

Hari 9 – OOP PHP

by Garuda Cyber Institute

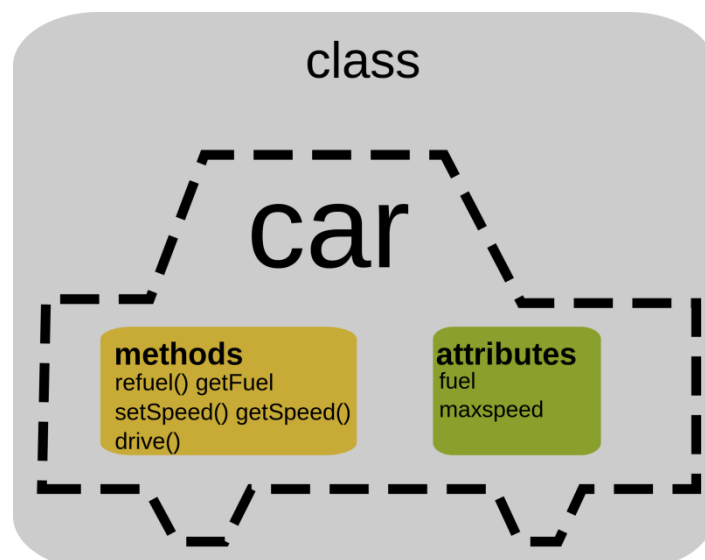
Apa itu OOP?

OOP (Object Oriented Programming) atau pemrograman berbasis objek merupakan paradigma pemrograman yang berorientasikan objek. Semua data dan fungsi di dalam paradigma ini dibungkus dalam kelas-kelas atau objek-objek.

Untuk mempermudah ilustrasi dalam belajar konsep OOP, saya akan ambil contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

Di dalam OOP, kita dapat memecahkan persoalan-persoalan dalam program dengan memecah masalah ke dalam class-class yang lebih kecil dan simpel agar solusi yang dibuat lebih spesifik.

Setiap class dalam OOP mempunyai method atau fungsi serta property atau atribut. method adalah kemampuan dari class untuk melakukan sesuatu. sedangkan property adalah segala sesuatu yang dimiliki oleh class.



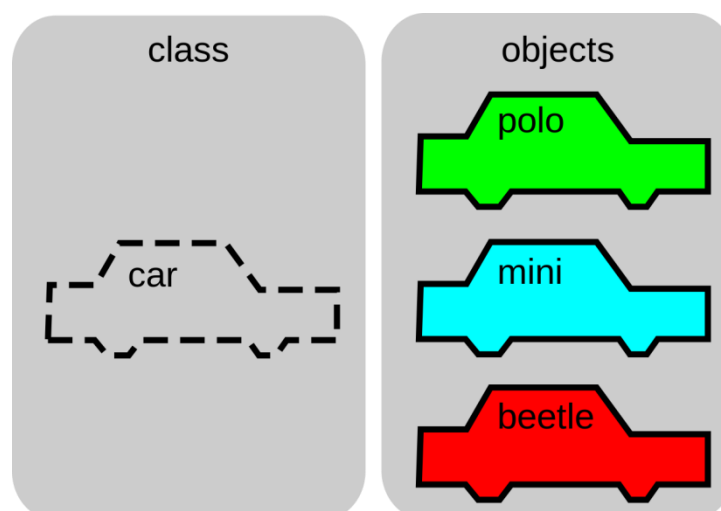
Kelebihan Konsep OOP

Konsep OOP telah menjadi standar dalam dunia pemrograman saat ini, dan telah menggeser konsep pemrograman prosedural yang memecah alur program menjadi beberapa prosedur dalam bentuk fungsi. OOP memiliki beberapa keunggulan dibandingkan pemrograman prosedural antara lain

1. OOP lebih cepat dan mudah untuk dieksekusi
2. OOP menyediakan struktur yang jelas untuk Program kita
3. OOP membuat Codingan kita Tidak dideklarasikan secara berulang-ulang, sehingga membuat Codingan kita lebih mudah untuk diedit, didebug, atau dimodifikasi
4. OOP memungkinkan kita membuat aplikasi yang dapat digunakan kembali dengan syntax yang lebih sedikit dan waktu developing yang lebih cepat.

Class dan Object

class adalah cetakan atau blueprint dari objek. Di dalam class terdapat property dan method. contohnya di bawah ini, terdapat class car yang merupakan cetakan dari objek-objek mobil. Pada gambar sebelumnya class car bisa memiliki method yaitu `refuel()`, `getSpeed()`, `setSpeed()`, `drive()` dan memiliki property `fuel`, `maxspeed`.



Contoh class



```
1 <?php
2 class Mobil {
3     public $roda = 4;
4     public function jalan() {
5         echo "Mobil berjalan";
6     }
7 }
8 ?>
```

Pada contoh di atas, terdapat class Mobil yang di dalamnya terdapat method `jalan()` dan property `$roda`.

Membuat Object (Intansiasi)

pada penjelasan sebelumnya bahwa class merupakan blueprint atau cetakan dari objek. Untuk membuat objek dari cetakan tersebut kita harus melakukan instansiasi atau pembuatan objek. caranya adalah seperti berikut:

```
$mini = new Mobil();
```

Pada contoh di atas dibuat sebuah object baru dengan nama `$mini` yang merupakan hasil instansiasi dari class Mobil. Karena `$mini` dihasilkan dengan cetakan mobil maka dia memiliki property dan method yang sama dengan class Mobil. Cara memanggil property dan method yaitu dengan tanda panah `->`.

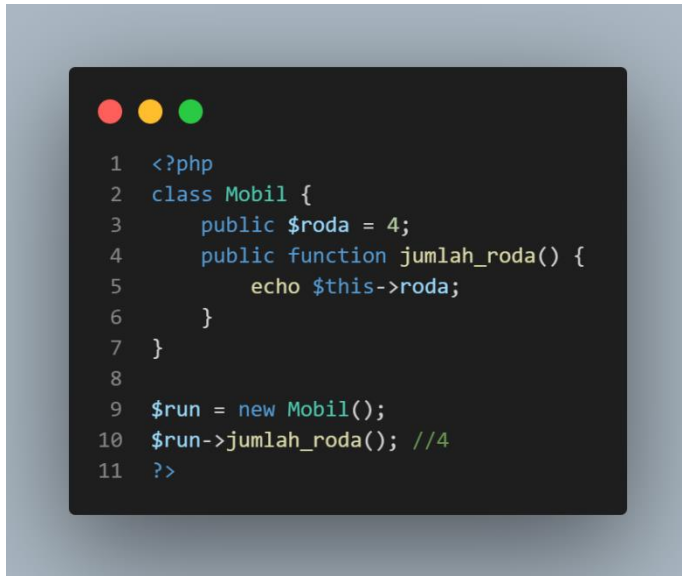


```
1 $run = new Mobil();
2 $run->jalan(); //Menampilkan echo 'Mobil berjalan'
3 echo $run->roda; //4
```

`$this` pada Class

Di dalam class kita akan sering menulis `$this` yang berarti merujuk kepada class itu sendiri. scope atau ruang lingkup `$this` adalah segala sesuatu yang ada di dalam tanda kurung kurawal { } setelah penamaan class NamaClass.

contohnya seperti berikut



```
1  <?php
2  class Mobil {
3      public $roda = 4;
4      public function jumlah_roda() {
5          echo $this->roda;
6      }
7  }
8
9  $run = new Mobil();
10 $run->jumlah_roda(); //4
11 ?>
```

pada contoh di atas, di dalam function `jumlah_roda()` dipanggil `$this->roda` yang merujuk kepada property `$roda` pada class `Mobil`

Melindungi Method Dan Property Dengan Enkapsulasi (Encapsulation)

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’. ada 3 jenis enkapsulasi yang bisa anda manfaatkan yaitu Public, Private Protected.

Belajar OOP PHP : Jenis Enkapsulasi Public

jika anda menggunakan jenis enkapsulasi public pada properti atau method dalam sebuah class, itu artinya method dan propertinya bisa di akses secara bebas baik dari dalam class, dari luar class bahkan dari class turunan nya sekalipun. berikut ini adalah contoh nya :

```
<?php
```

```
class person {
```

```
    public $name;
```

```
    function set_name($new_name) {
```

```
        $this->name = $new_name;
```

```
    }
```

```
}
```

```
function get_name() {
```

```
    return $this->name;
```

```
}
```

```
}
```

```
?>
```

```
<?php
```

```
$person1 = new Person();
```

```
// properti bisa di akses secara langsung
```

```
echo "Hai ".$person1->name='Nuris Akbar';
```

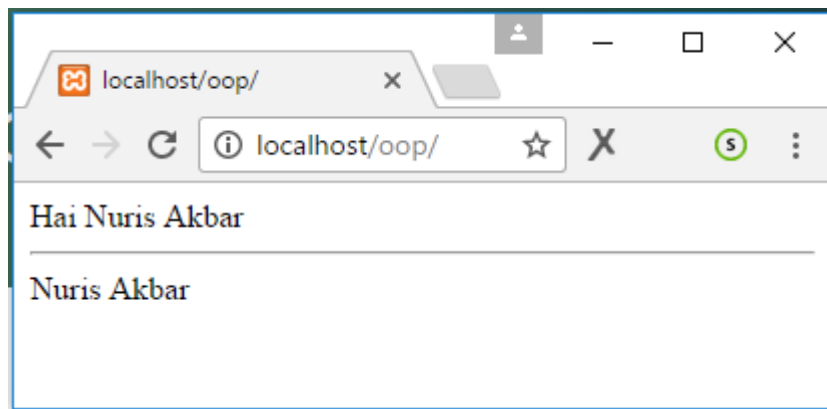
```
echo "<hr>";
```

```
// method bisa di akses secara langsung
```

```
echo $person1->get_name();
```

```
?>
```

seperti yang anda lihat pada script di atas bahwa kita bisa mengakses method atau properti secara langsung jika menggunakan keyword public.



Belajar OOP PHP : Jenis Enkapsulasi Private

lain halnya jika anda menggunakan jenis enkapsulasi private, itu artinya method dan propertinya hanya bisa di akses dari dalam class itu sendiri dan dari class turunan nya, berikut ini adalah contoh nya :

```
<?php
```

```
class person {
```

```
    private $name;
```

```
function set_name($new_name) {
```

```
    $this->name = $new_name;
```

```
}
```

```
function get_name() {
```

```
    return $this->name;
```

```
}
```

```
}
```

```
?>
```

```
<?php
```

```
$person1 = new Person();
```

```
// properti bisa di akses secara langsung
```

```
echo "Hai ".$person1->name='Nuris Akbar';
```

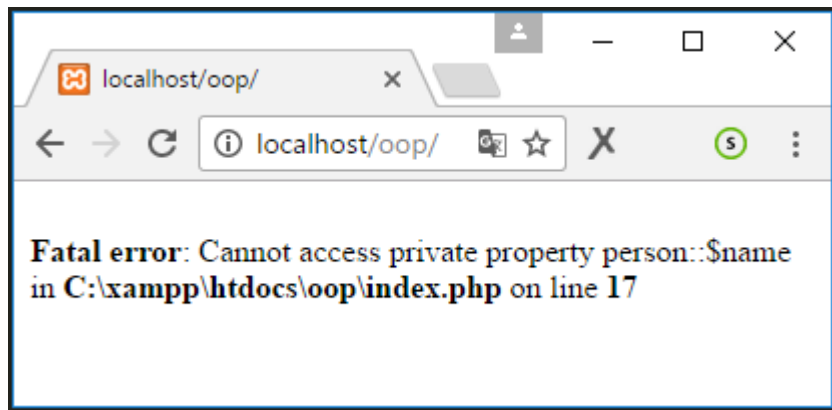
```
echo "<hr>";
```

```
// method tidak bisa di akses secara langsung
```

```
echo $person1->get_name();
```

```
?>
```

Jika anda menggnati jenis enkapsulasinya menjadi private maka akan muncul error seperti ini jika anda melakukan akses properti dari luar class.



Belajar OOP PHP : Jenis Enkapsulasi Protected

Lalu yang terakhir adalah jenis enkapsulasi protected, jenis enkapsulasi ini biasanya digunakan untuk melindungi informasi yang bersifat rahasia, dimana jika anda menggunakan protected pada method atau properti maka anda hanya bisa mengakses properti tersebut hanya dari dalam class tersebut, berikut contoh script nya :

```
<?php
```

```
class person {
```

```
protected $name;
```

```
function set_name($new_name) {
```

```
$this->name = $new_name;
```



```
}

```

```


```

```
function get_name() {

```

```
    return $this->name;

```

```

}
```

```

}
```

```
?>

```

```
<?php

```

```
$person1 = new Person();

```

```
// set value dari properti name

```

```
$person1->set_name('Muhammad Hafidz Muzaki');

```

```
// akses value dari properti name

```

```
echo $person1->get_name();

```

```
// properti tidak bisa di akses secara langsung, kana muncul error

```

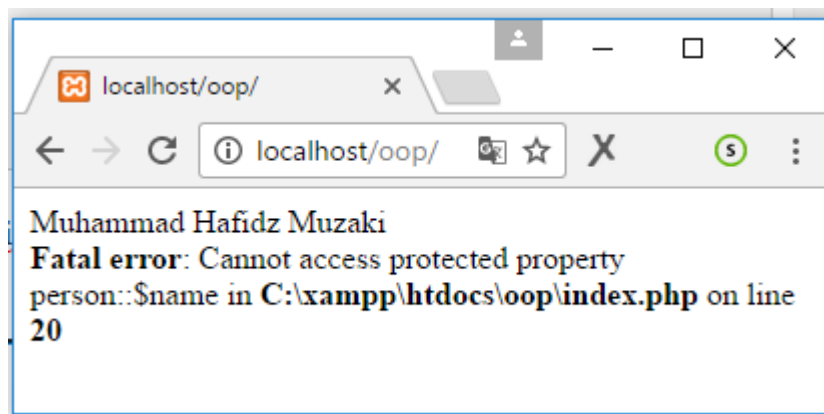
```
echo "Hai ".$person1->name='Nuris Akbar';

```

```
echo "<hr>";
```

```
?>
```

seperti yang saya jelaskan di atas bahwa perbedaan utama antara private dan protected hanyalah jika protected tidak bisa diakses kecuali dari dalam class itu sendiri. jika anda mengakses nya dari luar class maka akan muncul error seperti ini :



Visibilitas pada OOP

Dalam PHP, visibilitas dibagi menjadi 4 yaitu private, protected, public dan default. visibilitas digunakan untuk mengatur hak akses terhadap property dan method pada class. Hal ini dimaksudkan agar menjamin keamanan informasi yang terdapat pada property maupun method.

private

property atau method dengan visibilitas private maka property atau method tersebut hanya dapat diakses dari lingkup class dimana property atau method tersebut didefinisikan. contohnya:



```
1  <?php
2  class Mobil {
3      public $roda = 4;
4      public function jalan() {
5          echo "Mobil berjalan";
6      }
7  }
8
9  $run = new Mobil();
10 $run->jalan();
11 echo PHP_EOL;
12 echo $run->roda;
13 echo PHP_EOL;
14 ?>
```

bila program di atas dijalankan, maka akan muncul error **PHP Fatal error: Uncaught Error: Call to private method Mobil::jalan() from context** . Error tersebut muncul karena method jalan() hanya boleh diakses di dalam class Mobil dan tidak bisa diakses dari luar.

protected

method atau property yang diberikan visibilitas protected maka method atau property tersebut dapat diakses dari lingkup class dimana property atau method tersebut didefinisikan dan pada class turunan (inheritance) dari class tersebut.

```

1  <?php
2  class Mobil {
3      protected $roda = 4;
4  }
5
6  class MobilSport extends Mobil {
7      protected $maxSpeed;
8  }
9
10 $ferrari = new MobilSport();
11 echo $ferrari->roda(); //4
12 ?>

```

pada contoh di atas, class MobilSport merupakan inheritance atau turunan dari class Mobil. property \$roda yang dimiliki oleh class Mobil diturunkan ke class MobilSport dan tetap bisa dipakai di class MobilSport jika menggunakan visibilitas protected.

public

Jika property atau method diberikan visibilitas public maka method atau property tersebut dapat diakses baik dari lingkup class maupun object yang sudah diinstansiasi.

```

1  <?php
2  class Mobil {
3      public $roda = 4;
4      public function jumlahRoda() {
5          echo $this->roda;
6      }
7  }
8
9  $kijang = new Mobil();
10 $kijang->jumlahRoda(); //4
11 ?>

```

Constructor

constructor pada class yaitu method atau function yang akan dipanggil pertama kali ketika class tersebut diinstansiasi menjadi object. untuk membuat constructor kita buat method dengan nama __construct(). contohnya sebagai berikut:



```
1 <?php
2 class Mobil {
3     protected $roda = 4;
4     public $merk;
5     public function __construct($merk) {
6         $this->merk = $merk;
7     }
8 }
9
10 $avanza = new Mobil('Avanza');
11 echo $avanza->merk(); //Avanza
12 ?>
```

pada contoh di atas, property \$merk pada class Mobil hanya didefinisikan tanpa diberikan nilai. Lalu pada method construct dilakukan assign nilai merk diisi dengan parameter \$merk pada method construct tersebut.

Ketika object \$xeniya dibuat, cara instansiasinya adalah dengan mengetik `new Mobil("Xeniya")`. parameter “Xeniya” pada instansiasi tersebut adalah parameter yang akan terbaca pada metode construct sehingga object \$xeniya memiliki property \$merk yang bernilai “Xeniya”.