Konsep dan Teknik

Menguasai Modern OOP di PHP

Awan Pribadi Basuki

CV. LOKOMEDIA

Konsep dan Teknik Menguasai Modern OOP di PHP

Perpustakaan Nasional : Katalog Dalam Terbitan (KDT)
Penulis : Awan Pribadi Basuki
Konsep dan Teknik Menguasai Modern OOP di PHP
- Cet. I. - Yogyakarta : Penerbit Lokomedia, 2017
194 halaman; 14 x 21 cm
ISBN : 978-602-6231-09-3

Penerbit Lokomedia,
Cetakan Pertama : April 2017

Editor : Lukmanul Hakim
Cover : Subkhan Anshori
Layout : Lukmanul Hakim

Diterbitkan pertama kali oleh :
CV. LOKOMEDIA
Jl. Jambon, Perum. Pesona Alam Hijau 2 Kav. B-4, Kricak
Yogyakarta 55242.

email : redaksi@bukulokomedia.com
website : www.bukulokomedia.com

KATA PENGANTAR

PHP pada awalnya diciptakan sebagai bahasa scripting sederhana yang bersifat prosedural dan tidak mendukung Object Oriented Programming (OOP). Namun pada perkembangannya, ada kebutuhan bahwa PHP perlu mendukung OOP secara penuh mengingat manfaatnya dalam dunia pemrograman modern. Dengan OOP akan membuat pemrograman menjadi lebih mudah dan efisien.

Lahirnya berbagai Framework PHP yang menggunakan OOP sebagai cara kerjanya menandakan penerimaan konsep OOP pada komunitas PHP. Sebut saja Codeigniter, Symfony, Laravel, Yii, CakePHP dan sebagainya semuanya menerapkan OOP. Tentu saja penerapannya berbeda-beda pada tiap-tiap framework tersebut.

Tentu saja, untuk menggunakan framework-framework tersebut secara efisien dibutuhkan pengetahuan dan pemahaman yang cukup dasar-dasar OOP di PHP. Sayangnya tidak banyak referensi yang dapat dipakai oleh programmer PHP, terutama tingkat pemula yang ingin meningkatkan kemampuannya dalam menerapkan OOP pada PHP.

Buku ini ditulis sebagai petunjuk sederhana untuk memulai mempelajari OOP pada PHP. Pembahasan meliputi konsep-konsep inti OOP, yaitu konsep-konsep terkini yang banyak diterapkan pada pemrograman PHP modern, dan tentunya juga diterapkan pada framework-framework PHP masa kini.

Diharapkan buku ini akan menjadi panduan yang mudah dan cepat bagi para pemula untuk menguasai dan menerapkan OOP pada PHP.

Terima kasih Penerbit Lokomedia, yang untuk ke-sekian kalinya bersedia bekerja sama untuk menerbitkan buku yang saya tulis.

Buku ini tentu jauh dari sempurna. Untuk itu jika Anda mempunyai pertanyaan, saran atau kritik seputar buku ini bisa Anda sampaikan melalui email.

Pasirian, Desember 2016

Awan Pribadi Basuki

awan_pribadi@yahoo.com

Halaman ini Sengaja Dikosongkan

www.bukulokomedia.com

DAFTAR ISI

Halaman ini Sengaja Dikosongkan

www.bukulokomedia.com

BAB

Pembuka

BAB
Pembuka

Mengapa OOP?

Pada awalnya PHP ditujukan untuk pemrograman prosedural. Seiring dengan perkembangan zaman, pemrograman prosedural mulai ditinggalkan karena memiliki banyak keterbatasan. OOP hadir sebagai solusi untuk mengatasi keterbatasan yang ada pada pemograman prosedural.

Sebagai bahasa pemrograman modern yang terus berkembang, PHP juga mulai mendukung OOP. Dimulai pada PHP 5, dukungan terhadap OOP di PHP menjadi lebih "serius", tidak seperti di PHP 4. Oleh karena itu, para programmer di komunitas PHP juga mulai mengadopsi OOP untuk pembuatan kodenya.

Perkembangan OOP di PHP tidak lepas dengan munculnya berbagai framework PHP yang memakai OOP sebagai cara kerjanya. Jadi, para programmer harus mulai menguasai OOP jika ingin bekerja dengan framework-framework tersebut.

Jadi mengapa kita perlu mempelajari OOP di PHP? Sudah jelas, karena **kebanyakan kode PHP sekarang sudah ditulis dalam bentuk OOP**. Agar tidak tertinggal, maka kita juga harus mempelajari OOP.

Untuk Siapa Buku Ini Ditulis?

Buku ini BUKAN untuk Anda yang sudah mahir dan sudah terbiasa melakukan pemrograman OOP di PHP, Anda akan kecewa. Semua materi yang ada di buku ini akan terlalu mudah bagi Anda.

Buku ini juga BUKAN untuk Anda yang benar-benar baru mulai dengan pemrograman PHP, atau bahkan baru mulai mengenal dunia pembuatan web. Anda akan mengalami kesulitan.

Buku ini ditujukan bagi Anda yang sudah mengetahui dasar pembangunan web, dan sudah pernah menggunakan PHP. Buku ini ditujukan bagi Anda yang ingin meningkatkan *skill* PHP dengan menguasai teknik *Object Oriented Programming* (OOP) di PHP. Oleh karena itu, diasumsikan bahwa Anda sudah memahami

konsep-konsep dasar PHP, misalnya tentang variabel, *variable scope*, fungsi, *constant*, percabangan (if…else; switch), loop (for; foreach). Lebih baik lagi jika Anda sudah sempat membaca dan mengetahui sedikit tentang dasar-dasar OOP, tapi belum mantap dalam pemahaman Anda. Itu akan sangat membantu dalam mengikuti pembahasan di buku ini.

Catatan Tentang Penggunaan Istilah

Dalam dunia Object Oriented Programming, banyak sekali *jargon* dan istilah teknis untuk mewakili suatu konsep. Kebanyakan istilah-istilah tersebut berasal dari Bahasa Inggris. Misalnya: **class, object, inheritance, polymorphism, abstract, public, protected, private** dan sebagainya.

Penjelasan tentang istilah-istilah dan konsep-konsep tersebut akan diberikan pada bagiannya masing-masing. Namun selanjutnya penyebutan istilah-istilah tersebut akan tetap memakai istilah-istilah aslinya, yaitu dalam bentuk Bahasa Inggris. Istilah-istilah tersebut tidak disebutkan dalam terjemahan Bahasa Indonesia. Mengapa? Hal itu penulis lakukan agar penggunaan istilah-istilah tersebut konsisten dan tidak menimbulkan salah pengertian.

Berikut ini contoh istilah-istilah tersebut:

- ✓ **Class**: blueprint / cetak biru yang menjadi struktur dasar object. Terdiri dari method dan property. Selanjutnya istilah **class** tetap akan ditulis sebagai **class**, bukan **kelas**. Istilah **kelas** lebih mengacu pada ruang kelas (tempat belajar) atau tingkatan / level (SMA kelas XII)
- ✓ **Object**: hasil instansiasi dari suatu class. Selanjutnya istilah **object** tetap ditulis sebagai **object** bukan **obyek**. Istilah **obyek** mengacu pada suatu benda.
- ✓ **Private Property**: variabel pada suatu class yang hanya dapat diakses dari class itu sendiri. Selanjutnya istilah **private property** tetap ditulis sebagai **private property** bukan **properti pribadi**. Istilah **properti pribadi** lebih mengacu pada tanah / bangunan aset.

Ketentuan ini juga berlaku untuk istilah-istilah OOP yang lainnya. Sekali lagi itu agar penggunaan istilah-istilah tersebut konsisten dan tidak menimbulkan salah pengertian.

Komputer dan PHP yang Penulis Gunakan

Untuk membuat semua tutorial di buku ini, penulis memakai komputer yang

menjalankan Windows 7 SP1 yang sudah terinstal paket aplikasi Wampserver 3.0.6. Di dalamnya sudah termasuk PHP 5.6.25 dan PHP 7.0.10. Karena WampServer 3.0.6 secara default menawarkan 2 versi PHP seperti yang sudah disebutkan, maka penulis mengaktifkan PHP 7.0.10.

Untuk **mengaktifkan PHP 7.0.10 di Wampserver**:

Langkah 1:

Jalankan program Wampserver.

Langkah 2:

Klik icon Wampserver di status bar > PHP > Version. Pilih versi PHP yang akan dijalankan, dalam hal ini PHP 7.0.10.

Cara Menjalankan Skrip PHP

Ada 2 yang biasa dilakukan untuk menjalankan skrip PHP:

Melalui Browser

Untuk menjalankan skrip PHP dari browser cukup memanggil URL skrip PHP yang akan dijalankan. Tentu saja server sudah harus dalam kondisi "on".

Melalui Command Prompt

Untuk menjalankan skrip PHP melalui command prompt, maka command prompt harus mengenali perintah `php`. Agar perintah `php` dapat dikenali di jendela command prompt, yang harus dilakukan adalah menambahkan *environment path* file executable PHP (`php.exe`).

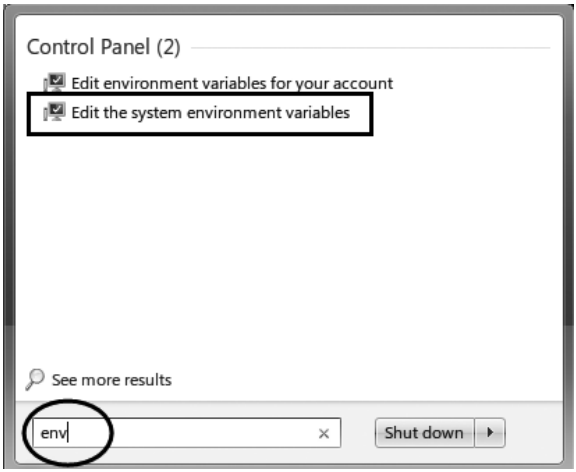
Berikut ini cara menambahkan *environment path* PHP:

Langkah 1:

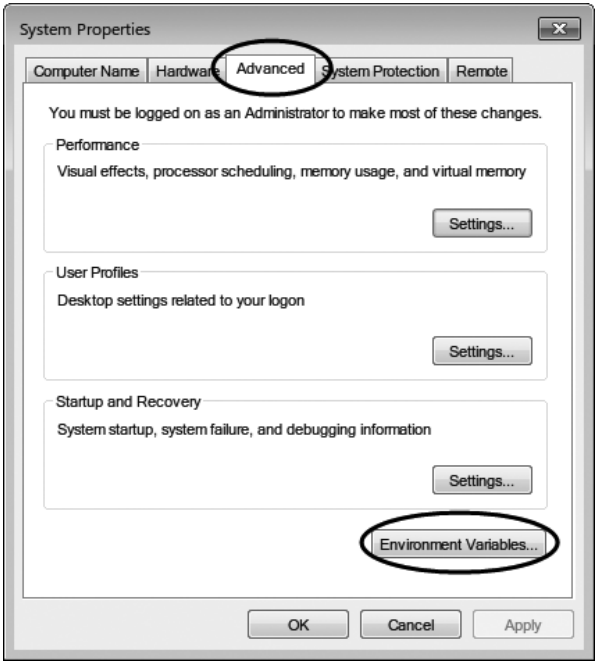
Klik start menu Windows, ketik "envi" (tanpa tanda petik). Pilih **Edit the system environment variabels**. Lihat gambar 1.

Langkah 2:

Pada jendela **System Properties**, pilih tab **Advanced** > **Environment Variables**. Lihat gambar 2.



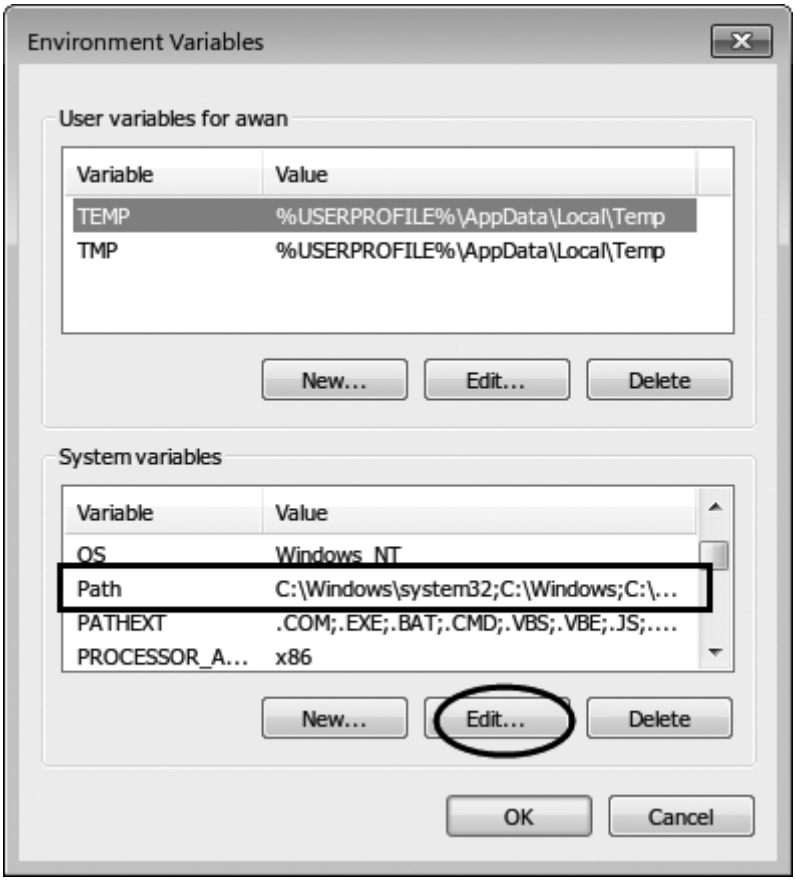
Gambar 1 Menjalankan environment variables



Gambar 2. Jendela System Properties

Langkah 3:

Pada opsi **System Variables**, pilih item **path**. Klik tombol **Edit**. Lihat gambar 3.



Gambar 3 Jendela Environment Variables

Langkah 4:

Pada jendela **Edit System Variable**, pada kotak **Variable value**, letakkan kursor pada bagian paling akhir. Ketik `;C:\wamp\bin\php\php7.0.10;` seperti yang tampak pada gambar 4.

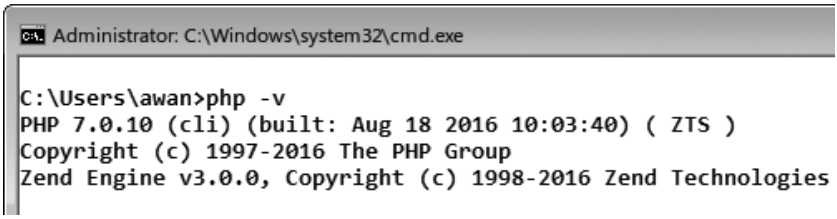
Gambar 4 Jendela Edit System Variable

Path `C:\wamp\bin\php\php7.0.10` adalah lokasi file `php.exe` untuk PHP 7.0.10 pada Wampserver yang penulis pakai. Tanda titik koma di awal dan di akhir path PHP berfungsi sebagai delimiter (pembatas) dengan path-path program lain.

Jika Anda memakai aplikasi lain misalnya Anda memakai XAMPP, silakan menyesuaikan path file `php.exe` sesuai dengan aplikasi server yang Anda pakai. Pada prinsipnya sama saja.

Langkah 5:

Untuk menguji apakah perintah `php` sudah dapat dikenali dan dijalankan, buka command prompt, kemudian ketik `php -v`. Jika path PHP sudah ditambahkan dengan benar, maka akan muncul tampilan seperti gambar berikut yang menunjukkan versi PHP yang berjalan pada komputer yang kita gunakan. Lihat gambar 5.



Gambar 5 Menjalankan perintah php melalui command prompt

Jika masih belum dikenali, tutup jendela command prompt, kemudian buka jendela command prompt baru. Ketik lagi perintah di atas.

Membuat Folder Tempat Menyimpan File Latihan

Seperti yang sudah penulis sebutkan sebelumnya bahwa penulis memakai Wampserver 3.0.6, maka lokasi default untuk menyimpan file web (file PHP) adalah di folder `C:/wamp/www`. Agar file-file latihan yang kita buat lebih teratur, buat sebuah folder bernama `oop` di dalam folder `C:/wamp/www`. Sehingga nanti kita akan menyimpan file-file latihan di `C:/wamp/www/oop`.

BAB I

Memahami Class dan Object

BAB 1
Memahami Class dan Object

1.1 Membuat Class

Pada dasarnya, cara kerja utama OOP adalah mengelompokkan variabel dan fungsi yang sejenis ke dalam satu class. Yang dimaksud dengan "sejenis" di \sini adalah yang **memiliki kedekatan satu sama lain**.

Misalnya pada sebuah blog, terdapat 3 komponen: **user**, **post** dan **comment**. Kita bisa mengelompokkan fungsi-fungsi dan variabel-variabel pada blog tersebut ke dalam 3 class: `User{}`, `Post{}` dan `Comment{}`.

Tiap-tiap class nantinya akan berisi variabel-variabel dan fungsi-fungsi yang berkaitan dengan object tersebut. Misalnya pada class `User{}`, maka akan terdapat **variabel**: `$username`, `$password`, `$lastLogin`. Selain itu, juga terdapat **fungsi**: `setUsername()`, `ubahPassword()` dan sebagainya.

Pada class `Post{}` terdapat **variabel**: `$judul`, `$isiPost`, `$tanggal` dan lainnya. Selain itu, juga terdapat **fungsi**: `simpanPost()`, `editPost()`, `hapusPost()` dan sebagainya. Demikian juga pada class `Comment{}` didalamnya terdapat variabel-variabel dan fungsi-fungsi yang berkaitan dengan komentar.

Sebelum digunakan, **object harus dirancang dulu dalam bentuk sebuah class**. Boleh dikatakan bahwa sebuah class adalah cetak biru / rancangan / cetakan dari sebuah object. Sehingga, dari sebuah class nanti bisa diciptakan banyak object.

Misalnya untuk menangani aplikasi yang berkaitan dengan mobil, maka kita bisa membuat sebuah class bernama `Mobil{}` seperti berikut:

Buat sebuah file bernama `latihan.php`, simpan di `C:/wamp/www/oop`. Anda masih ingat, kita sudah membuat folder `oop` pada bab sebelumnya bukan?

```
1    <?php
2
3    class Mobil
4    {
5        // kode program untuk class mobil
6    }
```

Nomor urut didepan kode program tidak usah diketik, itu hanya digunakan sebagai bantuan untuk menunjukkan nomor baris program.

Selanjutnya, file `latihan.php` akan kita gunakan untuk menuliskan kode-kode program untuk beberapa latihan ke depan.

Penjelasan Script:

- **Baris 3**. Untuk membuat sebuah class, kita memakai keyword `class`, di ikuti nama class, dalam hal ini `Mobil`.
- Nama class diawali dengan huruf kapital.
- Jika nama class terdiri dari beberapa kata, maka kata kedua dan seterusnya juga ditulis bersambung dan diawali pula dengan huruf kapital. Misalnya: `MobilJepang`, `MobilEropa`.
- Nama class di ikuti dengan tanda pembuka dan penutup kurung kurawal `{}`. Kode program untuk class akan ditulis diantara kedua tanda tersebut.

Catatan:

- ✓ Untuk membedakan antara nama variabel, fungsi dan class, maka dalam penjelasan penulis menambahkan tanda `{}` di belakang nama class, misalnya: `Mobil{}`, `Post{}`, `Comment{}`.
- ✓ Sedangkan untuk menunjukkan nama fungsi / method maka penulis menambahkan tanda `()` di belakang nama fungsi, misalnya: `setJudul()`, `hapusComment()`.
- ✓ Demikian juga untuk variabel, penulis menambahkan tanda `$` di depan nama variabel, misalnya: `$nama`, `$tanggalPosting`.

1.2 Menambahkan Property

Anda sudah mengenal variabel bukan? Variabel berfungsi **untuk menyimpan data sementara pada saat program berjalan**.

Seorang manusia tentu memiliki nama, tanggal lahir, kota lahir dan beberapa informasi yang melekat kepadanya. Dengan kata lain pada sebuah class bernama `Manusia{}` memiliki variabel-variabel: `$nama`, `$kotaLahir`, `$tanggalLahir`.

Sebuah mobil tentu memiliki merk, tahun keluaran, tipe / model, warna. Boleh dikatakan bahwa sebuah class `Mobil` memiliki variabel-variabel `$merk`, `$tahun` `$warna`, dan `$tipe`.

Dalam konteks OOP, kita menyebut **istilah variabel sebagai *property***. Jadi, **property adalah variabel-variabel yang melekat pada sebuah class**.

Untuk menambahkan property ke dalam class `Mobil`:

Pada file `latihan.php`, tambahkan kode berikut:

```
1    <?php
2
3    class Mobil
4    {
5        public $merk;
6        public $tipe;
7        public $tahun;
8        public $warna = 'biru';
9    }
```

Penjelasan Script:

- Untuk mendeklarasikan sebuah property pada suatu class diawali keyword `public`, di ikuti nama variabel / property.
- Keyword `public` berarti property tersebut dapat diakses langsung dari luar class. Selain `public`, ada beberapa tipe variabel lain yaitu `protected` dan `private`. Kita akan membahasnya di bagian yang lain.
- Tata cara pemberian nama property sama dengan pemberian nama variabel biasa pada PHP. Pada standar PHP nama variabel biasanya di awali dengan huruf kecil. Contoh: `$nama`, `$merk`, `$warna`.
- Jika nama property lebih dari satu kata, kata pertama diawali huruf kecil, kata-kata berikutnya ditulis bersambung dan diawali dengan huruf kapital. Misal: `$tanggalLahir`, `$jenisKelamin`, `$namaBuahFavorit`, dan sebagainya.
- Kita bisa memberikan nilai default (nilai awal) pada sebuah property. Misalnya pada **Baris 8**, property `$warna` kita memiliki nilai default `'biru'`.

1.3 Membuat Object dari Sebuah Class

Sebelum menggunakan sebuah class, kita **harus membuat object dari class** tersebut. Kita tidak dapat menggunakan rancangan sebuah mobil, karena itu hanyalah rancangan. Yang kita pakai adalah object mobil, yang benar-benar kita kendarai. Bukankah demikian?

Proses pembuatan object dari sebuah class disebut *instantiation*. Sehingga **sebuah object merupakan *instance* dari suatu class**.

Untuk membuat object dari class `Mobil{}`, caranya:

Ubah `latihan.php` menjadi:

```
1     <?php
2
3     class Mobil
4     {
5         public $merk;
6         public $tipe;
7         public $tahun;
8         public $warna = 'biru';
9     }
10
11    $mobil = new Mobil();
12    var_dump($mobil);
```

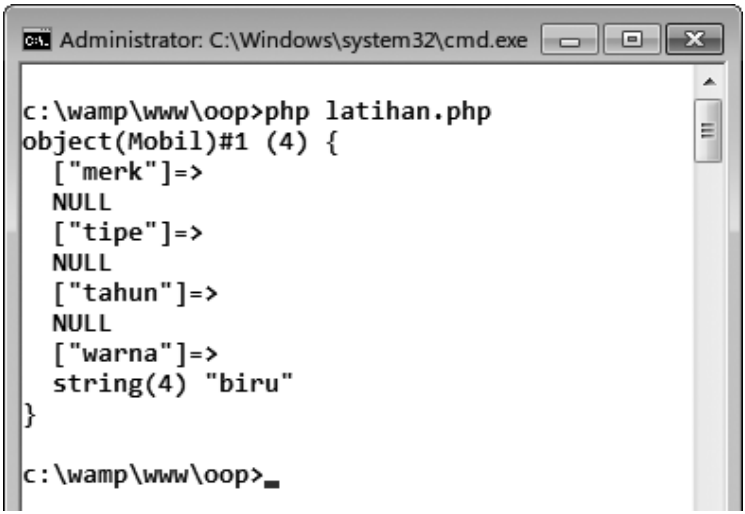
Penjelasan Skrip:

Baris 11. Untuk membuat object, gunakan keyword `new` di ikuti nama class-nya.

Untuk menjalankan skrip, buka command prompt. Masuk ke folder `C:/wamp/www/oop`. Untuk menjalankan skrip lewat command prompt harus berada di folder tempat menyimpan file yang akan dijalankan.

Kemudian ketik perintah: `php latihan.php`

Arti perintah di atas adalah menjalankan skrip php `latihan.php`. Lihat gambar 1.1.



Gambar 1.1 Hasil contoh script pembuatan object dari sebuah class

Jika Anda mendapatkan tampilan seperti gambar 1.1, maka Anda sudah berhasil membuat object dari class `Mobil{}`.

1.4 Mengakses Property

Sebelumnya kita sudah membuat class `Mobil{}` dan membuat object dari class tersebut. Di dalam class `Mobil{}` terdapat beberapa property. Bagaimana caranya jika kita ingin menampilkan nilai property-property tersebut?

Untuk mengakses nilai property:

Ubah `latihan.php` menjadi:

```
1     <?php
2
3     class Mobil
4     {
5         public $merk = 'Toyota';
6         public $tipe = 'Fortuner';
7         public $tahun = 2016;
8         public $warna = 'Putih';
9     }
10
11    $mobil = new Mobil();
12    echo "Mobil $mobil->merk \n";
13    echo "Model $mobil->tipe \n";
14    echo "Tahun $mobil->tahun \n";
```

Penjelasan Skrip:

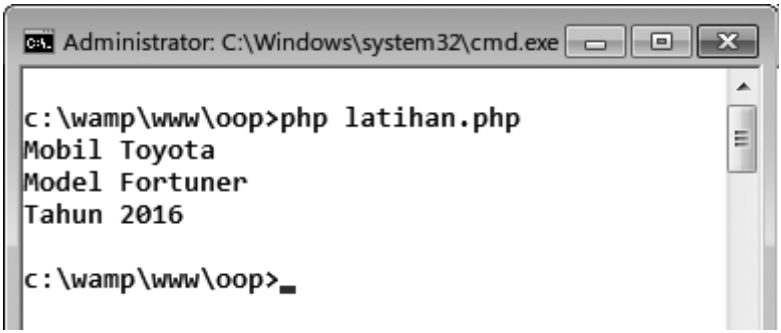
- **Baris 11**. Membuat object dari class `Mobil{}`, kemudian menyimpannya ke dalam variabel `$mobil`.
- **Baris 12**. Mengakses property `$merk` dari object `$mobil`. Perhatikan tanda dollar (`$`) ditulis sebelum nama variabel yang menyimpan instance object (`$mobil`). Sedangkan pada nama property tidak usah ditulis tanda dollar, sebagai gantinya tanda minus (`-`) dan lebih besar (`>`) yang membentuk tanda panah, kemudian diikuti nama property.

  Jadi: `$mobil->merk;`

  Dibaca: property `$merk` dari object `Mobil{}`.

  Perintah `echo` untuk mencetak ke layar, perintah `\n` untuk ganti baris.
- **Baris 13**. Mengakses property `$tipe` dari object / class `Mobil{}`.
- **Baris 14**. Mengakses property `$tahun` dari object / class `Mobil{}`.

Jalankan skrip `latihan.php`, maka hasilnya dapat dilihat pada gambar 1.2.



Gambar 1.2 Mengakses nilai property

1.5 Mengatur Nilai Property

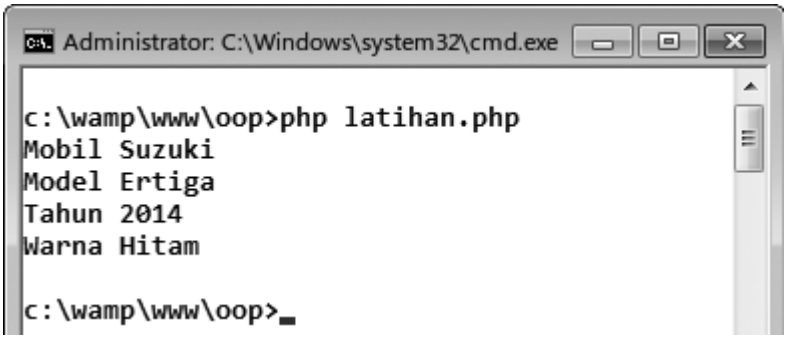
Cara mengatur nilai property pada suatu object **sangat mirip dengan cara mengakses nilainya**. Kita tinggal memberikan nilai yang kita kehendaki seperti saat kita memberikan nilai pada suatu variabel. Untuk mengatur nilai pada property. Ubah `latihan.php` menjadi:

```
1     <?php
2
3     class Mobil
4     {
5         public $merk = 'Toyota';
6         public $tipe = 'Fortuner';
7         public $tahun = 2016;
8         public $warna = 'Putih';
9     }
10
11    $mobil = new Mobil();
12
13    // Mengatur nilai property
14    $mobil->merk = 'Suzuki';
15    $mobil->tipe = 'Ertiga';
16    $mobil->tahun = 2014;
17    $mobil->warna = 'Hitam';
18
19    // Mencetak nilai property
20    echo "Mobil $mobil->merk \n";
21    echo "Model $mobil->tipe \n";
22    echo "Tahun $mobil->tahun \n";
23    echo "Warna $mobil->warna \n";
```

Penjelasan Skrip:

- **Baris 14 - 17**. Mengatur property `$merk`, `$tipe`, `$tahun` dan `$warna`. Object `$mobil` sudah memiliki nilai default (Baris 5 - 8). Kode pada Baris 14 - 17 akan mengubah nilai default property-property tersebut.
- **Baris 20 - 23**. Menampilkan nilai property.

Jalankan skrip `latihan.php`, maka hasilnya dapat dilihat pada gambar 1.3.



Gambar 1.3 Mengatur nilai property

1.6 Menambahkan Method dan Menjalankannya

Anda pasti sudah tahu tentang fungsi, bukan? Kita bisa membuat suatu fungsi di dalam sebuah class. Nah dalam konteks OOP, kita menyebut **fungsi yang ada di dalam suatu class sebagai *method***. Untuk membuat method:

Ubah `latihan.php` menjadi:

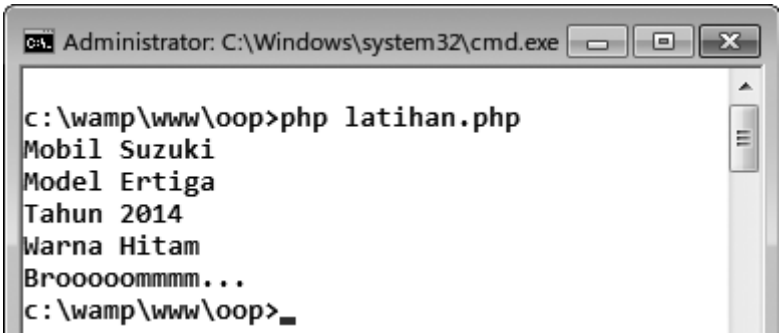
```
1     <?php
2
3     class Mobil
4     {
5         public $merk = 'Toyota';
6         public $tipe = 'Fortuner';
7         public $tahun = 2016;
8         public $warna = 'Putih';
9
10        public function maju()
11        {
12            echo 'Broooooommmm...';
13        }
14    }
```

```
15
16    $mobil = new Mobil();
17
18    // Mengatur nilai property
19    $mobil->merk = 'Suzuki';
20    $mobil->tipe = 'Ertiga';
21    $mobil->tahun = 2014;
22    $mobil->warna = 'Hitam';
23
24    // Mencetak nilai property
25    echo "Mobil $mobil->merk \n";
26    echo "Model $mobil->tipe \n";
27    echo "Tahun $mobil->tahun \n";
28    echo "Warna $mobil->warna \n";
29
30    // Menjalankan method maju()
31    $mobil->maju();
```

Penjelasan Skrip:

- **Baris 10 - 13**. Membuat method `maju()`.
- **Baris 31**. Untuk menjalankan method `maju()`, caranya hampir sama dengan mengakses nilai property. Sebutkan saja nama method-nya, disertai pula tanda `()` di belakang nama method tersebut.

Jalankan skrip `latihan.php`, maka hasilnya dapat dilihat pada gambar 1.4.



Gambar 1.4 Membuat dan menjalankan method

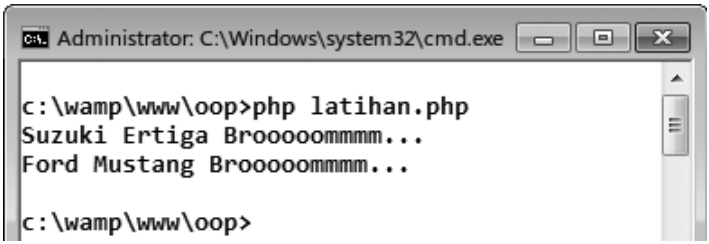
1.7 Membuat Beberapa Object dari Suatu Class

Seperti yang sudah disebutkan bahwa class adalah cetak biru / rancangan / cetakan dari suatu object, maka kita dapat membuat banyak object dari suatu class.

Untuk membuat beberapa object dari suatu class. Ubah `latihan.php` menjadi:

```
1     <?php
2
3     class Mobil
4     {
5         public $merk = '';
6         public $tipe = '';
7
8         public function maju()
9         {
10            return "Brooooommmm...\n";
11        }
12    }
13
14    // Membuat dan mengatur property object 1
15    $mobil1 = new Mobil();
16    $mobil1->merk = 'Suzuki';
17    $mobil1->tipe = 'Ertiga';
18
19  // Mencetak property & menjalankan method maju() object 1
20    echo $mobil1->merk . " " . $mobil1->tipe . " "
          .$mobil1->maju();
21
22    // Membuat dan mengatur property object 2
23    $mobil2 = new Mobil();
24    $mobil2->merk = 'Ford';
25    $mobil2->tipe = 'Mustang';
26
27  // Mencetak property & menjalankan method maju() object 2
28    echo $mobil2->merk . " " . $mobil2->tipe . " "
           .$mobil2->maju();
```

Jalankan skrip `latihan.php`, maka hasilnya dapat dilihat pada gambar 1.5.



Gambar 1.5 Beberapa object dari suatu class

Dengan OOP, kita dapat menciptakan beberapa instance object dari suatu class. Dengan demikian, **kode program yang kita buat pada suatu class bersifat *reusable***, dapat digunakan kembali. Sehingga pembuatan kode lebih efisien.