. Default Parameter](#)

[44. Variable Parameter](#)

Tutorial PHP Lanjutan: Form

[Tutorial PHP Lanjutan: Form](#) ▾

Tutorial PHP Lanjutan: PHP - MySQL

[Tutorial PHP Lanjutan: PHP - MySQL](#) ▾

Tutorial PHP Lanjutan: OOP PHP

[Tutorial PHP Lanjutan: OOP PHP](#) ▾

Berlangganan Artikel DuniaIlkom

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom



Tutorial Belajar OOP PHP Part 7: Cara Membuat Method dalam Pemrograman Objek PHP

02 Oct 14 | Andre | [Tutorial PHP](#) | 8 Comments

Setelah memahami arti dan fungsi variabel `$this` dalam 2 tutorial sebelumnya, dalam sesi *tutorial belajar OOP PHP* kali ini, kita akan mempelajari lebih dalam tentang [cara membuat method di dalam pemrograman objek PHP](#). Method di dalam PHP juga bisa ditambahkan dengan argumen/parameter seperti layaknya *function*.

Cara Membuat Method dengan Argumen/Parameter

Karena method pada dasarnya hanyalah *function* yang berada di dalam sebuah *class*, maka kita bisa memberikan *argumen/parameter* ke dalam *method* tersebut.



Jika anda belum mengenal mengenai istilah *function, parameter*, dan *argumen*, silahkan mempelajarinya dalam tutorial [Pengertian Fungsi \(function\) PHP](#) dan [Cara Penggunaan Fungsi PHP](#).

Langsung saja kita lihat struktur dasar pembuatan parameter di dalam method PHP:

```
1  hak_akses nama_method ($argumen1, argumen2, dst...)
2  {
3      //... isi dari method
4 }
```

Dengan menggunakan contoh method *hidupkan_laptop()*, kita bisa membuatnya menjadi:

eBook Terbaru DuniaIlkom



[Cara pemesanan eBook & Buku](#)

DuniaIlkom

Daftar eBook DuniaIlkom

[eBook Pascal Uncover](#)

[eBook HTML Uncover](#)

[eBook CSS Uncover](#)

[eBook PHP Uncover](#)

[eBook MySQL Uncover](#)

[eBook JavaScript Uncover](#)

[eBook Bootstrap Uncover](#)

[eBook OOP PHP Uncover](#)

[eBook Laravel Uncover](#)

```

1 public hidupkan_laptop($pemilik, $merk)
2 {
3     //... isi dari method
4 }
```

Sehingga apabila metode itu dipanggil dari objek, kita tinggal mengisi argumen dengan nilai yang diinginkan, seperti contoh berikut:

```
1 $laptop_andi('Andi', 'Lenovo');
```

Cara Membuat Argumen dalam Method Class

Sebagai contoh tutorial, saya akan kembali memodifikasi *class laptop* dengan menambahkan fitur argumen dalam metode:

```

1 <?php
2 // buat class laptop
3 class laptop {
4     // buat method untuk class laptop
5     public function hidupkan_laptop($pemilik,$merk) {
6         return "Hidupkan Laptop $merk punya $pemilik";
7     }
8 }
9
10 // buat objek dari class laptop (instansiasi)
11 $laptop_andi= new laptop();
12
13 echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
14 // hasil: "Hidupkan Laptop Lenovo punya Andi";
15 ?>
```

Dalam contoh diatas, saya memanggil *method hidupkan_laptop()* dengan 2 argumen, yakni "*Andi*" dan "*Lenovo*". Kedua nilai ini akan diproses oleh *method hidupkan_laptop()*.

Perhatikan bahwa saya tidak menggunakan variabel *\$this*, karena argumen tersebut 'milik' *method*, perhatikan bedanya jika saya mengubah *class laptop* menjadi berikut ini:

```

1 <?php
2 // buat class laptop
3 class laptop {
4     // buat property untuk class laptop
5     private $pemilik="Anto";
6     private $merk="Acer";
7
8     // buat method untuk class laptop
9     public function hidupkan_laptop($pemilik,$merk) {
10        return "Hidupkan Laptop $merk punya $pemilik";
11    }
12
13     public function hidupkan_laptop_anto() {
14        return "Hidupkan Laptop $this->merk punya $this->pemilik";
15    }
16
17 // buat objek dari class laptop (instansiasi)
18 $laptop_andi= new laptop();
19
20 echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
21 // hasil: "Hidupkan Laptop Lenovo punya Andi";
22
23 echo $laptop_andi->hidupkan_laptop_anto();
24 // hasil: "Hidupkan Laptop Acer punya Anto";
25
26 ?>
```

Pada *class laptop* diatas, saya menambahkan 2 *property*: *\$pemilik* dan *\$merk*, kemudian memberikan nilai "*Anto*" dan "*Acer*". Jika yang kita inginkan adalah nilai dari *variabel* ini, maka di dalam *method*, kita harus menggunakan *\$this*.

Semua fitur *function*, juga bisa diterapkan di dalam *method*, termasuk *default parameter* seperti yang pernah kita bahas pada [Tutorial Belajar PHP: Cara Pembuatan Default Parameter pada Fungsi PHP](#), seperti contoh berikut:

List Tutorial DuniaIlkom

[Tutorial Terbaru DuniaIlkom](#)

[Tutorial HTML](#)

[Tutorial PHP](#)

[Tutorial MySQL](#)

[Tutorial CSS](#)

[Tutorial JavaScript](#)

[Tutorial jQuery](#)

[Tutorial WordPress](#)

[Tutorial Pascal](#)

[Tutorial C](#)

[Tutorial Python](#)

[Tutorial Java](#)

[Tutorial Laravel](#)

[Membuat Web Online](#)

[Review Jurusan Kuliah](#)

[Blog DuniaIlkom](#)

Tutorial Dasar PHP

[Tutorial PHP Dasar](#)

[1. Pengertian PHP](#)

[2. Sejarah PHP](#)

[3. Menginstall XAMPP](#)

[4. Menjalankan Apache](#)

[5. Menjalankan File PHP](#)

[6. Cara Kerja WebServer](#)

[7. Input PHP ke HTML](#)

[8. File php.ini](#)

[9. Dasar Penulisan PHP](#)

[10. Penulisan Komentar](#)

[11. Penulisan Variabel](#)

[12. Penulisan Konstanta](#)

[13. Tipe Data Integer](#)

[14. Tipe Data Float](#)

[15. Tipe Data String](#)

[16. Tipe Data Boolean](#)

[17. Tipe Data Array](#)

```

1 <?php
2 // buat class laptop
3 class laptop {
4     // buat method untuk class laptop
5     public function hidupkan_laptop($pemilik="Joko",$merk="Samsung")
6         return "Hidupkan Laptop $merk punya $pemilik";
7     }
8 }
9
10 // buat objek dari class laptop (instansiasi)
11 $laptop_andi= new laptop();
12
13 echo $laptop_andi->hidupkan_laptop();
14 // hasil: "Hidupkan Laptop Samsung punya Joko";
15
16 echo "<br />";
17
18 echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
19 // hasil: "Hidupkan Laptop Lenovo punya Andi";
20 ?>

```

Pada contoh kode program PHP diatas, dengan membuat *method* sebagai berikut:

```
1 public function hidupkan_laptop($pemilik="Joko",$merk="Samsung")
```

Maka, ketika *method* tersebut dipanggil tanpa menambahkan argumen, nilai "Joko" dan "Samsung" akan digunakan sebagai nilai *default*, namun jika argumen ditulis, nilai argumen yang diinput akan menimpa nilai *default* ini.

Dalam tutorial belajar *PHP Object Oriented Programming* kali ini, kita telah membahas cara membuat *method* di dalam *PHP* dengan menggunakan argumen / parameter. Kesimpulannya, karena *method* adalah sebutan lain untuk *function* di dalam *class*, maka seluruh fitur *function* bisa digunakan untuk *method*.

Dalam tutorial OOP PHP berikutnya, kita akan membahas tentang pengertian constructor dan destructor dalam pemrograman objek PHP.

*** Artikel Terkait ***

- [18. Pengertian Operand](#)
- [19. Fungsi var_dump\(\)](#)
- [20. Operator Aritmatika](#)
- [21. Operator String](#)
- [22. Operator Logika](#)
- [23. Perbandingan](#)
- [24. Operator Increment](#)
- [25. Assignment PHP](#)
- [26. Operator Bitwise](#)
- [27. Operator Assigment](#)
- [28. Type Casting](#)
- [29. Struktur Logika IF](#)
- [30. Struktur ELSE](#)
- [31. Logika ELSE-IF](#)
- [32. Struktur Switch](#)
- [33. Perulangan For](#)
- [34. Perulangan While](#)
- [35. Do-While](#)
- [36. Perintah Break](#)
- [37. Perintah Continue](#)
- [38. Perulangan Foreach](#)
- [39. Pengertian Function](#)
- [40. Penulisan Function](#)
- [41. Variabel Scope](#)
- [42. Argumen Function](#)
- [43. Default Parameter](#)
- [44. Variable Parameter](#)

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Tags: [Argumen Method](#), [Belajar Objek PHP](#), [Cara Membuat Method PHP](#), [Method PHP](#), [Object Oriented Programming](#), [OOP PHP](#), [Parameter Method PHP](#), [Pemrograman Berbasis Objek](#), [Pemrograman Objek PHP](#)

8 COMMENTS

Berlangganan Artikel DuniaIlkom



[Home](#) | [Tutorial PHP](#) | [Tutorial Belajar OOP PHP Part 8: Pengertian Constructor dan Destructor](#)

Tutorial Belajar OOP PHP Part 8: Pengertian Constructor dan Destructor

03 Oct 14 | [Andre](#) | [Tutorial PHP](#) | [25 Comments](#)

Melanjutkan tutorial *Belajar Object Oriented Programming (OOP) PHP*, kali ini kita akan membahas tentang [Pengertian Constructor dan Destructor](#) dalam Pemrograman Objek, khususnya OOP PHP.

Pengertian Constructor dalam OOP

Constructor (bahasa indonesia: *konstruktor*) adalah *method* khusus yang akan dijalankan secara otomatis pada saat sebuah *objek* dibuat (*instansiasi*), yakni ketika perintah “**new**” dijalankan.

Constructor biasa digunakan untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai awal kepada *property*, memanggil *method* internal dan beberapa proses lain yang digunakan untuk ‘*mempersiapkan*’ objek.

Dalam PHP, **constructor** dibuat menggunakan *method* : `__construct()`.

Pengertian Destructor dalam OOP

Destructor (bahasa indonesia: *destruktur*) adalah *method* khusus yang dijalankan secara otomatis pada saat sebuah objek *dihapus*. Di dalam PHP, *seluruh objek* secara otomatis dihapus ketika halaman PHP dimana objek itu berada selesai diproses. Tetapi kita juga dapat menghapus objek secara manual.

Destructor biasanya digunakan untuk ‘*membersihkan*’ beberapa *variabel*, atau menjalankan proses tertentu sebelum objek dihapus.

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover



Dalam PHP, *destructor* dibuat menggunakan method : `_destruct()`.



PHP memiliki fitur '*penghapusan objek massal*' yang dikenal dengan ***Garbage Collection***. Fitur *garbage collection* akan menghapus seluruh *objek* secara otomatis ketika halaman PHP selesai di eksekusi, sehingga akan memanggil method `_destruct()` dari seluruh objek.

Namun PHP menyediakan cara jika kita ingin menghapus objek sebelum mekanisme ini berjalan, yaitu menggunakan *fungsi unset()* atau mengisi *variabel objek* dengan nilai ***null***. Kita akan melihat contoh penggunaannya dengan contoh program dibawah.

Cara Penggunaan Destructor dan Constructor dalam PHP

Sebagai tutorial kita kali ini, berikut adalah contoh penggunaan *constructor* dan *destructor* dalam PHP:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     private $pemilik = "Andi";
6     private $merk = "Lenovo";
7
8     public function __construct(){
9         echo "Ini berasal dari Constructor Laptop";
10    }
11
12     public function hidupkan_laptop(){
13         return "Hidupkan Laptop $this->merk punya $this->pemilik";
14    }
15
16     public function __destruct(){
17         echo "Ini berasal dari Destructor Laptop";
18    }
19
20
21 // buat objek dari class laptop (instansiasi)
22 $laptop_andi= new laptop();
23
24 echo "<br />";
25 echo $laptop_andi->hidupkan_laptop();
26 echo "<br />";
27 ?>

```

Dalam contoh diatas, saya membuat **class laptop** dengan 3 *method*:

- Method `__construct()` merupakan *constructor* dari **class laptop**. Method ini akan dipanggil secara otomatis ketika **class laptop** di *instansiasi*.
- Method `hidupkan_laptop()` merupakan method 'biasa' yang akan menampilkan hasil string. Untuk menggunakan method ini, kita memanggilnya dari objek.
- Method ketiga adalah `__destruct()` yang merupakan *destructor* dari **class laptop**. Method ini akan dipanggil saat objek dihapus.

Setelah pendefinisian **class**, saya membuat **objek \$laptop_andi**, dan memanggil method `hidupkan_laptop()`. Berikut adalah hasil yang didapat:

```

1 Ini berasal dari Constructor Laptop
2 Hidupkan Laptop Lenovo punya Andi
3 Ini berasal dari destructor Laptop

```

List Tutorial DuniaIlkom

Tutorial Terbaru DuniaIlkom

Tutorial HTML

Tutorial PHP

Tutorial MySQL

Tutorial CSS

Tutorial JavaScript

Tutorial jQuery

Tutorial WordPress

Tutorial Pascal

Tutorial C

Tutorial Python

Tutorial Java

Tutorial Laravel

Membuat Web Online

Review Jurusan Kuliah

Blog DuniaIlkom

Tutorial Dasar PHP

Tutorial PHP Dasar

1. Pengertian PHP

2. Sejarah PHP

3. Menginstall XAMPP

4. Menjalankan Apache

5. Menjalankan File PHP

6. Cara Kerja WebServer

7. Input PHP ke HTML

8. File php.ini

9. Dasar Penulisan PHP

10. Penulisan Komentar

11. Penulisan Variabel

12. Penulisan Konstanta

13. Tipe Data Integer

14. Tipe Data Float

15. Tipe Data String

16. Tipe Data Boolean

17. Tipe Data Array

Seperti yang terlihat, *method __construct()* dan *__destruct()* secara otomatis dipanggil saat *objek* dibuat dan saat *objek* dihapus. Untuk mencoba menghapus *objek \$laptop_andi* secara *manual*, kita bisa menggunakan fungsi *unset()* sebagai berikut:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     private $pemilik = "Andi";
6     private $merk = "Lenovo";
7
8     public function __construct(){
9         echo "Ini berasal dari Constructor Laptop";
10    }
11
12     public function hidupkan_laptop(){
13         return "Hidupkan Laptop $this->merk punya $this->pemilik";
14    }
15     public function __destruct(){
16         echo "Ini berasal dari Destructor Laptop";
17    }
18 }
19
20 // buat objek dari class laptop (instansiasi)
21 $laptop_andi= new laptop();
22
23 echo "<br />";
24 echo $laptop_andi->hidupkan_laptop();
25 echo "<br />";
26
27 // hapus objek $laptop_andi
28 unset($laptop_andi);
29
30 echo "<br />";
31 echo "Objek Telah Dihancurkan";
32
33 ?>

```

Jika kita menjalankan kode diatas, berikut adalah hasil yang didapat:

```

1 Ini berasal dari Constructor Laptop
2 Hidupkan Laptop Lenovo punya Andi
3 Ini berasal dari Destructor Laptop
4 Objek Telah Dihancurkan

```

Setelah memanggil *method \$laptop_andi->hidupkan_laptop()*, saya kemudian menghapus *objek \$laptop_andi* secara manual menggunakan fungsi *unset(\$laptop_andi)*.

Untuk membuktikan bahwa *constructor \$laptop_andi* sudah dijalankan, saya menambahkan perintah *echo "Objek Telah Dihancurkan"* diakhir halaman. Sehingga kita bisa melihat bahwa *constructor* objek *\$laptop_andi* di jalankan sebelum dihapus otomatis oleh PHP. Silahkan anda coba hapus perintah *unset(\$laptop_andi)*, maka string "Objek Telah Dihancurkan" akan tampil sebelum *constructor* objek *\$laptop_andi*.



Anda juga bisa mengganti perintah *unset(\$laptop_andi)* dengan *\$laptop_andi=null*. Kedua perintah ini akan menghapus objek *\$laptop_andi* dan men-trigger pemanggilan *constructor*.

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

Sebagai contoh terakhir dalam tutorial kali ini, saya akan membuat contoh *constructor* yang sering digunakan untuk membuat *objek* dengan nilai awal. Konsep ini sering digunakan dalam pemrograman objek. Berikut adalah contoh penggunaannya:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     private $pemilik;
6     private $merk;
7
8     // constructor sebagai pembuat nilai awal
9     public function __construct($pemilik, $merk) {
10         $this->pemilik = $pemilik;
11         $this->merk = $merk;
12     }
13
14     public function hidupkan_laptop() {
15         return "Hidupkan Laptop $this->merk punya $this->pemilik";
16     }
17 }
18
19 // buat objek dari class laptop (instansiasi)
20 $laptop_andi= new laptop("Andi", "Lenovo");
21
22 echo $laptop_andi->hidupkan_laptop();
23 echo "<br />";
24
25 $laptop_anto= new laptop("Anto", "Acer");
26
27 echo $laptop_anto->hidupkan_laptop();
28 ?>
```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Subscribe

Pada kode diatas, saya menggunakan *constructor* sebagai pembuat nilai awal dari *objek*. Method *constructor* menerima 2 buah *argumen* yang kemudian disimpan kedalam *property* internal *objek*.

Berikut adalah hasil yang didapat dari kode program diatas:

```

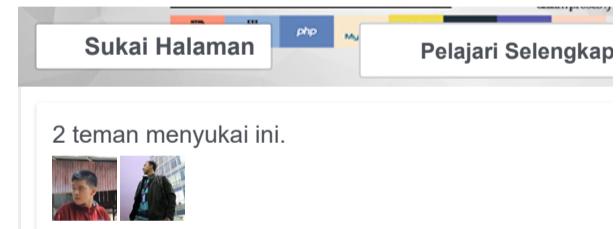
1 Hidupkan Laptop Lenovo punya Andi
2 Hidupkan Laptop Acer punya Anto
```



Di dalam PHP, *constructor* dan *destructor* harus memiliki hak akses **public**. Jika anda mengubah hak akses *constructor* atau *destructor* menjadi **protected** atau **private**, PHP akan mengeluarkan *error*:

```

1 Fatal error: Call to private laptop::__construct()
2 from invalid context
```



Dalam tutorial kali ini, kita telah membahas tentang [pengertian constructor dan destructor dalam Pemrograman objek PHP](#). Kemudian kita juga telah melihat contoh pembuatan *class* dan *objek* menggunakan *constructor* dan *destructor*. Dalam tutorial belajar OOP PHP berikutnya, kita akan membahas konsep penting lainnya di dalam OOP, yakni [penurunan/pewarisan \(inheritance\)](#) dalam Pemrograman objek.

*** Artikel Terkait ***



Home | Tutorial PHP | Tutorial Belajar OOP PHP Part 9: Pengertian Inheritance (Pewarisan)

Tutorial Belajar OOP PHP Part 9: Pengertian Inheritance (Pewarisan)

03 Oct 14 | Andre | Tutorial PHP | 35 Comments

Setelah mempelajari tentang [Pengertian Constructor dan Destructor dalam OOP PHP](#), melanjutkan tutorial belajar pemrograman objek, kali ini kita akan membahas dan mempelajari [Pengertian Inheritance atau Pewarisan dalam Pemrograman Objek](#), serta melihat contoh penggunaannya.

Pengertian Inheritance (Pewarisan) dalam OOP

Inheritance atau *Pewarisan/Penurunan* adalah konsep pemrograman dimana sebuah *class* dapat '*menurunkan*' *property* dan *method* yang dimilikinya kepada *class* lain. Konsep *inheritance* digunakan untuk memanfaatkan fitur '*code reuse*' untuk menghindari duplikasi kode program.

Konsep *inheritance* membuat sebuah struktur atau '*hierarchy class*' dalam kode program. Class yang akan '*diturunkan*' bisa disebut sebagai *class induk (parent class)*, *super class*, atau *base class*. Sedangkan class yang '*menerima penurunan*' bisa disebut sebagai *class anak (child class)*, *sub class*, *derived class* atau *heir class*.

Tidak semua *property* dan *method* dari *class* induk akan diturunkan. *Property* dan *method* dengan hak akses *private*, tidak akan diturunkan kepada *class* anak. Hanya *property* dan *method* dengan hak akses *protected* dan *public* saja yang bisa diakses dari *class* anak.

Cara Penggunaan Inheritance dalam PHP

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Di dalam PHP, *inheritance / penurunan* dari sebuah *class* kepada *class* lain menggunakan kata kunci: '*extends*', dengan penulisan dasar sebagai berikut:

```

1  class induk {
2      //...isi class induk
3  }
4
5  class anak extends induk
{
    //... class anak bisa mengakses
    //... property dan method class induk
}

```

Agar lebih mudah dipahami, kita akan langsung masuk kedalam contoh program penggunaan *inheritance/penurunan* di dalam PHP:

```

1  <?php
2  // buat class induk: komputer
3  class komputer {
4
5      public $merk;
6      public $processor;
7      public $memory;
8
9      public function beli_komputer() {
10         return "Beli komputer baru";
11     }
12 }
13
14 // turunkan class komputer ke laptop
15 class laptop extends komputer {
16
17     public function lihat_spec() {
18         return "merk: $this->merk, processor: $this->processor,
19             memory: $this->memory";
20     }
21 }
22
23 // buat objek dari class laptop (instansiasi)
24 $laptop_baru = new laptop();
25
26 // isi property objek
27 $laptop_baru->merk = "acer";
28 $laptop_baru->processor = "intel core i5";
29 $laptop_baru->memory = "2 GB";
30
31 // panggil method objek
32 echo $laptop_baru->beli_komputer();
33 echo "<br />";
34 echo $laptop_baru->lihat_spec();
35 ?>

```

Dalam contoh kode diatas, saya membuat *class komputer* dengan beberapa *property* dan sebuah *method*. *Property* class *komputer* belum berisi nilai apa-apa.

Dibawah *class komputer*, saya membuat *class laptop extends class komputer*. Disini saya menurunkan *class komputer* kedalam *class laptop*. Di dalam *class laptop*, kita bisa mengakses seluruh *property* dan *method* apapun dari *class komputer* selama memiliki hak akses *public* atau *protected*.

Untuk membuktikan hal tersebut, saya membuat objek *\$laptop_baru* dari *class laptop*. Perhatikan bahwa kita bisa mengakses *property \$merk*, *\$processor*, dan *\$memory* yang semuanya adalah milik *class komputer*, bukan *class laptop*. Method *beli_komputer()* juga sukses diakses dari objek *\$laptop_baru*. Inilah yang dimaksud dengan *inheritance/penurunan* class dalam OOP.

PHP tidak membatasi berapa banyak '*penurunan objek*' yang bisa dilakukan, dalam contoh berikut, saya membuat 3 buah class yang saling '*menurunkan*:

List Tutorial DuniaIlkom

- [Tutorial Terbaru DuniaIlkom](#)
- [Tutorial HTML](#)
- [Tutorial PHP](#)
- [Tutorial MySQL](#)
- [Tutorial CSS](#)
- [Tutorial JavaScript](#)
- [Tutorial jQuery](#)
- [Tutorial WordPress](#)
- [Tutorial Pascal](#)
- [Tutorial C](#)
- [Tutorial Python](#)
- [Tutorial Java](#)
- [Tutorial Laravel](#)
- [Membuat Web Online](#)
- [Review Jurusan Kuliah](#)
- [Blog DuniaIlkom](#)

Tutorial Dasar PHP

- [Tutorial PHP Dasar](#)
- [1. Pengertian PHP](#)
- [2. Sejarah PHP](#)
- [3. Menginstall XAMPP](#)
- [4. Menjalankan Apache](#)
- [5. Menjalankan File PHP](#)
- [6. Cara Kerja WebServer](#)
- [7. Input PHP ke HTML](#)
- [8. File php.ini](#)
- [9. Dasar Penulisan PHP](#)
- [10. Penulisan Komentar](#)
- [11. Penulisan Variabel](#)
- [12. Penulisan Konstanta](#)
- [13. Tipe Data Integer](#)
- [14. Tipe Data Float](#)
- [15. Tipe Data String](#)
- [16. Tipe Data Boolean](#)
- [17. Tipe Data Array](#)

```

1 <?php
2 // buat class komputer
3 class komputer {
4     protected function beli_komputer() {
5         return "Beli komputer baru";
6     }
7 }
8
9 // turunkan class komputer ke laptop
10 class laptop extends komputer {
11     protected function beli_laptop() {
12         return "Beli laptop baru";
13     }
14 }
15
16 // turunkan class laptop ke chromebook
17 class chromebook extends laptop {
18     protected function beli_chromebook() {
19         return "Beli chromebook baru";
20     }
21
22     public function beli_semua(){
23         $a = $this->beli_komputer();
24         $b = $this->beli_laptop();
25         $c = $this->beli_chromebook();
26         return "$a <br /> $b <br /> $c";
27     }
28 }
29
30 // buat objek dari class laptop (instansiasi)
31 $gadget_baru = new chromebook();
32
33 // panggil method objek
34 echo $gadget_baru->beli_semua();
35
36 // $gadget_baru->beli_komputer();
37 // Fatal error: Call to protected method komputer::beli_komputer()
38 ?>

```

Dalam contoh diatas, saya membuat **class komputer** yang diturunkan kepada **class laptop**, dan kemudian diturunkan lagi kepada **class chromebook**. Dari dalam **class chromebook** ini kemudian saya memanggil **method** dari **class** diatasnya.

Jika anda perhatikan, setiap method selain **method beli_semua()**, memiliki hak akses **protected**. Hak akses **protected** ini 'menghalangi' kode program lain untuk mengaksesnya, selain **class** turunan.

Pada baris terakhir, saya menyisipkan kode program untuk mencoba mengakses **method beli_komputer()**. Kode ini sengaja saya beri tanda komentar. Jika anda menghapus tanda komentar, PHP akan mengeluarkan error yang menyatakan kita tidak bisa mengakses method dengan hak akses **protected**:

```

1 <?
2 $gadget_baru->beli_komputer();
3 // Fatal error: Call to protected method komputer::beli_komputer()
4 ?>

```

Inilah yang dimaksud dengan **enkapsulasi** dalam OOP. Membatasi **method** yang tidak boleh diakses akan membuat kode program menjadi lebih terstruktur.



Penjelasan tentang **enkapsulasi** telah kita bahas dalam [Tutorial OOP PHP: Pengertian Enkapsulasi Objek \(Public, Protected dan Private\)](#)

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

[Tutorial PHP Lanjutan: Form](#) ▾

Tutorial PHP Lanjutan: PHP - MySQL

[Tutorial PHP Lanjutan: PHP - MySQL](#) ▾

Tutorial PHP Lanjutan: OOP PHP

[Tutorial PHP Lanjutan: OOP PHP](#) ▾

Berlangganan Artikel DuniaIlkom

Dalam tutorial belajar OOP PHP kali ini, kita telah mempelajari konsep [inheritance atau pewarisan di dalam pemrograman objek](#). Dalam beberapa tutorial selanjutnya, kita akan memperdalam konsep inheritance.

Penurunan class ini akan memberikan permasalahan tersendiri ketika terdapat **property** atau **method** dengan nama yang sama pada parent class dan child class. Mengenai hal ini akan kita bahas dalam tutorial belajar OOP PHP berikutnya: [Cara Mengakses Property dan Method Parent Class](#).

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Email Address

Subscribe

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class, object, property, hingga trait, namespace, autoloading* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

*** Artikel Terkait ***

Tags: [Belajar Inheritance Objek](#), [Belajar Objek PHP](#), [Object Oriented Programming](#), [Pemrograman Berbasis Objek](#), [Pemrograman Objek PHP](#), [Pengertian Inheritance](#), [Pengertian Pewarisan Objek](#), [Tutorial OOP PHP](#)

35 COMMENTS

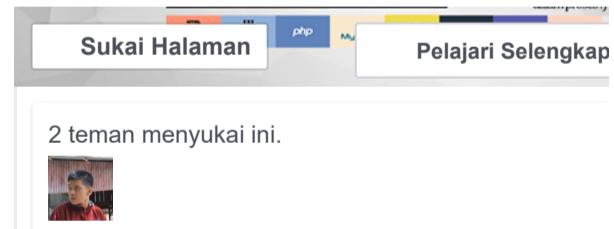


Falah

16 Dec 14

Saya coba contoh yang terakhir ternyata tidak error, hasilnya :
 "Beli komputer baru
 Beli laptop baru
 Beli chromebook baru"
 Gimana ini ya?

bukannya harusnya tidak error? kan class komputer di protected tetapi di akses oleh class chrome yang merupakan publik?





[Home](#) | [Tutorial PHP](#) | Tutorial Belajar OOP PHP Part 10: Cara Mengakses Property dan Method Parent Class

Tutorial Belajar OOP PHP Part 10: Cara Mengakses Property dan Method Parent Class

03 Oct 14 | [Andre](#) | [Tutorial PHP](#) | 7 Comments

Melanjutkan tutorial OOP PHP mengenai [Inheritance](#), kali ini kita akan mempelajari cara mengakses property dan method dari parent class, serta pengertian *Scope Resolution Operator*.

Cara Mengakses Property dan Method Parent Class

Konsep *pewarisan/inheritance* dimana sebuah *class* bisa memiliki *property* dan *method* dari *class* lain, bisa menjadi permasalahan ketika *property* atau *method* dari *class* anak memiliki nama yang sama dengan *class* induk, atau dikenal dengan istilah *overridden property* dan *overridden method*.

Untuk memahami pengertian *overridden property* dan *overridden method*, perhatikan contoh kode program berikut ini:

eBook Terbaru DuniaIlkom



[Cara pemesanan eBook & Buku](#)

DuniaIlkom

Daftar eBook DuniaIlkom

- [eBook Pascal Uncover](#)
- [eBook HTML Uncover](#)
- [eBook CSS Uncover](#)
- [eBook PHP Uncover](#)
- [eBook MySQL Uncover](#)
- [eBook JavaScript Uncover](#)
- [eBook Bootstrap Uncover](#)
- [eBook OOP PHP Uncover](#)
- [eBook Laravel Uncover](#)

```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     public function lihat_spec() {
6         return "Spec Komputer: Acer,
7             Processor Intel core i7, Ram 4GB";
8     }
9
10
11 // turunkan class komputer ke laptop
12 class laptop extends komputer {
13
14     public function lihat_spec() {
15         return "Spec Laptop: Asus,
16             Processor Intel core i5, Ram 2GB";
17     }
18
19 // buat objek dari class laptop (instansiasi)
20 $gadget_baru = new laptop();
21
22 // panggil method lihat_spec()
23 echo $gadget_baru->lihat_spec();
24 ?>

```

Pada kode program diatas, saya membuat 2 buah **class**: *komputer* dan *laptop*. Saya menurunkan **class** *komputer* kedalam **class** *laptop*, sehingga seluruh *property* dan *method* dari **class** *komputer* bisa diakses dari **class** *laptop*.

Namun perhatikan bahwa *method* pada **class** *komputer* memiliki nama yang sama dengan *method* dalam **class** *laptop*. Ketika kita memanggil *method* *lihat_spec()*, *method* manakah yang akan dijalankan?

Jika anda menjalankan kode diatas, maka hasilnya adalah:

```
1 Spec Laptop: Asus, Processor Intel core i5, Ram 2GB
```

Berdasarkan hasil yang di dapat, terlihat bahwa *method* yang dijalankan adalah *method* milik **class** *laptop*.

Di dalam PHP, ketika nama *property* atau nama *method* **child class** memiliki nama yang sama dengan **parent class**, maka yang dijalankan adalah *property* atau *method* milik **child class**.

Jadi, bagaimana cara mengakses *property* dan *method* milik **class** *komputer*? PHP mengatasi hal ini dengan menggunakan '**Scope Resolution Operator**'.

Pengertian Scope Resolution Operator PHP

Scope Resolution Operator adalah operator khusus di dalam PHP yang memungkinkan kita untuk mengakses '*informasi khusus*' dari dalam **class**.

Informasi khusus ini terdiri dari: **overridden property** atau **overridden method**, **static property** atau **static method**, serta **constanta class**. Untuk saat ini, kita akan fokus kepada **overridden property** atau **overridden method**. Mengenai **static property**, **static method**, dan **konstanta class** akan kita bahas dalam tutorial lainnya.

Scope Resolution Operator ditulis dengan tanda dua kali titik dua (*double colon*), yakni `::::`. Untuk mengakses *property* dan *method* dari **class** induk, kita mengaksesnya dengan perintah:

```
1 parent::nama_property;
2 parent::nama_method();
```

List Tutorial DuniaIlkom

Tutorial Terbaru DuniaIlkom

Tutorial HTML

Tutorial PHP

Tutorial MySQL

Tutorial CSS

Tutorial JavaScript

Tutorial jQuery

Tutorial WordPress

Tutorial Pascal

Tutorial C

Tutorial Python

Tutorial Java

Tutorial Laravel

Membuat Web Online

Review Jurusan Kuliah

Blog DuniaIlkom

Tutorial Dasar PHP

Tutorial PHP Dasar

1. Pengertian PHP

2. Sejarah PHP

3. Menginstall XAMPP

4. Menjalankan Apache

5. Menjalankan File PHP

6. Cara Kerja WebServer

7. Input PHP ke HTML

8. File php.ini

9. Dasar Penulisan PHP

10. Penulisan Komentar

11. Penulisan Variabel

12. Penulisan Konstanta

13. Tipe Data Integer

14. Tipe Data Float

15. Tipe Data String

16. Tipe Data Boolean

17. Tipe Data Array

Kembali kepada contoh program, kali ini kita ingin menampilkan method *lihat_spec()* dari *class komputer*:

```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     public function lihat_spec() {
6         return "Spec Komputer: Acer,
7             Processor Intel core i7, Ram 4GB";
8     }
9
10
11 // turunkan class komputer ke laptop
12 class laptop extends komputer {
13
14     public function lihat_spec() {
15         return "Spec Laptop: Asus,
16             Processor Intel core i5, Ram 2GB";
17     }
18
19     public function lihat_spec_komputer() {
20         return parent::lihat_spec();
21     }
22 }
23 // buat objek dari class laptop (instansiasi)
24 $gadget_baru = new laptop();
25
26 //panggil method lihat_spec()
27 echo $gadget_baru->lihat_spec();
28
29 echo "<br />";
30
31 //panggil method lihat_spec_komputer()
32 echo $gadget_baru->lihat_spec_komputer();
33 ?>

```

Kode program diatas adalah revisi dari contoh kita sebelumnya. Saya menambahkan sebuah method *lihat_spec_komputer()* kedalam *class laptop*. Method ini selanjutnya akan memanggil method *class komputer*, dengan perintah *parent::lihat_spec()*.

Hasilnya adalah:

```

1 Spec Laptop: Asus, Processor Intel core i5, Ram 2GB
2 Spec Komputer: Acer, Processor Intel core i7, Ram 4GB

```

Contoh diatas adalah cara mengakses **method parent class** dari *child class*. Lalu bagaimana dengan cara mengakses **parent property** dari *child class*? Jika anda berfikir sama seperti saya, maka kita bisa tebak bahwa caranya adalah menggunakan *parent::nama_property*, apakah bisa? mari kita coba:

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel Duniailkom ^

```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     public $merk = "acer";
6
7     public function lihat_spec() {
8         return "Spec Komputer: Acer,
9             Processor Intel core i7, Ram 4GB";
10    }
11 }
12
13 // turunkan class komputer ke laptop
14 class laptop extends komputer {
15     public $merk = "asus";
16
17     public function lihat_spec() {
18         return "Spec Laptop: Asus,
19             Processor Intel core i5, Ram 2GB";
20     }
21
22     public function lihat_spec_komputer() {
23         return parent::lihat_spec();
24     }
25     public function lihat_merk_komputer() {
26         return parent::$merk;
27     }
28 }
29 // buat objek dari class laptop (instansiasi)
30 $gadget_baru = new laptop();
31
32 //panggil method lihat_spec()
33 echo $gadget_baru->lihat_spec();
34
35 echo "<br />";
36
37 //panggil method lihat_spec_komputer()
38 echo $gadget_baru->lihat_spec_komputer();
39
40 //panggil method lihat_merk_komputer()
41 echo $gadget_baru->lihat_merk_komputer();
42 ?>

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Subscribe

Dalam kode program diatas, saya menambahkan sebuah *method* baru:

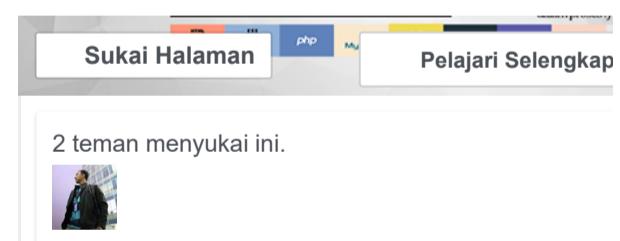
lihat_merk_komputer() yang akan menjalankan perintah *return parent::\$merk*. Jika kita menjalankan program diatas, hasilnya adalah:

```

1 Spec Laptop: Asus, Processor Intel core i5, Ram 2GB
2 Spec Komputer: Acer, Processor Intel core i7, Ram 4GB
3 <b>Fatal error</b>: Access to undeclared static property: komputer::

```

Apa yang terjadi?



Error diatas juga membuat saya bingung saat pertama kali mencobanya. Kode error diatas berarti kita mencoba mengakses *property static \$merk* dari *class komputer*, bukan *property public \$merk* dari *class komputer* sebagaimana kode error (*Property static* tidak sama dengan *property biasa*, kita akan mempelajarinya sesaat lagi di: [Pengertian static property dan static method](#)).

Pengertian *static property* belum kita pelajari sampai tutorial kali ini, dan akan saya bahas lengkap pada tutorial tersendiri. Namun berdasarkan kode error tersebut dan membaca beberapa sumber, saya mendapati bahwa *kita tidak bisa mengakses property parent class secara langsung*.

Jadi, bagaimana caranya?

Yang perlu menjadi perhatian disini adalah bahwa kode kita disini memiliki nama method dan property yang sama baik di child class dan juga pada parent class. Solusi yang paling mudah adalah: tidak menggunakan nama property dan method yang sama. Dengan demikian, kode program akan menjadi lebih jelas dan lebih mudah untuk dipahami.

Dalam tutorial kali ini, kita telah mempelajari [cara mengakses property dan method parent class](#) dengan **Scope Resolution Operator**, atau tanda ‘::’.

Dalam tutorial selanjutnya, masih berkaitan dengan *inheritance*, kita akan mempelajari efek penurunan ini kedalam **constructor** dan **destructor**. Kita akan mempelajari [cara mengakses constructor dan destructor parent class](#).

*** Artikel Terkait ***

Tags: [Belajar Inheritance Objek](#), [Belajar Objek PHP](#), [method parent class](#), [Object Oriented Programming](#), [OOP PHP](#), [Pemrograman Berbasis Objek](#), [property parent class](#), [Scope Resolution Operator](#)

7 COMMENTS



holong

01 Dec 14

mas kok outputnya tetap puna class laptop bukann class komputer,,pie
to mas,, apa aku yang salah mengartikannya ?
minta penjelasannya masss

[Reply](#)



Andre Author

02 Dec 14

Maaf gan, setelah saya baca ulang, memang terdapat kesalahan
dari penggunaan `$this` di parent class. Variabel `$this` ini ternyata
tetap merujuk kedalam child class, bukan parent-nya. Saya sudah





[Home](#) | [Tutorial PHP](#) | Tutorial Belajar OOP PHP Part 11: Cara Mengakses Constructor dan Destructor Parent Class

Tutorial Belajar OOP PHP Part 11: Cara Mengakses Constructor dan Destructor Parent Class

04 Oct 14 | Andre | [Tutorial PHP](#) | No Comments

Setelah membahas [cara mengakses property dan method parent class](#), dalam *tutorial belajar OOP PHP* kali ini akan membahas [cara mengakses constructor dan destructor parent class](#). Tutorial kali ini masih berkaitan dengan efek pewarisan (inheritance) dalam pemrograman objek.

Efek Inheritance dalam Constructor dan Destructor

Seperti yang telah kita pelajari dalam tutorial *Tutorial Belajar OOP PHP: Pengertian Constructor dan Destructor*, **konstruktor** dan **destructor** adalah *method khusus* yang dijalankan secara otomatis ketika sebuah **class** di *instansiasi* ke dalam sebuah **objek**, dan ketika objek tersebut dihapus.

Konsep **inheritance** atau pewarisan **class** memiliki efek khusus dalam **konstruktor** dan **destructor**. Terutama **konstruktor** dan **destructor** dari **parent class**.

Mari kita jelaskan dengan menggunakan contoh:

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover



```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     // buat constructor class komputer
6     public function __construct() {
7         echo "Constructor dari class komputer <br />";
8     }
9
10    // buat destructor class komputer
11    public function __destruct() {
12        echo "Destructor dari class komputer <br />";
13    }
14
15    // turunkan class komputer ke laptop
16    class laptop extends komputer {
17    }
18
19    // turunkan class laptop ke chromebook
20    class chromebook extends laptop {
21    }
22    // buat objek dari class chromebook (instansiasi)
23    $gadget_baru = new chromebook();
24
25    echo "Belajar OOP PHP <br />";
26
27 ?>

```

Pada kode diatas, saya membuat **class** komputer dengan **constructor** dan **destructor**. **Class komputer** kemudian diturunkan kepada **class laptop**, kemudian diturunkan kembali kepada **class chromebook**. Baik **class laptop** maupun **class chromebook** tidak memiliki property maupun method. **Class chromebook** inilah yang akan kita *instansiasi* kedalam objek **\$gadget_baru**.

Ketika program itu dijalankan, berikut adalah hasil yang didapat:

```

1 Construct dari class komputer
2 Belajar OOP PHP
3 Destructor dari class komputer

```

Dari hasil tersebut terlihat bahwa **constructor** dan **destructor** **class komputer** tetap dijalankan walaupun kita membuat objek dari **class chromebook**.

Pengertian Overridden Constructor dan Overridden Destructor

Dalam kode program diatas, saya tidak membuat **constructor** dan **destructor** untuk **class laptop** dan **class chromebook**. Tapi bagaimana jika ketiga **class** ini juga memiliki **constructor** dan **destructor**? Mari kita coba:

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array

```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     // buat constructor class komputer
6     public function __construct() {
7         echo "Constructor dari class komputer <br />";
8     }
9
10    // buat destructor class komputer
11    public function __destruct() {
12        echo "Destructor dari class komputer <br />";
13    }
14
15    // turunkan class komputer ke laptop
16    class laptop extends komputer {
17
18        // buat constructor class laptop
19        public function __construct() {
20            echo "Constructor dari class laptop <br />";
21        }
22
23        // buat destructor class laptop
24        public function __destruct() {
25            echo "Destructor dari class laptop";
26        }
27
28    }
29
30    // turunkan class laptop ke chromebook
31    class chromebook extends laptop {
32
33        // buat constructor class chromebook
34        public function __construct() {
35            echo "Constructor dari class chromebook <br />";
36        }
37
38        // buat destructor class chromebook
39        public function __destruct() {
40            echo "Destructor dari class chromebook <br />";
41        }
42
43        // buat objek dari class chromebook (instansiasi)
44        $gadget_baru = new chromebook();
45
46        echo "Belajar OOP PHP <br />";
47    ?>

```

Kode diatas memiliki *constructor* dan *destructor* pada masing-masing *class*, mari kita lihat hasilnya:

```

1 Constuctor dari class chromebook
2 Belajar OOP PHP
3 Destructor dari class chromebook

```

Kemana *constructor* dan *destructor* class lainnya?

Di dalam PHP, ketika *child class* memiliki *constructor* dan *destructor* sendiri, maka PHP akan melewatkannya *constructor* dan *destructor* *parent class*, kasus ini disebut dengan *Overridden Constructor* dan *Overridden Destructor*. Karena di dalam contoh kita *class chromebook* memiliki *constructor* dan *destructor*, maka *constructor* dan *destructor* class induknya tidak dijalankan.

Bagaimana jika kita ingin *constructor* dan *destructor* *parent class* tetap dijalankan?

Solusinya, kita harus memanggil *constructor* dan *destructor* parent class secara manual dengan *Scope Resolution Operator*, yakni: *parent::__construct()* dan *parent::__destruct()*.

Berikut adalah *modifikasi* kode program kita diatas:

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

```

1 <?php
2 // buat class komputer
3 class komputer {
4     public function __construct() {
5         echo "Constructor dari class komputer <br />";
6     }
7
8     public function __destruct() {
9         echo "Destructor dari class komputer <br />";
10    }
11
12 }
13
14 // turunkan class komputer ke laptop
15 class laptop extends komputer {
16     public function __construct() {
17         parent::__construct();
18         echo "Constructor dari class laptop <br />";
19     }
20
21     public function __destruct() {
22         echo "Destructor dari class laptop <br />";
23         parent::__destruct();
24     }
25
26 // turunkan class laptop ke chromebook
27 class chromebook extends laptop {
28     public function __construct() {
29         parent::__construct();
30         echo "Constructor dari class chromebook <br />";
31     }
32
33     public function __destruct() {
34         echo "Destructor dari class chromebook <br />";
35         parent::__destruct();
36     }
37 }
38
39 // buat objek dari class chromebook (instansiasi)
40 $gadget_baru = new chromebook();
41
42 echo "Belajar OOP PHP <br />";
43 ?>

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Subscribe

Hasil yang kita dapat adalah:

```

1 Constructor dari class komputer
2 Constructor dari class laptop
3 Constructor dari class chromebook
4 Belajar OOP PHP
5 Destructor dari class chromebook
6 Destructor dari class laptop
7 Destructor dari class komputer

```



Dengan memanggil manual perintah `parent::__construct()` dan `parent::__destruct()`, kita bisa menjalankan seluruh *constructor* dan *destructor* dari *parent class*.

Dalam tutorial belajar *Object Oriented Programming (OOP) PHP* kali ini, kita telah mempelajari [cara kerja constructor](#) dan *destructor* jika sebuah class diturunkan dari class lain (*inheritance*).

Constructor dan *destructor* *parent class* akan dijalankan jika *child class* tidak mendefenisikan *constructor* dan *destructor* sendiri. Namun jika *child class* juga memiliki *constructor* dan *descrtuctor*, maka kita harus memanggil *constructor* dan *destructor* *parent class* secara manual.

Dalam tutorial OOP PHP berikutnya, kita akan mempelajari [pengertian static property](#) dan [static method](#) dalam pemrograman objek.

*** Artikel Terkait ***

2 teman menyukai ini.



Sukai Halaman

Pelajari Selengkap



Tutorial Belajar OOP PHP Part 12: Pengertian Static Property dan Static Method

04 Oct 14 | Andre | [Tutorial PHP](#) | 28 Comments

eBook Terbaru DuniaIlkom



[Cara pemesanan eBook & Buku](#)

DuniaIlkom

Daftar eBook DuniaIlkom

[eBook Pascal Uncover](#)

[eBook HTML Uncover](#)

[eBook CSS Uncover](#)

[eBook PHP Uncover](#)

[eBook MySQL Uncover](#)

[eBook JavaScript Uncover](#)

[eBook Bootstrap Uncover](#)

[eBook OOP PHP Uncover](#)

[eBook Laravel Uncover](#)

Dalam lanjutan tutorial belajar OOP PHP kali ini, kita akan mempelajari tentang [Pengertian Static Property dan Static Method](#) dalam Pemrograman Objek, kemudian kita akan melihat cara penggunaannya di dalam PHP.

Pengertian Static Property dan Static Method

Jika di awal tutorial Pemrograman objek PHP ini saya menjelaskan bahwa seluruh *property* dan *method* hanya bisa diakses dari *objek*, maka **static property** dan **static method** adalah pengecualianya.

Static property dan **static method** adalah *property (variabel)* dan *method (function)* yang melekat kepada *class*, bukan kepada objek. Konsep **static property** memang ‘agak keluar’ dari konsep objek sebagai tempat melakukan proses, karena sebenarnya class hanya merupakan ‘*blueprint*’ saja.

Untuk membuat **static property** dan **static method**, kita menambahkan keyword ‘*static*’ setelah penulisan *akses level* property atau method, seperti contoh berikut:

```

1 // static property
2 public static $harga_beli;
3
4 // static method
5 public static function beli_laptop() {
6     //...isi method
7 }
```

Dalam contoh diatas, saya menggunakan hak akses ***public***, tetapi kita juga bisa menggunakan hak akses lain seperti ***private*** dan ***protected*** untuk static property dan static method.

Karena *static property* dan *static method* adalah milik **class**, maka kita tidak perlu membuat objek untuk mengaksesnya, tapi langsung menyebutkan nama class dan menggunakan operator ‘::’, berikut adalah contoh pengaksesan *static property* dan *static method* dari *class laptop*:

```
1 echo laptop::$harga_beli;
2 echo laptop::beli_laptop();
```

Tutorial Cara Penggunaan Static Property dan Static Method

Agar lebih memahami cara penggunaan *static property* dan *static method*, langsung saja kita masuk ke dalam kode program:

```
1 <?php
2 // buat class laptop
3 class laptop {
4     public $merk;
5     public $pemilik;
6
7     // static property
8     public static $harga_beli;
9
10    //static method
11    public static function beli_laptop() {
12        return "Beli Laptop";
13    }
14
15
16    // set static property
17    laptop::$harga_beli=4000000;
18
19    // get static property
20    echo "harga beli : Rp".laptop::$harga_beli;
21
22    echo "<br />";
23
24    // panggil static method
25    echo laptop::beli_laptop();
26 ?>
```

Dalam kode diatas, saya membuat *class laptop* dengan 2 *property* ‘biasa’, 1 *static property* dan 1 *static method*. Perhatikan cara mengakses keduanya tanpa membuat objek.

Cara Mengakses Static Property dan Static Method Dari Class Itu Sendiri

Jika kita menggunakan variabel ***\$this*** untuk mengakses *property* dan *method* ‘normal’ dari dalam class, maka untuk mengakses *static property* dan *static method*, kita menggunakan keyword “***self::***”. Berikut contoh penggunaannya:

List Tutorial DuniaIlkom

- [Tutorial Terbaru DuniaIlkom](#)
- [Tutorial HTML](#)
- [Tutorial PHP](#)
- [Tutorial MySQL](#)
- [Tutorial CSS](#)
- [Tutorial JavaScript](#)
- [Tutorial jQuery](#)
- [Tutorial WordPress](#)
- [Tutorial Pascal](#)
- [Tutorial C](#)
- [Tutorial Python](#)
- [Tutorial Java](#)
- [Tutorial Laravel](#)
- [Membuat Web Online](#)
- [Review Jurusan Kuliah](#)
- [Blog DuniaIlkom](#)

Tutorial Dasar PHP

- [Tutorial PHP Dasar](#)
- [1. Pengertian PHP](#)
- [2. Sejarah PHP](#)
- [3. Menginstall XAMPP](#)
- [4. Menjalankan Apache](#)
- [5. Menjalankan File PHP](#)
- [6. Cara Kerja WebServer](#)
- [7. Input PHP ke HTML](#)
- [8. File php.ini](#)
- [9. Dasar Penulisan PHP](#)
- [10. Penulisan Komentar](#)
- [11. Penulisan Variabel](#)
- [12. Penulisan Konstanta](#)
- [13. Tipe Data Integer](#)
- [14. Tipe Data Float](#)
- [15. Tipe Data String](#)
- [16. Tipe Data Boolean](#)
- [17. Tipe Data Array](#)

```

1 <?php
2 // buat class laptop
3 class laptop {
4     public $merk;
5     public $pemilik;
6
7     // static property
8     public static $harga_beli;
9
10    //static method
11    public static function beli_laptop() {
12        return "Beli laptop seharga Rp".self::$harga_beli;
13    }
14
15    // set static property
16    laptop::$harga_beli=4000000;
17
18    // panggil static method
19    echo laptop::beli_laptop();
20
21 ?>
```

Pada kode program PHP diatas, saya menggunakan perintah `self::$harga_beli`, untuk memanggil *static property* dari dalam *class laptop* itu sendiri.

Cara Mengakses Static Property dan Static Method Parent Class

Untuk class dengan penurunan (*inheritance*), kita bisa menggunakan *keyword* `parent::nama_property` dan `parent::nama_method` untuk mengakses *static property* dan *static method* dari *parent class*.

Misalnya *class laptop* adalah turunan dari *class komputer*, kita bisa menggunakan perintah `parent::beli_komputer()` untuk mengakses *static method* pada *class komputer* dari dalam *class laptop*.

Berikut adalah contoh pengaksesan *static method* milik *parent class*:

```

1 <?php
2 // buat class komputer
3 class komputer {
4
5     // protected static method
6     protected static function beli_komputer(){
7         return "Beli Komputer Baru";
8     }
9
10
11 // turunkan class komputer ke class laptop
12 class laptop extends komputer{
13
14     // private static method
15     private static function beli_laptop(){
16         return "Beli Laptop Baru";
17     }
18
19     // public static method
20     public static function beli_semua(){
21         echo parent::beli_komputer();
22         echo "<br />";
23         echo self::beli_laptop();
24     }
25
26
27
28 // panggil static method
29 laptop::beli_semua();
30
31 // coba panggil private static method
32 // laptop::beli_laptop();
33 // Fatal error: Call to private method laptop::beli_laptop()
34 ?>
```

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom

Pada kode diatas, saya membuat *class komputer* dengan sebuah *static method beli_komputer()*. Method ini memiliki hak akses *protected*, sehingga hanya bisa diakses dari dalam *class* itu sendiri atau dari dalam *class* turunan.

Class komputer kemudian ‘diturunkan’ kepada *class laptop*. Di dalam *class laptop*, saya membuat dua buah *static method*. Static method *beli_laptop()* di set dengan hak akses *private*, sehingga tidak bisa diakses dari luar *class laptop*.

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Email Address

Subscribe

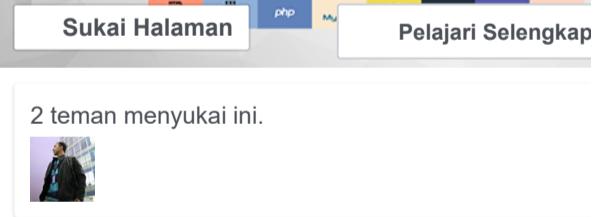
Dalam method *beli_semua()*, saya memanggil method *beli_komputer()* milik *class komputer* menggunakan perintah *parent::beli_komputer()*. Ini adalah cara pemanggilan *static method* milik *parent class*. Kemudian masih di dalam method *beli_semua()*, saya memanggil method *beli_laptop()* dengan perintah *self::beli_laptop()*, karena method ini ada di dalam *class laptop* itu sendiri.

Untuk menguji apakah method *beli_semua()* sukses dijalankan, saya kemudian memanggilnya dengan perintah *laptop::beli_semua()*.

Perhatikan juga pada bagian komentar di akhir kode diatas. Jika kita mencoba memanggil method *laptop::beli_laptop()*, PHP akan mengeluarkan *error* karena method *beli_laptop()* memiliki hak akses *private*, sehingga tidak bisa diakses dari luar class.

Dalam membuat *program berbasis objek*, penggunaan *static property* (dan juga *static method*) sebaiknya dibatasi, karena *static method* cenderung susah dideteksi jika terjadi kesalahan. Namun konsep *property* dan *method* yang melekat kepada *class* ini banyak juga digunakan untuk membuat *design pattern*. Bahkan di dalam framework PHP seperti *laravel*, *static method* merupakan mekanisme utama untuk menjalankan sebagian besar kode program.

Dalam tutorial selanjutnya, kita akan mempelajari [pengertian konstanta class](#) dalam pemrograman objek.



*** Artikel Terkait ***



Tutorial Belajar OOP PHP Part 13: Pengertian Konstanta Class dalam Pemrograman Objek

09 Oct 14 | Andre | [Tutorial PHP](#) | 4 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Dalam tutorial belajar *object oriented programming* PHP kali ini, kita akan mempelajari [pengertian konstanta class](#). Cara pengaksesan *konstanta class* dalam PHP, mirip dengan cara mengakses [static property](#) yang telah kita pelajari dalam tutorial sebelumnya.

Pengertian Konstanta Class

Konstanta Class atau ***class constant*** adalah *konstanta yang berada di dalam class*. Selain memiliki *property* dan *method*, PHP juga membolehkan kita menggunakan konstanta (*constant*) di dalam class.

Sebagaimana sifat *konstanta reguler*, *class constant* juga tidak bisa diubah nilainya ketika sudah didefinisikan. Untuk membuat *class constant* di dalam PHP, kita menggunakan perintah: **const**.



Hampir semua sifat *konstanta reguler* juga sama di dalam *konstanta class*.

Pembahasan tentang konstanta reguler telah kita bahas dalam [Tutorial PHP](#)

Dasar: Pengertian Konstanta dan Cara Penulisan Konstanta PHP

Berikut adalah contoh kode program pembuatan *constant* di dalam *class*:

```
1 class nama_class {
2     const NAMA_KONSTANTA = nilai_konstanta;
3 }
```



Penulisan nama *konstanta* dengan huruf besar bukan keharusan, namun lebih kepada kebiasaan programmer PHP agar mudah dibedakan dengan *variabel* yang umumnya ditulis dengan huruf kecil.

Di dalam PHP, *class constant* seolah-olah berprilaku sebagai *static property*. *Class constant* juga terikat kepada *class*, bukan *objek*. Oleh karena itu, untuk mengakses nilai *konstanta*, kita menggunakan operator yang sama seperti *static property*, yakni menggunakan *double colon* '::'.

Jika kita memiliki *class laptop* dan konstanta MERK, maka cara mengaksesnya adalah sebagai berikut:

```
1 | laptop::MERK;
```

Cara Penulisan Konstanta Class dalam PHP

Untuk melihat cara penulisan dan penggunaan *konstanta class*, kita akan langsung menggunakan kode program. Berikut adalah contoh *class laptop* dengan sebuah konstanta **DOLLAR**:

```
1 | <?php
2 | // buat class laptop
3 | class laptop {
4 |     // buat konstanta
5 |     const DOLLAR = '12000';
6 |
7 |
8 |     // panggil konstanta class
9 |     echo "Harga dollar saat ini = Rp. ".laptop::DOLLAR;
10 |    // hasil: Harga dollar saat ini = Rp. 12000
11 | ?>
```

Perhatikan bahwa untuk mengakses *class constant DOLLAR* milik *class laptop*, kita menggunakan perintah *laptop::DOLLAR*.

Selain mengakses konstanta dengan menggunakan *nama class*, PHP juga memiliki cara lain, yakni dengan mengaksesnya dari *objek*. Fitur ini hanya bisa digunakan untuk PHP versi 5.3 keatas. Berikut contohnya:

```
1 | <?php
2 | // buat class laptop
3 | class laptop {
4 |     // buat konstanta
5 |     const DOLLAR = '12000';
6 |
7 |
8 |     // buat objek dari class laptop (instansiasi)
9 |     $laptop_baru = new laptop();
10 |
11 |     // panggil konstanta class
12 |     echo "Harga dollar saat ini = Rp ".$laptop_baru::DOLLAR;
13 |     // hasil: Harga dollar saat ini = Rp. 12000
14 | ?>
```

Dalam kode diatas, kita mengakses nilai *kontanta class* dari *objek \$laptop_baru* menggunakan perintah *\$laptop_baru::DOLLAR*.

PHP versi 5.3 keatas juga membolehkan pemanggilan *property* dengan *nama class yang berada di dalam variabel*. Berikut contohnya:

List Tutorial DuniaIlkom

- [Tutorial Terbaru DuniaIlkom](#)
- [Tutorial HTML](#)
- [Tutorial PHP](#)
- [Tutorial MySQL](#)
- [Tutorial CSS](#)
- [Tutorial JavaScript](#)
- [Tutorial jQuery](#)
- [Tutorial WordPress](#)
- [Tutorial Pascal](#)
- [Tutorial C](#)
- [Tutorial Python](#)
- [Tutorial Java](#)
- [Tutorial Laravel](#)
- [Membuat Web Online](#)
- [Review Jurusan Kuliah](#)
- [Blog DuniaIlkom](#)

Tutorial Dasar PHP

- [Tutorial PHP Dasar](#)
- [1. Pengertian PHP](#)
- [2. Sejarah PHP](#)
- [3. Menginstall XAMPP](#)
- [4. Menjalankan Apache](#)
- [5. Menjalankan File PHP](#)
- [6. Cara Kerja WebServer](#)
- [7. Input PHP ke HTML](#)
- [8. File php.ini](#)
- [9. Dasar Penulisan PHP](#)
- [10. Penulisan Komentar](#)
- [11. Penulisan Variabel](#)
- [12. Penulisan Konstanta](#)
- [13. Tipe Data Integer](#)
- [14. Tipe Data Float](#)
- [15. Tipe Data String](#)
- [16. Tipe Data Boolean](#)
- [17. Tipe Data Array](#)

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat konstanta
6     const DOLLAR = '12000';
7 }
8
9 // buat variabel dengan nama class
10 $nama = "laptop";
11
12 // panggil konstanta class
13 echo "Harga dollar saat ini = Rp. ".$nama::DOLLAR;
14 // hasil: Harga dollar saat ini = Rp. 12000
15 ?>

```

Pada kode program diatas, saya tidak menggunakan *objek*, tetapi membuat variabel **\$nama** dan memberikannya nilai **laptop**. Karena nama class kita juga adalah *laptop*, maka PHP membolehkan pemanggilan kosntanta **DOLLAR** dengan **\$nama::DOLLAR**. Nama *variabel* yang digunakan boleh bebas, selama nilainya cocok dengan *nama class* tempat *konstanta* itu berada.

Cara Mengakses Konstanta Class dari dalam Class itu Sendiri

Untuk mengakses *class constant* dari *dalam class itu sendiri*, PHP menggunakan cara yang sama dengan *static property*, yaitu dengan perintah ***self::nama_konstanta***. Berikut contohnya:

```

1 <?php
2 // buat class laptop
3 class laptop {
4
5     // buat konstanta
6     const DOLLAR = '12000';
7
8     // buat method
9     public function beli_laptop($harga) {
10         return "Beli Komputer Baru, Rp. ".$harga*self::DOLLAR;
11     }
12 }
13
14 // buat objek dari class laptop (instansiasi)
15 $laptop_baru=new laptop();
16
17 echo $laptop_baru->beli_laptop(400);
18 // hasil: Beli Komputer Baru, Rp. 4800000
19 ?>

```

Saya membuat *class laptop* dengan sebuah method *beli_laptop()*. Method *beli_laptop()* digunakan untuk menghitung harga *laptop* dengan mengalikan *konstanta class DOLLAR* dengan parameter *\$harga*. Perhatikan bahwa kita mengakses *class constant* dengan perintah ***self::DOLLAR***.

Cara Mengakses Konstanta Class milik Parent Class

Pewarisan *class* (*class inheritance*) dari sebuah *class* kedalam *class* lain, juga akan menurunkan konstanta. Jika kebetulan *class* yang diturunkan (*child class*) memiliki nama *konstanta* yang sama dengan *parent class*, konstanta tersebut akan *'terimpas'*.

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

Subscribe

PHP menggunakan operator `parent::nama_konstanta` untuk mengakses konstanta milik *parent class*.

Agar lebih mudah, berikut adalah contoh kode program penggunaan operator `parent::nama_konstanta`:

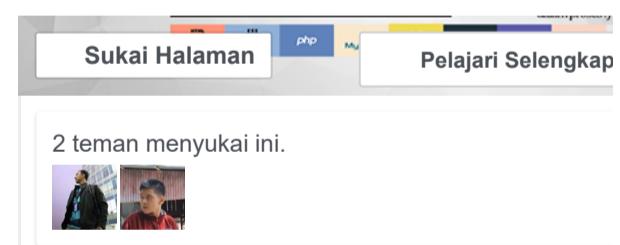
```

1 <?php
2 // buat class komputer
3 class komputer {
4     // buat konstanta class komputer
5     const DOLLAR = '11000';
6 }
7
8 // turunkan class komputer ke class laptop
9 class laptop extends komputer {
10    // buat konstanta class laptop
11    const DOLLAR = '12000';
12
13   // buat method dengan konstanta class komputer
14   public function beli_komputer($harga){
15       return "Beli Komputer Baru, Rp ." . $harga * parent::DOLLAR;
16   }
17
18   // buat method dengan konstanta class laptop
19   public function beli_laptop($harga){
20       return "Beli Komputer Baru, Rp ." . $harga * self::DOLLAR;
21   }
22 }
23
24 // buat objek dari class laptop (instansiasi)
25 $laptop_baru=new laptop();
26
27 echo $laptop_baru->beli_laptop(400);
28 echo "<br />";
29 echo $laptop_baru->beli_komputer(400);
30 ?>
```

Saya membuat konstanta **DOLLAR** di dalam *class komputer*. *Class komputer* kemudian diturunkan ke dalam *class laptop*. Di dalam *class laptop*, saya mendefenisikan kembali konstanta **DOLLAR**. Karena kedua konstanta ini memiliki nama yang sama, maka saya harus menggunakan perintah `parent::DOLLAR` untuk memanggil konstanta **DOLLAR** miliki *class komputer*.

Dalam *tutorial OOP PHP* kali ini kita telah mempelajari tentang [pengertian konstanta class](#) dan cara penggunaan konstanta class dalam PHP. Walaupun konstanta class jarang digunakan di dalam pemograman umum, namun fitur yang ditawarkan mungkin bisa membantu untuk penyelesaian kasus-kasus tertentu.

Dalam *tutorial OOP PHP* berikutnya, kita akan mempelajari [Pengertian Final Method dan Final Class Pemrograman Objek PHP](#).



eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoloading* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).



Home | Tutorial PHP | Tutorial Belajar OOP PHP Part 14: Pengertian Final Method dan Final Class Pemrograman Objek

Tutorial Belajar OOP PHP Part 14: Pengertian Final Method dan Final Class Pemrograman Objek

09 Oct 14 | Andre | Tutorial PHP | 6 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Setelah sebelumnya mempelajari tentang [class constant](#), dalam tutorial belajar OOP PHP kali ini kita akan membahas [pengertian final method dan final class dalam pemrograman objek PHP](#).

Pengertian Final Method dan Final Class

Dalam membuat desain *class*, kita sering menurunkan sebuah *class* kepada *class* lain, atau yang dikenal dengan *inheritance/pewarisan*. Pemrograman objek juga membolehkan kita untuk '*menimpa*' *method* milik *parent class* dengan *method* milik *child class*. Proses *menimpa* *method* atau dikenal dengan istilah ***overridden method*** ini dilakukan dengan cara membuat *nama method* yang sama dengan *nama method* yang ada di dalam *parent class*.



Mengenai ***overridden method***, telah kita bahas dalam tutorial belajar oop php: [cara mengakses property dan method parent class](#).

Bagaimana jika kita menginginkan sebuah mekanisme untuk **milarang** *class anak* untuk membuat *method* yang akan menimpa *method* *class induk*? Atau bahkan milarang sebuah *class* untuk diturunkan sama sekali? Untuk keperluan ini, pemrograman objek PHP menggunakan keyword: **final**.



Dengan menambahkan keyword ***final*** kepada sebuah *method*, maka *method* tersebut tidak dapat didefinisikan ulang di dalam *child class*. Dan jika sebuah *class* ditambahkan keyword ***final***, maka *class* tersebut tidak bisa diturunkan sama sekali. Inilah *pengertian* dari ***final method*** dan ***final class***.



Dalam PHP, tidak dikenal istilah ***final property***, sehingga tidak ada mekanisme untuk membatasi *class anak* untuk menimpa nilai *property* dari *class induk*.

Untuk membuat ***final method***, kita tinggal menambahkan kata ***final*** sebelum keyword hak akses, seperti berikut ini:

```
1 | final public function nama_method(){
2 |     //... isi method
3 | }
```

Sedangkan untuk membuat ***final class***, kita menambahkan kata ***final*** sebelum nama *class*, seperti contoh berikut ini:

```
1 | final class nama_class {
2 |     //... isi class
3 | }
```

Cara Penggunaan Final Method dan Final Class dalam PHP

Untuk membahas cara menggunakan ***final method*** dan ***final class***, pertama kali mari kita lihat cara penulisan dan penggunaan ***final method*** dalam PHP.

Dalam contoh berikut, saya membuat *class komputer* dengan sebuah ***final method***, kemudian saya mencoba menimpa *method* ini dari *class laptop*:

```
1 | <?php
2 | class komputer{
3 |     final public function lihat_spec(){
4 |         return "Lihat Spesifikasi Komputer";
5 |     }
6 |
7 |
8 | class laptop extends komputer{
9 |     public function lihat_spec(){
10 |         return "Lihat Spesifikasi Laptop";
11 |     }
12 |
13 |
14 | $laptop_baru=new laptop();
15 | // Fatal error: Cannot override final method
16 | // komputer::lihat_spec()
17 | ?>
```

Pada kode program diatas, *class komputer* memiliki *method lihat_spec()* yang di-set sebagai ***final***. Kemudian *class komputer* ini saya turunkan kepada *class laptop*. Di dalam *class laptop*, saya mendefenisikan ulang *method lihat_spec()*, sehingga *method lihat_spec()* milik *class komputer* akan tertimpa oleh *method lihat_spec()* milik *class laptop*.

Saat saya membuat objek ***\$laptop_baru*** dari *class laptop*, PHP akan mengeluarkan error: *Cannot override final method komputer::lihat_spec()*. Error ini menjelaskan bahwa kita tidak bisa menimpa *method lihat_spec()* milik *class komputer*, karena *method tersebut telah di set sebagai ***final****.

List Tutorial DuniaIlkom

Tutorial Terbaru DuniaIlkom

Tutorial HTML

Tutorial PHP

Tutorial MySQL

Tutorial CSS

Tutorial JavaScript

Tutorial jQuery

Tutorial WordPress

Tutorial Pascal

Tutorial C

Tutorial Python

Tutorial Java

Tutorial Laravel

Membuat Web Online

Review Jurusan Kuliah

Blog DuniaIlkom

Tutorial Dasar PHP

Tutorial PHP Dasar

1. Pengertian PHP

2. Sejarah PHP

3. Menginstall XAMPP

4. Menjalankan Apache

5. Menjalankan File PHP

6. Cara Kerja WebServer

7. Input PHP ke HTML

8. File php.ini

9. Dasar Penulisan PHP

10. Penulisan Komentar

11. Penulisan Variabel

12. Penulisan Konstanta

13. Tipe Data Integer

14. Tipe Data Float

15. Tipe Data String

16. Tipe Data Boolean

17. Tipe Data Array

Jika sebuah *final method* tidak bisa ditimpa nilainya, maka sebuah **final class** tidak bisa diturunkan kepada *class* lain. Sebagai contoh kedua kita, berikut adalah kode PHP dimana saya mencoba menurunkannya:

```

1 <?php
2  final class komputer{
3      function lihat_spec(){
4          return "Lihat Spesifikasi Komputer";
5      }
6  }
7
8  class laptop extends komputer{
9  }
10
11 $laptop_baru=new laptop();
12 // Fatal error: Class laptop may not inherit
13 // from final class (komputer)
14 ?>

```

Saya membuat sebuah *final class komputer*, kemudian mencoba menurunkannya kepada *class laptop*. Saat proses pembuatan objek: `$laptop_baru=new laptop()`, PHP akan komplain dan mengeluarkan error: *Fatal error: Class laptop may not inherit from final class (komputer)*. Dari hasil error yang didapat, PHP melarang kita untuk menurunkan *class komputer*, karena telah di set sebagai *final class*.

Sebagai contoh ketiga, walaupun PHP memang tidak memiliki *final property*, namun mari kita coba membuatnya:

```

1 <?php
2  class komputer {
3      final $merk;
4  }
5
6  $laptop_baru=new komputer();
7  // Fatal error: Cannot declare property komputer::$merk final,
8  // the final modifier is allowed only for methods and classes
9 ?>

```

Dari hasil error yang didapat, ternyata PHP memang tidak mendukung *final property* :)

Final method dan **final class** bisa digunakan untuk membuat desain *class* yang terstruktur. Dalam tutorial OOP PHP selanjutnya, kita akan membahas tentang **Pengertian Abstract Class dan Abstract Method** dalam Pemrograman Objek PHP.

*** Artikel Terkait ***

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^



[Home](#) | [Tutorial PHP](#) | Tutorial Belajar OOP PHP Part 15: Pengertian Abstract Class dan Abstract Method PHP

Tutorial Belajar OOP PHP Part 15: Pengertian Abstract Class dan Abstract Method PHP

10 Oct 14 | Andre | [Tutorial PHP](#) | 19 Comments

Dalam tutorial belajar OOP PHP kali ini kita akan membahas tentang [pengertian abstract class dan abstract method dalam pemrograman objek](#), serta cara pembuatan *abstract class* dan *abstract method* dengan PHP.

Pengertian Abstract Class dan Abstract Method

Abstract Class adalah sebuah class yang tidak bisa di-*instansiasi* (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi class turunannya. Di dalam *abstract class* umumnya akan memiliki *abstract method*.

Abstract Method adalah sebuah ‘*method dasar*’ yang harus diimplementasikan ulang di dalam class anak (*child class*). *Abstract method* ditulis tanpa isi dari *method*, melainkan hanya ‘**signature**’-nya saja. **Signature** dari sebuah *method* adalah bagian *method* yang terdiri dari nama *method* dan parameternya (jika ada).

Abstract class digunakan di dalam **inheritance** (pewarisan class) untuk ‘memaksakan’ implementasi *method* yang sama bagi seluruh class yang diturunkan dari *abstract class*. *Abstract class* digunakan untuk membuat struktur logika penurunan di dalam pemrograman objek.

Konsep *abstract class* dan *abstract method* akan lebih mudah dipahami dengan menggunakan contoh.

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Misalkan kita ingin membuat *class* yang terdiri dari berbagai jenis komputer seperti *class laptop*, *class PC*, *class netbook*, dan lain-lain. Seluruh *class* ini tentunya memiliki sifat-sifat *komputer*, seperti *memiliki spesifikasi*, *memiliki processor*, dan *membutuhkan listrik*. Dalam implementasinya, kita bisa membuat seluruh *class* ini diturunkan dari *class komputer*.

Agar lebih seragam, kita ingin seluruh *class* yang diturunkan dari *class komputer*, memiliki method yang ‘pasti’ ada dalam setiap *class anak*. Setiap komputer tentunya memiliki spesifikasi, sehingga kita ingin setiap *class* yang diturunkan dari *class komputer* memiliki method *lihat_spec()*.

Bagaimana caranya ‘*memaksa*’ setiap *class* agar memiliki method *lihat_spec()*? Untuk kebutuhan ini, kita bisa *membuat class komputer sebagai abstract class*, dan method *lihat_spec()* sebagai *abstract method*.

Lebih lanjut, *abstract method* tidak hanya membuat setiap *class* memiliki method *lihat_spec()*, tetapi memaksa setiap method mengimplementasikan method *lihat_spec()* dengan isi method di serahkan kepada masing-masing *class*. Tentunya spesifikasi *class laptop* akan berbeda dengan spesifikasi *class PC*.

Abstract class memiliki aturan yang membedakannya dengan *class* biasa. Kita akan membahas aturan-aturan tersebut dengan menggunakan contoh program PHP.

Cara Membuat Abstract Class

Karena kita ingin membuat *class komputer* sebagai *abstract class*, maka berikut adalah cara penulisannya di dalam PHP:

```
1 <?php
2 abstract class komputer {
3     // isi dari class komputer
4 }
5 ?>
```

Untuk membuat *abstract class* di dalam PHP, kita tinggal menambahkan keyword **abstract** sebelum nama *class*. Sebuah *abstract class* bisa memiliki *property* dan *method* biasa layaknya sebuah *class ‘normal’*, namun juga bisa memiliki *abstract method*.

Cara Membuat Abstract Method

Jika sebuah *method* dinyatakan sebagai *abstract method*, maka kita tidak perlu membuat isi methodnya, tetapi hanya *signature* dari method tersebut. *Signature* terdiri dari nama method dan parameteranya (jika ada) seperti contoh berikut:

```
1 abstract public function lihat_spec();
2 abstract public function lihat_spec($merk);
```

Kenapa kita tidak perlu membuat isi dari *method*? Ini karena jika sebuah *method* dinyatakan sebagai *abstract method*, isi dari method tersebut akan dibuat dalam *class turunan*. *Abstract method harus berada di dalam abstract class*.

Sebagai contoh, berikut adalah cara penulisan *abstract method lihat_spec()* di dalam *abstract class komputer*:

```
1 <?php
2 abstract class komputer {
3     abstract public function lihat_spec();
4 }
5 ?>
```

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array

Perhatikan bahwa kita tidak perlu (baca:tidak bisa) membuat isi dari *abstract method*.

Abstract Class Tidak Bisa Diinstansiasi

Sesuai dengan sifatnya, kita tidak bisa membuat objek dari *abstract class*. *Abstract class* digunakan hanya sebagai '*blueprint*' untuk class-class lain, bukan untuk digunakan langsung.

```

1 abstract class komputer {
2     abstract public function lihat_spec($pemilik);
3 }
4
5 $komputer_baru=new komputer();
6 // Fatal error: Cannot instantiate abstract class komputer
7
8 ?>

```

Error diatas terjadi karena kita mencoba membuat objek dari *abstract class*. Untuk menggunakan class komputer, kita harus menurunkannya kepada class lain.

Abstract Class Bisa Memiliki Property dan Method 'biasa'

Jika sebuah class dinyatakan sebagai *abstract class*, class tersebut juga bisa memiliki *property* dan *method* 'normal'. Namun kita hanya bisa mengakses *property* dan *method* ini dari class turunan, karena *abstract class* tidak bisa diinstansiasi.

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4
5     // buat abstract method
6     abstract public function lihat_spec($pemilik);
7
8     // buat method 'biasa'
9     public function hidupkan_komputer(){
10        echo "Hidupkan Komputer";
11    }
12 }
13 ?>

```

Class Turunan Harus Mengimplementasikan Abstract Method

Jika sebuah class diturunkan dari *abstract class*, maka class tersebut harus *membuat ulang seluruh abstract method yang terdapat dalam abstract class*, dan juga harus sesuai dengan *signature*-nya.

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assigment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP

Berlangganan Artikel DuniaIlkom

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function lihat_spec();
6 }
7
8 class laptop extends komputer{
9     public function beli_laptop(){
10        return "Beli Laptop...";
11    }
12 }
13
14 // buat objek dari class laptop
15 $laptop_baru = new laptop();
16
17 // Fatal error: Class laptop contains 1 abstract method
18 // and must therefore be declared abstract or implement
19 // the remaining methods (komputer::lihat_spec)
20 ?>

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

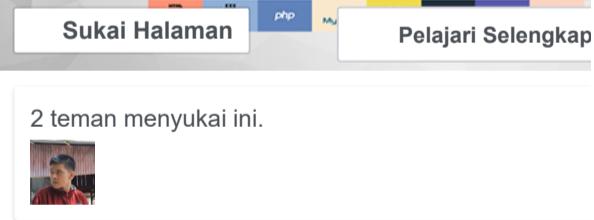
Join 3,082 other subscribers

Dalam contoh diatas, saya menurunkan *class komputer* kepada *class laptop*. Karena *class komputer* adalah *abstract class* dan memiliki *abstract method lihat_spec()*, maka di dalam *class laptop* kita harus membuat ulang method *lihat_spec()*. Jika tidak, akan terjadi kode error seperti diatas. Untuk mengatasinya, mari kita revisi kode diatas menjadi berikut ini :

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function lihat_spec();
6 }
7
8 class laptop extends komputer{
9
10 // implementasi abstract method
11     public function lihat_spec(){
12         return "Lihat Spec Laptop...";
13     }
14
15 // method 'biasa'
16     public function beli_laptop(){
17         return "Beli Laptop...";
18     }
19 }
20
21 // buat objek dari class laptop
22 $laptop_baru = new laptop();
23 echo $laptop_baru->lihat_spec();
24 // Lihat Spec Laptop...
25
26 echo "<br />";
27
28 echo $laptop_baru->beli_laptop();
29 // Beli Laptop...
30 ?>

```



Dalam kode diatas, *method lihat_spec()* telah kita implementasikan di dalam *class laptop*. Fitur inilah yang menjadi fungsi dari *abstract method*, yakni '*memaksa*' setiap *class turunan* untuk memiliki *method lihat_spec()*.

Implementasi dari *abstract method*, juga harus sesuai dengan signaturenya, yakni *nama method* beserta *parameter*. Jika kita membuat *abstract method lihat_spec(\$merk)*, maka di dalam *class turunan*, kita juga harus membuat *\$merk* sebagai parameter method. Jika tidak, maka PHP akan menghasilkan error sebagai berikut:

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function lihat_spec($pemilik);
6 }
7
8 class laptop extends komputer{
9     public function lihat_spec(){
10     return "Lihat Spec Laptop...";
11 }
12 }
13
14 // buat objek dari class laptop
15 $laptop_baru = new laptop();
16
17 // Fatal error: Declaration of laptop::lihat_spec()
18 // must be compatible with komputer::lihat_spec($pemilik)
19 ?>

```

Abstract Class Bisa Memiliki Static Method

Salah satu fitur '*khusus*' untuk *abstract class* di dalam PHP, adalah: *abstract class* bisa memiliki *static method*. Berikut contohnya:

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function lihat_spec($pemilik);
6
7     public static function hidupkan_komputer(){
8         echo "Hidupkan Komputer";
9     }
10 }
11
13 echo komputer::hidupkan_komputer();
14 // Hidupkan Komputer
15 ?>

```

Fungsi Abstract Class dan Abstract Method

Abstract class dan *abstract method* berfungsi untuk membuat '*kerangka*' bagi seluruh class dibawahnya. Seperti contoh-contoh kita diatas, setiap class yang diturunkan dari *class komputer*, '*pasti*' akan memiliki method *lihat_spec()*.

Dalam tutorial ini saya menyederhanakan contoh kode program dengan hanya 1 *abstract method*. Kita bisa membuat beberapa *abstract method* di dalam *abstract class*, seperti contoh berikut:

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function lihat_spec();
6     abstract public function lihat_processor();
7     abstract public function lihat_harddisk();
8     abstract public function lihat_pemilik();
9 }
10
11 class laptop extends komputer{
12     // .. isi class laptop
13 }
14
15 class pc extends komputer{
16     // .. isi class pc
17 }
18
19 class netbook extends komputer{
20     // .. isi class netbook
21 }
22 ?>

```

Dengan membuat *class komputer* sebagai *abstract*, maka kita bisa menebak bahwa di dalam *class laptop*, *class pc* dan *class netbook*, pasti memiliki method *lihat_spec()*, *lihat_processor()*, *lihat_harddisk()* dan *lihat_pemilik()*. Dengan demikian, kita bisa membuat program yang lebih terstruktur.

Dalam tutorial OOP PHP berikutnya, kita akan membahas tentang [Object Interfaces](#) atau dikenal dengan sebutan *Interface* saja. *Interface* di dalam pemrograman objek sangat mirip dengan *abstract class*. Kita akan membahas pengertian, cara penggunaan serta perbedaan Object Interface dengan Abstract Class.

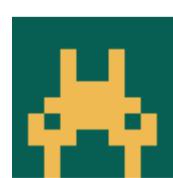
eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoload* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke eBook OOP PHP Uncover DuniaIlkom.

*** Artikel Terkait ***

Tags: [Belajar Objek PHP](#), [Belajar OOP PHP](#), [OOP PHP](#), [Pemrograman Berbasis Objek](#), [Pengertian Abstract Class PHP](#), [Pengertian Abstract Method PHP](#), [Pengertian Array](#), [Tutorial OOP PHP](#)

19 COMMENTS



fjakldfjsa

23 Jan 15

Coding yang paling terakhir itu error gan ?
Fatal error: Class laptop contains 4 abstract methods and must therefore be declared abstract or implement the remaining methods

[Reply](#)

Andre Author



24 Jan 15



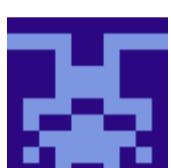
Memang akan error gan, karena di dalam ke-3 child class (laptop, pc dan netbook) saya tidak membuat implementasi dari masing-masing abstract method-nya.

Kalau agan telah memahami maksud dan pengertian dari abstract class dan abstract method, saya yakin agan bisa tahu dimana letak kesalahannya :)

[Reply](#)**Anonymous**

06 Mar 15

Baru kali ini Saya meliha situs yang menjelaskan permasalahan yang secara terperinci ...!

[Reply](#)**ERWIN JAYA SANTOSA**

03 Jun 15

wah.. jelas sekali... terimkasih byk

[Reply](#)**Mahrizal**

28 Jan 16

Terima kasih mas sebelumnya saya belum paham bener apa itu abstract class kenapa harus pakai abstract class
tapi setelah baca artikel ini saya jadi paham yaitu untuk memaksakan class turunan menggunakan method yang ada di abstract class

[Reply](#)**Andre** Author

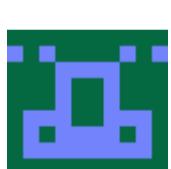
28 Jan 16

Yup benar mas, itulah konsep dasar dari sebuah abstract class di OOP :)

[Reply](#)**ardsc**

07 Oct 16

terimakasih banyak

[Reply](#)**minyak**

29 Nov 16

terus untuk apa, dan apa fungsinya abstrac method dan class itu kalo di kelas turunannya harus dibikin method lg.. ?
bukankah lebih baik bikin class biasa terus bikin 2 method saja.. tanpa ekstend dr turunan abstrac classnya ?

[Reply](#)**Andre** Author

29 Nov 16

Fungsi abstract class ini baru perlu untuk aplikasi yang rumit dan besar. Agan akan ketemu konsep ini ketika mempelajari "design pattern", yakni materi yang khusus membahas tentang implementasi konsep OOP.





[Home](#) | [Tutorial PHP](#) | Tutorial Belajar OOP PHP Part 16: Pengertian Object Interface Dalam Pemrograman Berbasis Objek

Tutorial Belajar OOP PHP Part 16: Pengertian Object Interface Dalam Pemrograman Berbasis Objek

10 Oct 14 | [Andre](#) | [Tutorial PHP](#) | [19 Comments](#)

Setelah dalam tutorial sebelumnya kita mempelajari tentang [abstract class](#) dan [abstract method](#), dalam tutorial belajar OOP PHP kali ini kita akan membahas tentang [Objek Interface dalam pemrograman berbasis objek](#). **Object Interface** ini sangat mirip dengan [Abstract class](#).

Pengertian Object Interface

Secara sederhana, **Object Interface** adalah sebuah '*kontrak*' atau perjanjian *implementasi method*.

Bagi *class* yang menggunakan *object interface*, *class* tersebut harus mengimplementasikan ulang seluruh *method* yang ada di dalam *interface*. Dalam pemrograman objek, penyebutan *object interface* sering disingkat dengan '*Interface*' saja.

Jika anda telah mempelajari [abstract class](#), maka *interface* bisa dikatakan sebagai bentuk lain dari [abstract class](#). Walaupun secara konsep teoritis dan tujuan penggunaannya berbeda.

Sama seperti [abstract class](#), *interface* juga hanya berisi *signature* dari *method*, yakni hanya nama *method* dan *parameternya* saja (jika ada). Isi dari *method* akan dibuat ulang di dalam *class* yang menggunakan *interface*.

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

[eBook Pascal Uncover](#)

[eBook HTML Uncover](#)

[eBook CSS Uncover](#)

[eBook PHP Uncover](#)

[eBook MySQL Uncover](#)

[eBook JavaScript Uncover](#)

[eBook Bootstrap Uncover](#)

[eBook OOP PHP Uncover](#)

[eBook Laravel Uncover](#)

Jika kita menganggap *abstract class* sebagai ‘kerangka’ atau ‘blue print’ dari class-class lain, maka *interface* adalah implementasi method yang harus ‘tersedia’ dalam sebuah objek. *Interface* tidak bisa disebut sebagai ‘kerangka’ class.

Menyambung analogi kita tentang *class komputer*, *interface* bisa dicontohkan dengan ‘mouse’, atau ‘keyboard’. Di dalam *interface mouse*, kita bisa membuat method seperti *klik_kiri()*, *klik_kanan()*, dan *double_klik()*. Jika class laptop ‘menggunakan’ *interface mouse*, maka class tersebut harus membuat ulang method *klik_kiri()*, *klik_kanan()*, dan *double_klik()*.

Cara Membuat Interface dalam PHP

Untuk membuat *Interface* di dalam PHP, kita menulisnya mirip seperti membuat *class*, tetapi menggunakan keyword **interface**, seperti contoh berikut:

```
1 <?php
2 interface mouse
3 {
4     //...isi dari interface mouse
5 }
6 ?>
```

Isi dari *interface* adalah *signature method* (*nama dan parameter method*):

```
1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5     public function scroll();
6     public function double_klik();
7 }
8 ?>
```

Untuk menggunakan method kedalam class, kita menggunakan keyword **implements**, seperti contoh berikut:

```
1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 class laptop implements mouse{
8     //... isi dari class laptop
9 }
10
11 class pc implements mouse{
12     //... isi dari class pc
13 }
14 ?>
```

Interface adalah ‘*perjanjian method*’, dimana jika sebuah *class* menggunakan *interface*, maka di dalam class tersebut harus tersedia *implementasi* dari *method tersebut*.

Jika di dalam *interface mouse* terdapat *signature method* *klik_kanan()*, maka di dalam class laptop yang menggunakan *interface mouse*, harus terdapat method *klik_kanan()*. Berikut contoh kode PHPnya:

List Tutorial DuniaIlkom

- Tutorial Terbaru DuniaIlkom
- Tutorial HTML
- Tutorial PHP
- Tutorial MySQL
- Tutorial CSS
- Tutorial JavaScript
- Tutorial jQuery
- Tutorial WordPress
- Tutorial Pascal
- Tutorial C
- Tutorial Python
- Tutorial Java
- Tutorial Laravel
- Membuat Web Online
- Review Jurusan Kuliah
- Blog DuniaIlkom

Tutorial Dasar PHP

- Tutorial PHP Dasar
- 1. Pengertian PHP
- 2. Sejarah PHP
- 3. Menginstall XAMPP
- 4. Menjalankan Apache
- 5. Menjalankan File PHP
- 6. Cara Kerja WebServer
- 7. Input PHP ke HTML
- 8. File php.ini
- 9. Dasar Penulisan PHP
- 10. Penulisan Komentar
- 11. Penulisan Variabel
- 12. Penulisan Konstanta
- 13. Tipe Data Integer
- 14. Tipe Data Float
- 15. Tipe Data String
- 16. Tipe Data Boolean
- 17. Tipe Data Array

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 class laptop implements mouse{
8     public function klik_kanan(){
9         return "Klik Kanan...";
10    }
11    public function klik_kiri(){
12        return "Klik Kiri...";
13    }
14 }
15
16 $laptop_baru = new laptop();
17 echo $laptop_baru->klik_kanan();
18 // Klik Kanan...
19 ?>

```

Apabila kita tidak membuat ulang salah satu method yang ada di interface, PHP akan mengeluarkan *error*:

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 class laptop implements mouse{
8     public function klik_kanan(){
9         return "Klik Kanan...";
10    }
11 }
12
13 $laptop_baru = new laptop();
14 // Fatal error: Class laptop contains 1 abstract method
15 // and must therefore be declared abstract
16 // or implement the remaining methods (mouse::klik_kiri)
17 ?>

```

Method Interface Harus di set Sebagai Public

Sesuai dengan tujuannya untuk membuat *interface/antar muka* bagi *class*, method di dalam perancangan *interface* harus memiliki *hak akses public*, atau tidak ditulis sama sekali (dimana PHP akan menganggapnya sebagai *public*). Jika kita mengubah hak akses method di dalam *interface* menjadi *private* atau *protected*, PHP akan mengeluarkan *error*:

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     protected function klik_kiri();
5 }
6
7 class laptop implements mouse{
8     public function klik_kanan(){
9         return "Klik Kanan...";
10    }
11    public function klik_kiri(){
12        return "Klik Kiri...";
13    }
14 }
15
16 $laptop_baru = new laptop();
17 // Fatal error: Access type for interface
18 // method mouse::klik_kiri() must be omitted
19 ?>

```

Di dalam *class* yang menggunakan *interface*, *method* yang berasal dari *interface* juga harus memiliki hak akses *public*. Kita tidak bisa mengubahnya menjadi *protected* atau *private*.

18. Pengertian Operand

19. Fungsi var_dump()

20. Operator Aritmatika

21. Operator String

22. Operator Logika

23. Perbandingan

24. Operator Increment

25. Assignment PHP

26. Operator Bitwise

27. Operator Assigment

28. Type Casting

29. Struktur Logika IF

30. Struktur ELSE

31. Logika ELSE-IF

32. Struktur Switch

33. Perulangan For

34. Perulangan While

35. Do-While

36. Perintah Break

37. Perintah Continue

38. Perulangan Foreach

39. Pengertian Function

40. Penulisan Function

41. Variabel Scope

42. Argumen Function

43. Default Parameter

44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 class laptop implements mouse{
8     public function klik_kanan(){
9         return "Klik Kanan...";
10    }
11
12    protected function klik_kiri(){
13        return "Klik Kiri...";
14    }
15 }
16
17 $laptop_baru = new laptop();
18 // Fatal error: Access level to laptop::klik_kiri()
19 // must be public (as in class mouse)
20 ?>

```

Dalam contoh diatas, saya mengubah hak akses *method klik_kiri()* menjadi *protected* di dalam *class laptop*. Hal ini akan menghasilkan *error*.

Interface bisa di Turunkan (Inherit)

Di dalam PHP, *interface* bisa diturunkan kedalam *interface* lain. Prosesnya mirip dengan penurunan *class*, yakni dengan menggunakan kata kunci *extends*:

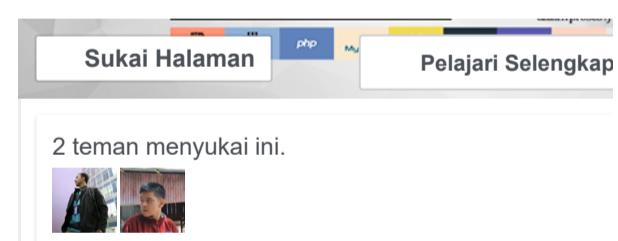
```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 interface mouse_gaming extends mouse{
8     public function ubah_dpi();
9 }
10
11 class laptop implements mouse_gaming{
12     public function klik_kanan(){
13         return "Klik Kanan...";
14     }
15
16     public function klik_kiri(){
17         return "Klik Kiri...";
18     }
19
20     public function ubah_dpi(){
21         return "Ubah settingan DPI mouse";
22     }
23 }
24
25 $laptop_baru = new laptop();
26 echo $laptop_baru->ubah_dpi();
27 // Ubah settingan DPI mouse
28 ?>

```

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers



Interface Bisa Memiliki Konstanta

Dalam PHP, Interface bisa memiliki konstanta . Berikut adalah contoh penggunaan konstanta di dalam interface:

```

1 <?php
2 interface mouse{
3     const JENIS = "Laser Mouse";
4     public function klik_kanan();
5     public function klik_kiri();
6 }
7
8 echo mouse::JENIS;
9 // Laser Mouse
10 ?>

```

Untuk mengakses konstanta dari *interface*, kita menggunakan perintah `nama_interface::nama_konstanta`.

Interface Tidak Bisa Memiliki Method 'normal'

Salah satu yang membedakan *interface* dengan *abstract class* adalah *kita tidak bisa membuat method biasa di dalam Interface*. Contoh berikut akan menghasilkan *error*:

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri(){
5         return "Klik Kiri...";
6     }
7 }
8
9 class laptop implements mouse{
10    public function klik_kanan(){
11        return "Klik Kanan...";
12    }
13 }
14
15 $laptop_baru = new laptop();
16 // Fatal error: Interface function
17 // mouse::klik_kiri() cannot contain body
18 ?>

```

Sebuah Class Bisa Menggunakan Banyak Interface

Perbedaan lain antara *Interface* dengan *Abstract Class* adalah: Sebuah *class* bisa menggunakan lebih dari 1 *interface*, sedangkan untuk *abstract class*, kita hanya bisa menggunakan 1 *abstract class* dalam sekali penurunan *class*.

Untuk menggunakan lebih dari 1 *interface*, kita tinggal menuliskan semua *interface* setelah keyword *implements*. Setiap nama *interface* dipisahkan dengan tanda koma.

Berikut contoh penggunaan 2 buah *interface* di dalam PHP:

```

1 <?php
2 interface mouse{
3     public function klik_kanan();
4     public function klik_kiri();
5 }
6
7 interface keyboard{
8     public function tekan_enter();
9 }
10
11 class laptop implements mouse, keyboard{
12     public function klik_kanan(){
13         return "Klik Kanan...";
14     }
15
16     public function klik_kiri(){
17         return "Klik Kiri...";
18     }
19
20     public function tekan_enter(){
21         return "Tekan Tombol Enter...";
22     }
23 }
24
25 $laptop_baru = new laptop();
26 echo $laptop_baru->tekan_enter();
27 // Tekan Tombol Enter...
28 ?>

```

Dalam contoh diatas saya membuat dua buah *interface*, yakni *mouse* dan *keyboard*. Kedua *interface* ini kemudian digunakan oleh *class laptop*.

Fungsi Interface dalam Pemrograman Objek

Jika anda telah mempelajari [abstract class](#) dalam tutorial kita sebelumnya, sedikit banyak kita bisa memahami fungsi *interface* jika dibandingkan dengan fungsi *abstract class*. Kedua konsep ini sering membuat bingung karena mirip dalam implementasinya.

Interface lebih berperan untuk *menyeragamkan method*. Ia tidak masuk kedalam struktur class seperti *abstract class*. Jika kita menggunakan *abstract class komputer* sebagai ‘konsep class’ untuk kemudian diturunkan kepada class lain seperti *class laptop*, *class pc*, dan *class netbook*, maka interface hanya ‘*penyedia method*’. *Interface* tidak termasuk kedalam pewarisan class.

[Object Interface](#) dan [Abstract Class](#) merupakan implementasi dari konsep pemrograman objek yang disebut sebagai *Polimorfisme*. Pengertian *Polimorfisme* atau *polymorphism*, akan kita bahas dalam tutorial OOP PHP selanjutnya.

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoloading* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

*** Artikel Terkait ***

Tags: [Cara Penulisan Interface PHP](#), [Pemrograman Objek PHP](#), [Pengertian Interface](#), [Pengertian Object Interface](#), [tutorial belajar PHP](#), [Tutorial OOP PHP](#)

19 COMMENTS

yusuf





Home | Tutorial PHP | Tutorial Belajar OOP PHP Part 17: Pengertian Polimorfisme dalam Pemrograman Objek PHP

Tutorial Belajar OOP PHP Part 17: Pengertian Polimorfisme dalam Pemrograman Objek PHP

16 Oct 14 | Andre | Tutorial PHP | 100 Comments

eBook Terbaru DuniaIlkom



Cara pemesanan eBook & Buku

DuniaIlkom

Daftar eBook DuniaIlkom

eBook Pascal Uncover

eBook HTML Uncover

eBook CSS Uncover

eBook PHP Uncover

eBook MySQL Uncover

eBook JavaScript Uncover

eBook Bootstrap Uncover

eBook OOP PHP Uncover

eBook Laravel Uncover

Abstract Class dan **Object Interfaces** yang kita pelajari dalam 2 tutorial sebelum ini merupakan implementasi dari konsep pemrograman berbasis objek yang dinamakan **Polimorfisme**. Dalam tutorial OOP PHP kali ini, kita akan membahas [pengertian polimorfisme dalam pemrograman objek PHP](#), disertai contoh penggunaannya.

Pengertian Polimorfisme

Dari segi bahasa, **Polimorfisme** (bahasa Inggris: *Polymorphism*) berasal dari dua kata bahasa Latin yakni **poly** dan **morph**. **Poly** berarti banyak, dan **morph** berarti bentuk. **Polimorfisme** berarti banyak bentuk ([wikipedia](#)).

Di dalam pemrograman objek, **polimorfisme** adalah *konsep dimana terdapat banyak class yang memiliki signature method yang sama*. Implementasi dari method-method tersebut diserahkan kepada tiap class, akan tetapi cara pemanggilan method harus sama. Agar kita dapat ‘memaksakan’ signature method yang sama pada banyak class, class tersebut harus diturunkan dari sebuah **abstract class** atau **object interface**.

Sebagai contoh, berikut adalah kode PHP yang mengimplementasikan konsep **polimorfisme**:



```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function booting_os();
6 }
7
8 class laptop extends komputer{
9     public function booting_os(){
10        return "Proses Booting Sistem Operasi Laptop";
11    }
12 }
13
14 class pc extends komputer{
15     public function booting_os(){
16        return "Proses Booting Sistem Operasi PC";
17    }
18 }
19
20 class chromebook extends komputer{
21     public function booting_os(){
22        return "Proses Booting Sistem Operasi Chromebook";
23    }
24 }
25
26
27 // buat objek dari class diatas
28 $laptop_baru = new laptop();
29 $pc_baru = new pc();
30 $chromebook_baru = new chromebook();
31
32 // buat fungsi untuk memproses objek
33 function booting_os_komputer($objek_komputer){
34     return $objek_komputer->booting_os();
35 }
36
37 // jalankan fungsi
38 echo booting_os_komputer($laptop_baru);
39 echo "<br />";
40 echo booting_os_komputer($pc_baru);
41 echo "<br />";
42 echo booting_os_komputer($chromebook_baru);
43 ?>

```

Contoh kode diatas cukup panjang, namun jika anda mengikuti tutorial OOP PHP sebelumnya (tentang *abstract class*), maka kode diatas akan bisa dipahami dengan baik.

Pada awal program, saya membuat *abstract class komputer* yang kemudian diturunkan kedalam 3 class lain, yakni: *class laptop*, *class pc* dan *class chromebook*. *Abstract class komputer* memiliki *abstract method booting_os()*, yang harus diimplementasikan ulang pada tiap class yang diturunkan dari *class komputer*. Setelah pendefenisian class, saya membuat 3 objek dari masing-masing class.

Perhatikan bahwa setelah pembuatan objek dari masing-masing *class*, saya membuat fungsi *booting_os_komputer()*. Fungsi ini berperan untuk memanggil *method-method* dari setiap *class*.

Konsep *polimorfisme* dari contoh diatas adalah, fungsi *booting_os_komputer()* akan selalu berhasil dijalankan, selama *argumen* yang diberikan berasal dari class yang diturunkan dari *class abstract komputer*.

Peran Abstract Class dan Interface dalam Polimorfisme

Baik *abstract class* maupun *interface* bisa digunakan untuk membuat banyak class dengan method yang sama. Bahkan keduanya sering digunakan secara bersama-sama.

List Tutorial DuniaIlkom

- [Tutorial Terbaru DuniaIlkom](#)
- [Tutorial HTML](#)
- [Tutorial PHP](#)
- [Tutorial MySQL](#)
- [Tutorial CSS](#)
- [Tutorial JavaScript](#)
- [Tutorial jQuery](#)
- [Tutorial WordPress](#)
- [Tutorial Pascal](#)
- [Tutorial C](#)
- [Tutorial Python](#)
- [Tutorial Java](#)
- [Tutorial Laravel](#)
- [Membuat Web Online](#)
- [Review Jurusan Kuliah](#)
- [Blog DuniaIlkom](#)

Tutorial Dasar PHP

- [Tutorial PHP Dasar](#)
- [1. Pengertian PHP](#)
- [2. Sejarah PHP](#)
- [3. Menginstall XAMPP](#)
- [4. Menjalankan Apache](#)
- [5. Menjalankan File PHP](#)
- [6. Cara Kerja WebServer](#)
- [7. Input PHP ke HTML](#)
- [8. File php.ini](#)
- [9. Dasar Penulisan PHP](#)
- [10. Penulisan Komentar](#)
- [11. Penulisan Variabel](#)
- [12. Penulisan Konstanta](#)
- [13. Tipe Data Integer](#)
- [14. Tipe Data Float](#)
- [15. Tipe Data String](#)
- [16. Tipe Data Boolean](#)
- [17. Tipe Data Array](#)

Berikut adalah revisi kode program kita sebelumnya dengan menggunakan ***abstract class*** dan ***interface***:

```

1 <?php
2 // buat abstract class
3 abstract class komputer{
4     // buat abstract method
5     abstract public function booting_os();
6 }
7
8 interface mouse{
9     public function double_klik();
10 }
11
12 class laptop extends komputer implements mouse{
13     public function booting_os(){
14         return "Proses Booting Sistem Operasi Laptop";
15     }
16     public function double_klik(){
17         return "Double Klik Mouse Laptop";
18     }
19 }
20
21 class pc extends komputer implements mouse{
22     public function booting_os(){
23         return "Proses Booting Sistem Operasi PC";
24     }
25     public function double_klik(){
26         return "Double Klik Mouse PC";
27     }
28 }
29
30 class chromebook extends komputer implements mouse{
31     public function booting_os(){
32         return "Proses Booting Sistem Operasi Chromebook";
33     }
34     public function double_klik(){
35         return "Double Klik Mouse Chromebook";
36     }
37 }
38
39 // buat objek dari class diatas
40 $laptop_baru = new laptop();
41 $pc_baru = new pc();
42 $chromebook_baru = new chromebook();
43
44 // buat fungsi untuk memproses objek
45 function booting_os_komputer($objek_komputer){
46     return $objek_komputer->booting_os();
47 }
48
49 function double_klik_komputer($objek_komputer){
50     return $objek_komputer->double_klik();
51 }
52
53 // jalankan fungsi
54 echo booting_os_komputer($laptop_baru);
55 echo "<br />";
56 echo double_klik_komputer($laptop_baru);
57 echo "<br />";
58 echo "<br />";
59
60 echo booting_os_komputer($pc_baru);
61 echo "<br />";
62 echo double_klik_komputer($pc_baru);
63 echo "<br />";
64 echo "<br />";
65
66 echo booting_os_komputer($chromebook_baru);
67 echo "<br />";
68 echo double_klik_komputer($chromebook_baru);
69 ?>

```

Pada kode program diatas, saya membuat 1 ***abstract class***: *komputer*, dan 1 ***interface***: *mouse*. Keduanya kemudian di turunkan kepada 3 class: *class laptop*, *class pc*, dan *class chromebook*.

Selama sebuah class diturunkan dari ***abstract class komputer***, dan menggunakan ***interface mouse***, fungsi ***booting_os_komputer()*** dan fungsi ***double_klik_komputer()*** akan selalu berhasil di jalankan, terlepas dari apapun nama objek dan implementasi method yang digunakan.

- 18. Pengertian Operand
- 19. Fungsi var_dump()
- 20. Operator Aritmatika
- 21. Operator String
- 22. Operator Logika
- 23. Perbandingan
- 24. Operator Increment
- 25. Assignment PHP
- 26. Operator Bitwise
- 27. Operator Assignment
- 28. Type Casting
- 29. Struktur Logika IF
- 30. Struktur ELSE
- 31. Logika ELSE-IF
- 32. Struktur Switch
- 33. Perulangan For
- 34. Perulangan While
- 35. Do-While
- 36. Perintah Break
- 37. Perintah Continue
- 38. Perulangan Foreach
- 39. Pengertian Function
- 40. Penulisan Function
- 41. Variabel Scope
- 42. Argumen Function
- 43. Default Parameter
- 44. Variable Parameter

Tutorial PHP Lanjutan: Form

Tutorial PHP Lanjutan: Form ▾

Tutorial PHP Lanjutan: PHP - MySQL

Tutorial PHP Lanjutan: PHP - MySQL ▾

Tutorial PHP Lanjutan: OOP PHP

Tutorial PHP Lanjutan: OOP PHP ▾

Berlangganan Artikel DuniaIlkom ^

Konsep **polimorfisme** yang kita bahas dalam tutorial ini bertujuan untuk membuat struktur pola dari class dan turunannya. Lebih jauh lagi, *polimorfisme* menekankan alur kode program yang terorganisir untuk mengurangi adanya perulangan kode program.

eBook OOP PHP Uncover DuniaIlkom

DuniaIlkom telah menerbitkan buku yang secara detail membahas pemrograman object PHP. Mulai dari materi dasar OOP seperti *class*, *object*, *property*, hingga *trait*, *namespace*, *autoload* dan *exception*. Di akhir buku juga terdapat studi kasus pembuatan library dan aplikasi CRUD. Penjelasan lebih lanjut bisa ke [eBook OOP PHP Uncover DuniaIlkom](#).

Dapatkan pemberitahuan untuk setiap artikel dan tutorial terbaru DuniaIlkom

Join 3,082 other subscribers

*** Artikel Terkait ***

Tags: [Belajar Objek PHP](#), [OOP PHP](#), [Pemrograman Berbasis Objek](#), [Pemrograman Objek PHP](#), [Pengertian Polimorfisme](#), [Pengertian Polimorfisme PHP](#), [tutorial belajar PHP](#), [Tutorial OOP PHP](#)

100 COMMENTS

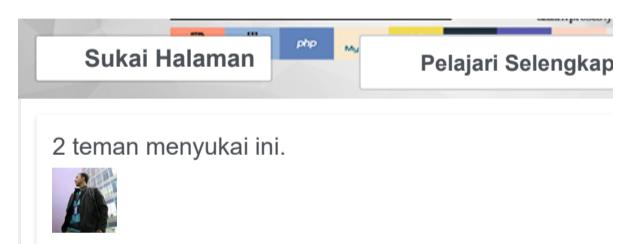


aemana

22 Dec 14

gan makasi banyak ya website nya banyak membantu saya dalam belajar dimohon contohnya diperbanyak dan contoh yg sering terjadi gan atau buat studi kasus membuat webite toko online dr awal atau apa gitu gan tp skali lg makasi banyak yo gan. semoga terus diupdate materinya, ajax , jquery dll. semoga web nya semakin ramai jd gan.

[Reply](#)



Andre

Author

23 Dec 14



Tugas

1. Setiap mahasiswa mengikuti seluruh tutorial berikut dan mendokumentasikan program, hasil beserta penjelasannya dalam sebuah laporan praktikum.
 - Link tutorial 1: **OOP DASAR pada PHP**
https://www.youtube.com/watch?v=ZKDUFoouyBI&list=PLFIM0718LjIWvxll-6wLxrC_16h_BI_p
 - Link tutorial 2: **Membuat Aplikasi MVC dengan PHP**
https://www.youtube.com/watch?v=tBKOb8Ib5nI&list=PLFIM0718LjIVEh_d-h5wAjsdv2W4SAtkx
2. Setiap mahasiswa mengupload kode program dari link tutorial 1 dan link tutorial 2 kedalam code repository bernama github menggunakan **git command/git bash (bukan upload manual)**. Mungkin link tutorial ini sedikit membantu anda:
<https://www.tutsmake.com/upload-project-files-on-github-using-command-line/>
3. Anda hanya perlu mencantumkan URL project github anda pada laporan praktikum dan pastikan project/repository yang anda buat bersifat **public**.

Referensi

https://www.tutorialspoint.com/php/php_object_oriented.htm

https://www.w3schools.com/php/php_oop_whatis.asp