

PHP – Object Oriented Programming

1. Pengenalan OOP

a. Apa itu Object Oriented Programming?

- Object Oriented Programming adalah sudut pandang bahasa pemrograman yang berkonsep “objek”
- Ada banyak sudut pandang bahasa pemrograman, namun OOP adalah yang sangat populer saat ini.
- Ada beberapa istilah yang perlu dimengerti dalam OOP, yaitu: Object dan Class

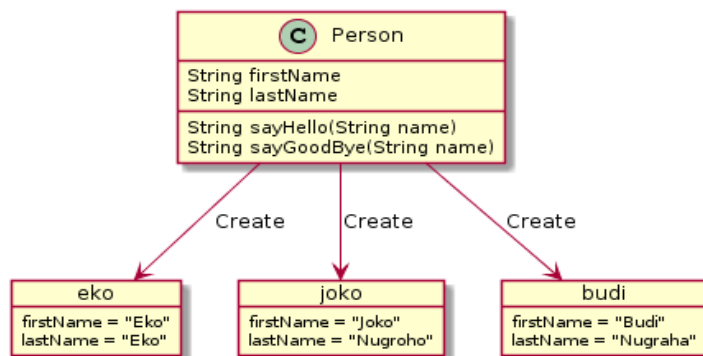
b. Apa itu Object?

Object adalah data yang berisi field / properties / attributes dan method / function / behavior

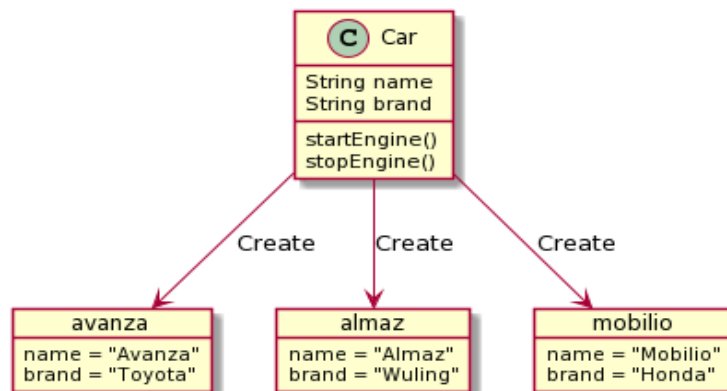
c. Apa itu Class?

- Class adalah blueprint, prototype atau cetakan untuk membuat Object
- Class berisikan deklarasi semua properties dan functions yang dimiliki oleh Object
- Setiap Object selalu dibuat dari Class
- Dan sebuah Class bisa membuat Object tanpa batas

Class dan Object : Person



Class dan Object : Car



2. Membuat Class

- Untuk membuat class, kita bisa menggunakan kata kunci class
- Penamaan class biasa menggunakan format CamelCase

Buat proyek (folder) dengan nama web-oop, buka teks editor lalu buat file dengan nama Mobil.php

```
Mobil.php > ...  
1  <?php  
2  
3  class Mobil  
4  {  
5  
6  }  
7  
8  ?>
```

3. Membuat Object

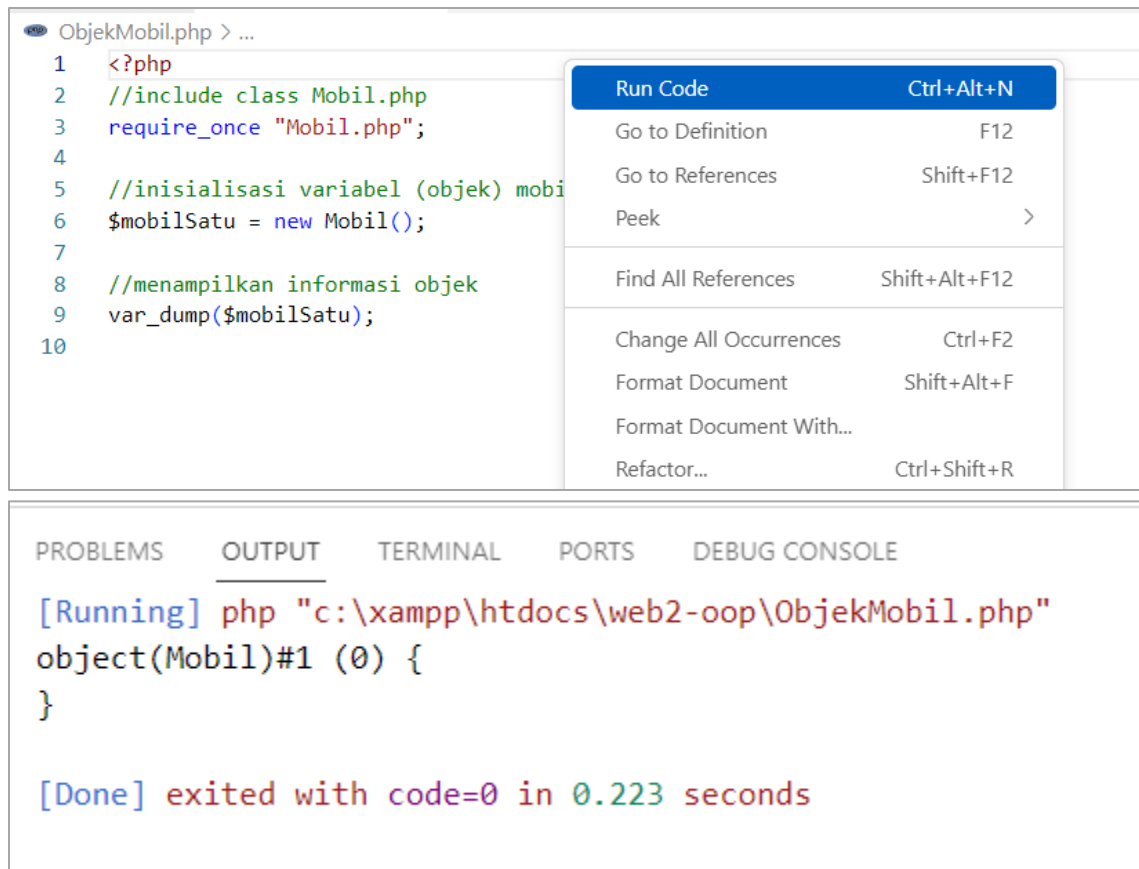
- Object adalah hasil instansiasi dari sebuah class
- Untuk membuat object kita bisa menggunakan kata kunci new, dan diikuti dengan nama Class dan kurung ()

Buat sebuah file baru dengan nama objekMobil.php

```
ObjekMobil.php > ...  
1  <?php  
2  //include class Mobil.php  
3  require_once "Mobil.php";  
4  
5  //inisialisasi variabel (objek) mobilSatu  
6  $mobilSatu = new Mobil();  
7  
8  //menampilkan informasi objek  
9  var_dump($mobilSatu);
```

Nb : install extention **code running** pada vs code untuk menjalankan code program

Pemrograman Berbasis Web 2



```
ObjekMobil.php > ...
1  <?php
2  //include class Mobil.php
3  require_once "Mobil.php";
4
5  //inisialisasi variabel (objek) mobil
6  $mobilSatu = new Mobil();
7
8  //menampilkan informasi objek
9  var_dump($mobilSatu);
10
```

Run Code Ctrl+Alt+N

Go to Definition F12

Go to References Shift+F12

Peek >

Find All References Shift+Alt+F12

Change All Occurrences Ctrl+F2

Format Document Shift+Alt+F

Format Document With...

Refactor... Ctrl+Shift+R

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"

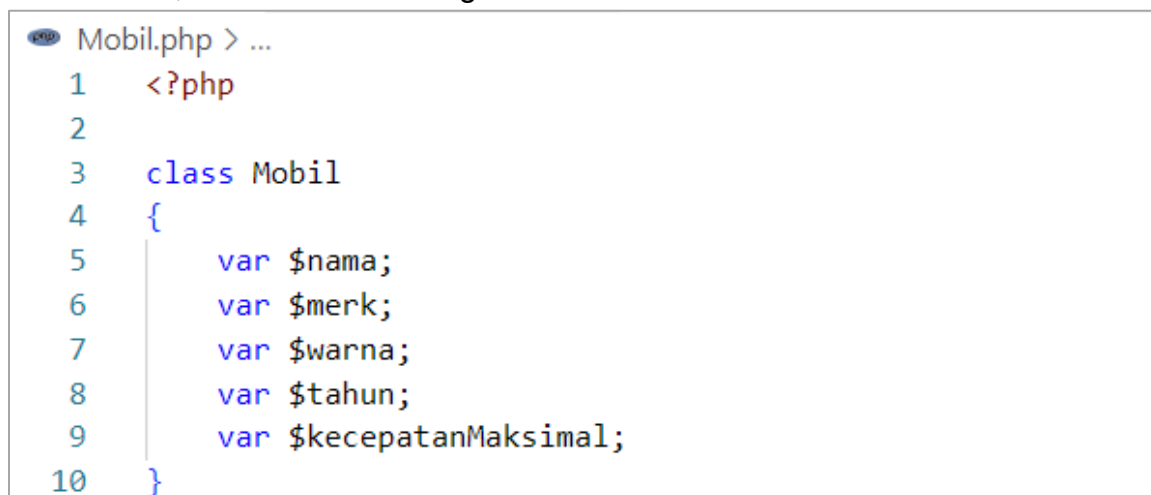
object(Mobil)#1 (0) {

}

[Done] exited with code=0 in 0.223 seconds

4. Properties

- Fields / Properties / Attributes adalah data yang bisa kita sisipkan di dalam Object
- Namun sebelum kita bisa memasukkan data di fields, kita harus mendeklarasikan data apa aja yang dimiliki object tersebut di dalam deklarasi class-nya
- Membuat field sama seperti membuat variable, namun ditempatkan di block class, namun diawali dengan kata kunci var



```
Mobil.php > ...
1  <?php
2
3  class Mobil
4  {
5      var $nama;
6      var $merk;
7      var $warna;
8      var $tahun;
9      var $kecepatanMaksimal;
10 }
```

Pemrograman Berbasis Web 2

Manipulasi Properties

- Fields yang ada di object, bisa kita manipulasi.
- Untuk memanipulasi data field, sama seperti cara pada variable
- Untuk mengakses field, kita butuh kata kunci -> setelah nama object dan diikuti nama fields nya

```
ObjekMobil.php > ...
1  <?php
2  //include class Mobil.php
3  require_once "Mobil.php";
4  //inisialisasi variabel (objek) mobilSatu
5  $mobilSatu = new Mobil();
6
7  //menambahkan properties pada objek mobilSatu
8  $mobilSatu->nama = "Avanza";
9  $mobilSatu->merk = "Toyota";
10 $mobilSatu->warna = "Hitam";
11 $mobilSatu->tahun = 2019;
12 $mobilSatu->kecepatanMaksimal = 1300;
13
14 //menampilkan informasi objek mobilSatu
15 echo " Nama : $mobilSatu->nama" . PHP_EOL;
16 echo " Merk : $mobilSatu->nama" . PHP_EOL;
17 echo " Warna : $mobilSatu->nama" . PHP_EOL;
18 echo " Tahun: $mobilSatu->nama" . PHP_EOL;
19 echo " Kecepatan Maksimal : $mobilSatu->nama" . PHP_EOL;
20
21 //membuat objek mobilDua
22 $mobilDua = new Mobil();
23
24 //menambahkan properties pada objek mobilDua
25 $mobilDua->nama = "Brio";
26 $mobilDua->merk = "Honda";
27 $mobilDua->warna = "Merah";
28 $mobilDua->tahun = 2018;
29 $mobilDua->kecepatanMaksimal = 1000;
30
31 //menampilkan informasi objek mobilDua
32 var_dump($mobilDua);
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

```
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"
Nama : Avanza
Merk : Toyota
Warna : Hitam
Tahun: 2019
Kecepatan Maksimal : 1300
object(Mobil)#2 (5) {
  ["nama"]=>
  string(4) "Brio"
  ["merk"]=>
  string(5) "Honda"
  ["warna"]=>
  string(5) "Merah"
  ["tahun"]=>
  int(2018)
  ["kecepatanMaksimal"]=>
  int(1000)
}

[Done] exited with code=0 in 0.251 seconds
```

Properties Type Declaration

- Sama seperti di function, di properties pun, kita bisa membuat type declaration
- Ini membuat PHP otomatis mengecek tipe data yang sesuai dengan type declaration yang telah ditentukan
- Jika kita mencoba mengubah properties dengan type yang berbeda, maka otomatis akan error
- Ingat, bahwa PHP memiliki fitur type juggling, yang secara otomatis bisa mengkonversi ke tipe data lain
- Untuk menambahkan type declaration, kita bisa tambahkan setelah kata kunci var di properties

```
Mobil.php > ...  
1  <?php  
2  
3  class Mobil  
4  {  
5      var string $nama;  
6      var string $merk;  
7      var string $warna;  
8      var int $tahun;  
9      var int $kecepatanMaksimal;  
10 }
```

Default Properties Value

- Sama seperti variable, di properties juga kita bisa langsung mengisi value nya
- Ini mirip seperti default value, jadi jika tidak diubah di object, maka properties akan memiliki value tersebut

```
Mobil.php > ...  
1  <?php  
2  
3  class Mobil  
4  {  
5      var string $nama;  
6      var string $merk = "Toyota";  
7      var string $warna;  
8      var int $tahun;  
9      var int $kecepatanMaksimal;  
10 }
```

Nullable Properties

- Saat kita menambah type declaration di properties atau di function argument, maka secara otomatis kita tidak bisa mengirim data null ke dalam properties atau function argument tersebut

Pemrograman Berbasis Web 2

- Di PHP 7.4 dikenalkan nullable type, jadi kita bisa mengirim data null ke properties atau function arguments
- Caranya sebelum type declaration nya, kita bisa menambahkan tanda “?”

```
Mobil.php > ...
1  <?php
2
3  class Mobil
4  {
5      var string $nama;
6      var string $merk;
7      var ?string $warna = null;
8      var int $tahun;
9      var int $kecepatanMaksimal;
10 }
```

```
7  //menambahkan properties pada objek mobilSatu
8  $mobilSatu->nama = null;
9  $mobilSatu->merk = "Toyota";
10 $mobilSatu->warna = null;
11 $mobilSatu->tahun = 2019;
12 $mobilSatu->kecepatanMaksimal = 1300;
```

Error, karena pada deklarasi tidak dibuat null

Valid, karena sudah diklarasikan boleh null

5. Function

- Selain menambahkan properties, kita juga bisa menambahkan function ke object
- Cara dengan mendeklarasikan function tersebut di dalam block class
- Sama seperti function biasanya, kita juga bisa menambahkan return value dan parameter
- Untuk mengakses function tersebut, kita bisa menggunakan tanda -> dan diikuti dengan nama method nya. Sama seperti mengakses properties

```
Mobil.php > ...
1  <?php
2
3  class Mobil
4  {
5      var string $nama;
6      var string $merk;
7      var ?string $warna = null;
8      var int $tahun;
9      var int $kecepatanMaksimal;
10
11      function tambahKecepatan()
12      {
13          echo "Kecepatan Bertambah!";
14      }
15 }
```

Membuat function tambahKecepatan

Pemrograman Berbasis Web 2

Mengakses function pada objek (class objekMobil.php)

```
21 //mengakses function/method
22 $mobilSatu->tambahKecepatan();
```

output

```
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"
Nama : Avanza
Merk : Toyota
Warna :
Tahun: 2019
Kecepatan Maksimal : 1300
Kecepatan Bertambah!
[Done] exited with code=0 in 0.252 seconds
```

6. This Keyword

- Saat kita membuat kode di dalam function di dalam class, kita bisa menggunakan kata kunci this untuk mengakses object saat ini
- Misal kadang kita butuh mengakses properties atau function lain di class yang sama

Membuat function infoMobil() pada class Mobil.php

```
Mobil.php > ...
16 function infoMobil(?string $nama)
17 {
18     //kondisi jika parameter kosong maka akan ditampilkan properties dari objek
19     if (is_null($nama)) {
20         echo "Nama Mobil adalah {$this->nama}" . PHP_EOL;
21         //jika parameter memiliki nilai maka yang ditampilkan adalah parameter pada function
22     } else {
23         echo "Mobil Anda adalah $nama" . PHP_EOL;
24     }
25 }
```

Mengakses function infoMobil() dengan parameter null, maka yang ditampilkan adalah properties dari objekMobil

```
ObjekMobil.php > ...
24 //mengakses function infoMobil()
25 $mobilSatu->infoMobil(null);
```

Output

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"
Nama : Avanza
Merk : Toyota
Warna :
Tahun: 2019
Kecepatan Maksimal : 1300
Kecepatan Bertambah!
Nama Mobil adalah Avanza
[Done] exited with code=0 in 0.3 seconds
```

Pemrograman Berbasis Web 2

Mengakses function infoMobil() dengan memiliki nilai, maka yang ditampilkan adalah properties dari function yang diakses pada saat ini

```
ObjekMobil.php > ...  
24 //mengakses function infoMobil()  
25 $mobilSatu->infoMobil("Xenia");
```

Output

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE  
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"  
Nama : Avanza  
Merk : Toyota  
Warna :  
Tahun: 2019  
Kecepatan Maksimal : 1300  
Kecepatan Bertambah!  
Mobil Anda adalah Xenia  
[Done] exited with code=0 in 0.253 seconds
```

7. Constant

- Properties di class bisa diubah, mirip seperti variable
- Di class juga kita membuat constant, data yang tidak bisa diubah
- Di materi PHP Dasar, kita belajar untuk membuat constant itu perlu menggunakan function define()
- Namun sejak PHP 7.4, kita bisa menggunakan kata kunci const untuk membuat constant, mirip seperti variable, namun tidak menggunakan karakter \$

```
Mobil.php > Mobil > infoMobil  
1 <?php  
2  
3 class Mobil  
4 {  
5     const AUTHOR = "Ari PW";
```

Memanggil constant

```
ObjekMobil.php > ...  
27 //memanggil dan menampilkan constant  
28 echo Mobil::AUTHOR;
```

Output

```
PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE  
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil.php"  
Nama : Avanza  
Merk : Toyota  
Warna :  
Tahun: 2019  
Kecepatan Maksimal : 1300  
Kecepatan Bertambah!  
Mobil Anda adalah Xenia  
Ari PW  
[Done] exited with code=0 in 0.338 seconds
```


Properties vs Constant

- Saat kita membuat object, properties yang terdapat di class akan secara otomatis dibuat per object, oleh karena itu untuk mengakses properties, kita perlu menggunakan object, atau jika dari dalam object tersebut sendiri, kita perlu menggunakan kata kunci this
- Sedangkan berbeda dengan constant, constant di class tidak akan dibuat per object. Constant itu hidupnya di class, bukan di object, oleh karena itu untuk mengaksesnya kita perlu menggunakan NamaClass::NAMA_CONSTANT
- Secara sederhana, properties akan dibuat satu per instance class (object), sedangkan constant dibuat satu per class

8. Constructor

- Saat kita membuat Object, maka kita seperti memanggil sebuah function, karena kita menggunakan kurung ()
- Di dalam class PHP, kita bisa membuat constructor, constructor adalah function yang akan dipanggil saat pertama kali Object dibuat.
- Mirip seperti di function, kita bisa memberi parameter pada constructor
- Nama constructor di PHP haruslah __construct()

Membuat function construct

```
1  <?php
2  class Mobil
3  {
4      var string $nama;
5      var ?string $merk = null;
6      var int $tahun;
7
8      public function __construct(string $nama, ?string $merk)
9      {
10         $this->nama = $nama;
11         $this->merk = $merk;
12     }
13
14     public function infoMobil()
15     {
16         return "$this->nama, $this->merk, $this->tahun;";
17     }
18 }
```

Membuat objek baru dengan mengakses function construct

```
1  <?php
2  require_once "Mobil.php";
3
4  $mobilSatu = new Mobil("Avanza", "Toyota");
5  $mobilSatu->tahun = 2019;
6
7  $mobilDua = new Mobil("Brio", null);
8  $mobilDua->tahun = 2020;
9
10 echo $mobilSatu->infoMobil() . PHP_EOL;
11 echo $mobilDua->infoMobil();
```

Pemrograman Berbasis Web 2

output

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop\ObjekMobil1.php"
Avanza, Toyota, 2019;
Brio, , 2020;
[Done] exited with code=0 in 0.329 seconds
```

TUGAS

Buatlah Class Mahasiswa dan Objek dari class Mahasiswa, tambahkan properties dan function yang ada pada Objek Mahasiswa.

PHP – Object Oriented Programming (Lanjutan)

1. Inheritance

- Inheritance atau pewarisan adalah kemampuan untuk menurunkan sebuah class ke class lain
- Dalam artian, kita bisa membuat class Parent dan class Child
- Class Child, hanya bisa punya satu class Parent, namun satu class Parent bisa punya banyak class Child
- Saat sebuah class diturunkan, maka semua properties dan function yang ada di class Parent, secara otomatis akan dimiliki oleh class Child
- Untuk melakukan pewarisan, di class Child, kita harus menggunakan kata kunci `extends` lalu diikuti dengan nama class parent nya.

Buat sebuah folder/projek dengan nama `web2-oop2`, lalu buat file dengan nama `Mobil.php` (Class `Mobil` - Parent)

```
Mobil.php > ...
1  <?php
2  class Mobil
3  {
4      var string $nama;
5      var string $brand;
6      var int $tahun;
7
8      function __construct(string $nama, string $brand, int $tahun)
9      {
10         $this->nama = $nama;
11         $this->brand = $brand;
12         $this->tahun = $tahun;
13     }
14
15     function infoMobil()
16     {
17         return "Nama Mobil : $this->nama" . PHP_EOL
18             . "Brand : $this->brand " . PHP_EOL
19             . "Tahun : " . $this->tahun . PHP_EOL;
20     }
21 }
```

Buat file dengan nama `MobilSport.php` (Class `MobilSport` – Child)

```
MobilSport.php > ...
1  <?php
2
3  class MobilSport extends Mobil
4  {
5      public $turbo = false;
6
7      function jalankanTurbo()
8      {
9          $this->turbo = true;
10         return "Jalankan Turbo!";
11     }
12 }
```

Pemrograman Berbasis Web 2

Buat file dengan nama ObjekMobil.php untuk membuat objek dari class Mobil dan MobilSport

```
ObjekMobil.php > ...
1  <?php
2  require_once "Mobil.php";
3  require_once "MobilSport.php";
4
5  // objek dari class mobil
6  $avanza = new Mobil("Avanza", "Toyota", 2020);
7  echo $avanza->infoMobil();
8
9  // objek dari class mobilsport
10 $supra = new MobilSport("Supra", "Totoya", 2021);
11 echo $supra->infoMobil();
12 echo $supra->jalankanTurbo();
```

Output

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop2\ObjekMobil.php"
Nama Mobil : Avanza
Brand : Toyota
Tahun : 2020
Nama Mobil : Supra
Brand : Totoya
Tahun : 2021
Jalankan Turbo!
[Done] exited with code=0 in 0.305 seconds
```

2. Function Overriding

- Function overriding adalah kemampuan mendeklarasikan ulang function di child class, yang sudah ada di parent class
- Saat kita melakukan proses overriding tersebut, secara otomatis ketika kita membuat object dari class child, function yang di class parent tidak bisa diakses lagi

Tambahkan function infoMobil() pada class MobilSport seperti berikut

Pemrograman Berbasis Web 2

```
7     function jalankanTurbo()
8     {
9         $this->turbo = true;
10        return "Jalankan Turbo!";
11    }
12
13    function infoMobil()
14    {
15        return "Nama Mobil Sport: $this->nama" . PHP_EOL
16            . "Brand : $this->brand " . PHP_EOL
17            . "Tahun : " . $this->tahun . PHP_EOL
18            . "Turbo : " . $this->turbo = true . PHP_EOL;
19    }
20 }
```

Output

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop2\ObjekMobil.php"
Nama Mobil : Avanza
Brand : Toyota
Tahun : 2020
Nama Mobil Sport: Supra
Brand : Totoya
Tahun : 2021
Turbo : 1
Jalankan Turbo!
[Done] exited with code=0 in 0.234 seconds
```

3. Visibility

- Visibility / Access modifier adalah kemampuan properties, function dan constant dapat diakses dari mana saja
- Secara default, properties, function dan constant yang kita buat di dalam class bisa diakses dari mana saja, atau artinya dia adalah public
- Selain public, masih ada beberapa visibility lainnya
- Secara default kata kunci var untuk properties adalah sifatnya public

Access Level

Modifier	Class	Subclass	World
public	Y	Y	Y
protected	Y	Y	N
private	Y	N	N

Pemrograman Berbasis Web 2

Buat file dengan nama Product.php

```
1  <?php
2
3  class Product
4  {
5      //deklarasi variabel
6      private string $name;
7      private int $price;
8      //deklatasi konstruktor
9      public function __construct(string $name, int $price)
10     {
11         $this->name = $name;
12         $this->price = $price;
13     }
14 }
```

Buat file dengan nama ObjekProduct

```
ObjekProduct.php > ...
1  <?php
2
3  require_once "Product.php";
4  //membuat objek produk
5  $product = new product("Apple", 20000);
6
7  //menampilkan info name dan price
8  echo $product->name() . PHP_EOL;
9  echo $product->price() . PHP_EOL;
```

Pada baris ke 8 dan 9 akan ada error karena variabel yang akan ditampilkan memiliki visibility private, agar bisa menampilkan info name dan price salah satu caranya adalah bisa membuat function getName dan getPrice yang memiliki visibility public pada class product

```
Product.php > product
15     public function getName(): string
16     {
17         return $this->name;
18     }
19
20     public function getPrice(): int
21     {
22         return $this->price;
23     }
24 }
```

Pemrograman Berbasis Web 2

Menampilkan informasi name dan price pada class ObjekProduct

```
ObjekProduct.php > ...
1  <?php
2
3  require_once "Product.php";
4  //membuat objek produk
5  $product = new product("Apple", 20000);
6
7  //menampilkan info name dan price
8  echo $product->getName() . PHP_EOL;
9  echo $product->getPrice() . PHP_EOL;
```

Output

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop2\ObjekProduct.php"
Apple
20000

[Done] exited with code=0 in 0.293 seconds
```

Begitu juga ketika membuat class turunan, variabel dengan visibility private tidak bisa diakses pada class turunannya. Buat file ProdukTurunan.php untuk class Child

```
ProdukTurunan.php > ...
1  <?php
2  class ProdukTurunan extends product
3  {
4
5      public function info()
6      {
7          echo "Name $this->name" . PHP_EOL;
8          echo "Price $this->price" . PHP_EOL;
9      }
10 }
```

Agar bisa diakses maka perlu diubah untuk visibility pada variabel name dan price pada class Product

Pemrograman Berbasis Web 2

```
Product.php > product
1  <?php
2
3  class product
4  {
5      //deklarasi variabel
6      protected string $name;
7      protected int $price;
```

Membuat objek dari class tutunan dan nemampilkan informasi name dan price

```
ObjekProduct.php > ...
1  <?php
2
3  require_once "Product.php";
4  require_once "ProdukTurunan.php";
5
6  //membuat objek produk
7  $product = new product("Apple", 20000);
8
9  //menampilkan info name dan price
10 echo $product->getName() . PHP_EOL;
11 echo $product->getPrice() . PHP_EOL;
12
13 $product2 = new produkTurunan("Nanas", 30000);
14 $product2->info();
```

4. Getter dan Setter

Encapsulation

- Encapsulation artinya memastikan data sensitif sebuah object tersembunyi dari akses luar
- Hal ini bertujuan agar kita bisa menjaga agar data sebuah object tetap baik dan valid
- Untuk mencapai ini, biasanya kita akan membuat semua properties menggunakan access modifier private, sehingga tidak bisa diakses atau diubah dari luar
- Agar bisa diubah, kita akan menyediakan function untuk mengubah dan mendapatkan properties tersebut

Getter dan Setter

- Di PHP, proses encapsulation sudah dibuat standarisasinya, dimana kita bisa menggunakan Getter dan Setter method.
- Getter adalah function yang dibuat untuk mengambil data field
- Setter ada function untuk mengubah data field

Pemrograman Berbasis Web 2

Buat file dengan nama Category.php

```
Category.php > ...
1  <?php
2  class Category
3  {
4      private string $name;
5      private bool $expensive;
6
7      public function getName(): string
8      {
9          return $this->name;
10     }
11
12     public function setName(string $name): void
13     {
14         $this->name = $name;
15     }
16
17     public function isExpensive(): bool
18     {
19         return $this->expensive;
20     }
21
22     public function setExpensive(bool $expensive): void
23     {
24         $this->expensive = $expensive;
25     }
26 }
```

Berikutnya buat file ObjekCategory.php

```
ObjekCategory.php > ...
1  <?php
2  require_once "Category.php";
3
4  $objCategory = new Category();
5  //untuk instan variabel tidak lagi menggunakan seperti ini
6  // $objCategory->name="hanphone";
7  $objCategory->setName("Handphone");
8  $objCategory->setExpensive(true);
9
10 //mengakses data
11 echo "Name : {$objCategory->getName()}" . PHP_EOL;
12 echo "Expensive : {$objCategory->isExpensive()}" . PHP_EOL;
```

Output

```
PROBLEMS  OUTPUT  TERMINAL  PORTS  DEBUG CONSOLE
[Running] php "c:\xampp\htdocs\web2-oop2\ObjekCategory.php"
Name : Handphone
Expensive : 1

[Done] exited with code=0 in 0.324 seconds
```

5. Abstract Class

- Saat kita membuat class, kita bisa menjadikan sebuah class sebagai abstract class.
- Abstract class artinya, class tersebut tidak bisa dibuat sebagai object secara langsung, hanya bisa diturunkan
- Untuk membuat sebuah class menjadi abstract, kita bisa menggunakan kata kunci abstract sebelum kata kunci class
- Sehingga Abstract Class bisa kita gunakan sebagai kontrak child class

Buat file dengan nama Location.php

```
Location.php > ...
1  <?php
2  // parent class
3  abstract class Location
4  {
5      public string $name;
6  }
7
8  // child class
9  class City extends Location
10 {
11 }
12
13 class Province extends Location
14 {
15 }
16
17 class Country extends Location
18 {
19 }
```

Buat file objekLocation untuk membuat objek dari class abstrak

```
ObjekLoacation.php > ...
1  <?php
2  require_once "Location.php";
3
4  // membuat objek
5  $obj1 = new Location(); //error
6  $obj2 = new City();
```

6. Interface

- Sebelumnya kita sudah tahu bahwa abstract class bisa kita gunakan sebagai kontrak untuk class child nya.
- Namun sebenarnya yang lebih tepat untuk kontrak adalah Interface
- Jangan salah sangka bahwa Interface disini bukanlah User Interface
- Interface mirip seperti abstract class, yang membedakan adalah di Interface, semua method otomatis abstract, tidak memiliki block
- Di interface kita tidak boleh memiliki properties, kita hanya boleh memiliki constant
- Untuk mewariskan interface, kita tidak menggunakan kata kunci extends, melainkan implements
- Dan berbeda dengan class, kita bisa implements lebih dari satu interface

Buat file dengan nama Car.php

```
Car.php > ...
1  <?php
2
3  interface Car
4  {
5      function drive(): void;
6      function getTire(): int;
7  }
8
9  // implementasi interface pada class childnya
10 class Avanza implements Car
11 {
12     public function drive(): void
13     {
14         echo "Drive Avanza" . PHP_EOL;
15     }
16
17     public function getTire(): int
18     {
19         return 4;
20     }
21 }
```

Buat file dengan nama ObjekCar.php untuk mengakses class interface

```
ObjekCar.php > ...
1  <?php
2  require_once "Car.php";
3
4  $car = new Avanza();
```

7. Interface Inheritance

- Sebelumnya kita sudah tahu kalo di PHP, child class hanya bisa punya 1 class parent
- Namun berbeda dengan interface, sebuah child class bisa implement lebih dari 1 interface
- Bahkan interface pun bisa implement interface lain, bisa lebih dari 1. Namun jika interface ingin mewarisi interface lain, kita menggunakan kata kunci `extends`, bukan `implements`

Modifikasi class Car

```
Car.php > ...
1  <?php
2
3  interface HasBrand
4  {
5      function getBrand(): string;
6  }
7
8  interface IsMaintenance
9  {
10     function isMaintenance(): bool;
11 }
```

Menyatakan bahwa `Car` merupakan turunan dari class `HasBrand`, dan menambahkan function `getBrand()`

```
Car.php > ...
12
13 interface Car extends HasBrand
14 {
15     function drive(): void;
16     function getTire(): int;
17 }
18
19 // implementasi interface pada class childnya
20 class Avanza implements Car
21 {
22     public function drive(): void
23     {
24         echo "Drive Avanza" . PHP_EOL;
25     }
26
27     public function getTire(): int
28     {
29         return 4;
30     }
31     public function getBrand(): string
32     {
33         return "Toyota";
34     }
35 }
```

Menambahkan parent isMaintenance dan function isMaintenance()

```
Car.php > ...
20 class Avanza implements Car, IsMaintenance
21 {
22     public function drive(): void
23     {
24         echo "Drive Avanza" . PHP_EOL;
25     }
26
27     public function getTire(): int
28     {
29         return 4;
30     }
31     public function getBrand(): string
32     {
33         return "Toyota";
34     }
35     public function isMaintenance(): bool
36     {
37         return false;
38     }
39 }
```

8. Namespace

- Saat kita membuat aplikasi, bisa dipastikan kita akan banyak sekali membuat class
 - Jika class terlalu banyak, kadang akan menyulitkan kita untuk mencari atau mengklasifikasikan jenis-jenis class
 - PHP memiliki fitur namespace, dimana kita bisa menyimpan class-class kita di dalam namespace
 - Namespace bisa nested, dan jika kita ingin mengakses class yang terdapat di namespace, kita perlu menyebutkan nama namespace nya
 - Namespace bagus ketika kita punya beberapa class yang sama, dengan menggunakan namespace nama class sama tidak akan menjadikan error di PHP
- Buat file dengan nama Conflict.php dan kita buat dua class dengan nama Conflict pasti akan terjadi error

```
Conflict.php > ...
1 <?php
2 class Conflict{
3
4 }
5 class Conflict{
6
7 }
```

Pemrograman Berbasis Web 2

Buat file dengan nama Namespace.php

Output namespace

```
Namespace.php
1  <?php
2  require_once "Conflict.php";
3
```

PROBLEMS 2 OUTPUT TERMINAL PORTS DEBUG CONSOLE Code

[Running] php "c:\xampp\htdocs\web2-oop2\Namespace.php"

PHP Fatal error: Cannot declare class Conflict, because the name is already in use in C:\xampp\htdocs\web2-oop2\Conflict.php on line 5

Fatal error: Cannot declare class Conflict, because the name is already in use in C:\xampp\htdocs\web2-oop2\Conflict.php on line 5

[Done] exited with code=255 in 0.305 seconds

Membuat Namespace

- Untuk membuat namespace, kita bisa menggunakan kata kunci namespace
- Jika kita ingin membuat sub namespace, kita cukup gunakan karakter \ setelah namespace sebelumnya

```
Conflict.php > {} Data\Two > Conflict
1  <?php
2
3  namespace Data\One {
4      class Conflict
5      {
6      }
7  }
8
9  namespace Data\Two {
10     class Conflict
11     {
12     }
13 }
```

Output namespace

```
Namespace.php
1  <?php
2  require_once "Conflict.php";
3
```

PROBLEMS OUTPUT TERMINAL PORTS DEBUG CONSOLE

[Running] php "c:\xampp\htdocs\web2-oop2\Namespace.php"

[Done] exited with code=0 in 0.321 seconds

Membuat objek conflict

```
Namespace.php > ...  
1  <?php  
2  require_once "Conflict.php";  
3  $conflict1 = new Data\One\Conflict();  
4  $conflict2 = new Data\Two\Conflict();  
5
```