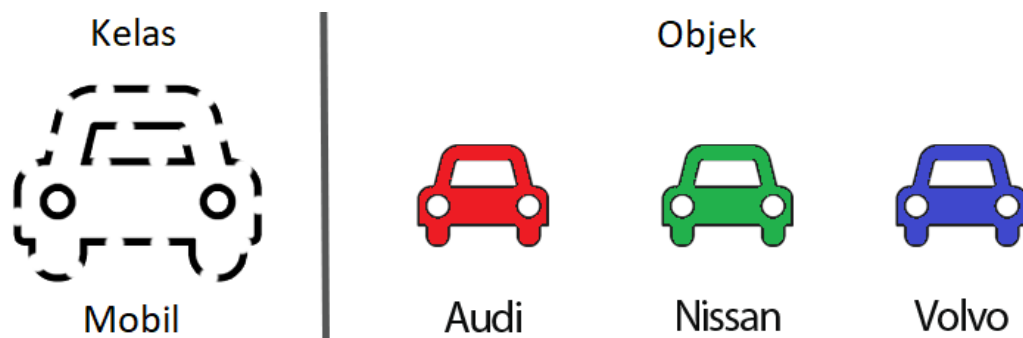


PHP Object Oriented

Object Oriented Programming (OOP) atau yang diterjemahkan sebagai pemrograman berbasis objek adalah sebuah paradigma pemrograman yang berbasis pada konsep kelas dan objek. Objek merupakan sebuah entitas yang memiliki properti di mana merepresentasikan informasi dan memiliki method yang merepresentasikan tingkah laku dari objek tersebut.

Kelas dalam OOP adalah sebuah cetak biru/blueprint dari objek. Kelas digunakan untuk mendefinisikan informasi apa yang akan dimiliki oleh sebuah objek dan apa yang dapat dilakukan oleh objek.



Seperti pada contoh di atas, kelas mobil digunakan untuk memproduksi berbagai objek misalnya Audi, Nissan, dan Volvo. Masing-masing objek ini memiliki properti yang berbeda dan objek yang berbeda. Mengubah properti objek Audi tidak akan mengubah properti objek Volvo semenjak mereka adalah objek yang berbeda.

A. Mendefinisikan Kelas

Sebuah kelas dapat definisikan menggunakan klausa `class` diikuti dengan nama kelas dan sepasang kurung kurawal `{}`. Semua method dan properti akan definisikan di dalam kurung kurawal tersebut.

Nama File	:	apel.php
Deskripsi	:	Membuat kelas Apel
<pre><?php class Apel{ } </pre>		

B. Mendefinisikan Properti

Pada contoh di bawah ini akan ditambahkan 2 properti untuk kelas Apel. Properti ini antara lain warna dan berat. Dua informasi ini merupakan informasi yang biasanya ditemukan pada sebuah objek apel. Properti didefinisikan dengan keyword "access modifier" diikuti dengan nama properti.

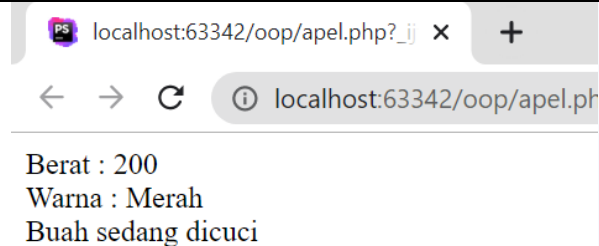
Nama File	:	apel.php
Deskripsi	:	Menambahkan properti

```
<?php

class Apel{
    // Properti
    private $warna;
    private $berat;
}
```

C. Mendefinisikan Method

Kemudian kita akan menambahkan method pada kelas Apel. Method yang ditambahkan antara lain setBerat() untuk mengatur berat, setWarna() untuk mengatur warna, info() untuk menampilkan informasi terkait objek, dan cuci() untuk merepresentasikan bahwa objek apel dapat dicuci.

Nama File	:	apel.php
Deskripsi	:	Menambahkan method
<pre><?php class Apel{ // Properti private \$warna; private \$berat; // Method public function setBerat(\$berat){ \$this->berat = \$berat; } public function setWarna(\$warna){ \$this->warna = \$warna; } public function info(){ echo "Berat : {\$this->berat}
"; echo "Warna : {\$this->warna}
"; } public function cuci(){ echo "Buah sedang dicuci
"; } } \$apel1 = new Apel(); \$apel1->setBerat(200); \$apel1->setWarna("Merah"); \$apel1->info(); \$apel1->cuci();</pre>		
		

Pada kode di atas terdapat inisialisasi kelas Apel ke dalam obyek apel1. Inisialisasi dapat dilakukan dengan menggunakan keyword new diikuti dengan nama kelas.

```
$variabel = new NamaKelas();
```

Kemudian kita mendefinisikan obyek bahwa berat memiliki nilai 200 dan warna memiliki nilai merah. Pemanggilan method info mencetak kedua informasi tersebut. Sedangkan pemanggilan method cuci akan mencetak teks "Buah sedang dicuci" pada browser.

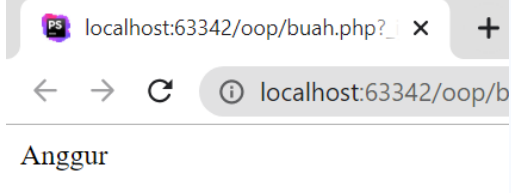
D. Penggunaan Keyword \$this

Keyword \$this digunakan untuk merujuk objek saat ini. Sehingga \$this->berat dapat digunakan untuk merujuk pada properti berat pada objek apel. Namun keyword \$this hanya dapat digunakan di dalam method. Lihat contoh di bawah ini!

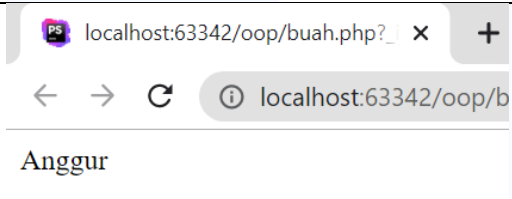
Nama File	:	buah.php
Deskripsi	:	Menjelaskan keyword \$this
<pre><?php class Buah{ public \$nama; }</pre>		

Jadi, di mana kita bisa mengubah nilai properti \$nama? Terdapat dua cara:

1. Di dalam kelas dengan menambahkan method setNama() dan menggunakan keyword \$this.

Nama File	:	buah.php
Deskripsi	:	Menjelaskan keyword \$this
<pre><?php class Buah{ public \$nama; public function setNama(\$nama){ \$this->nama = \$nama; } } \$buah = new Buah(); \$buah->setNama("Anggur"); echo \$buah->nama;</pre>		
		

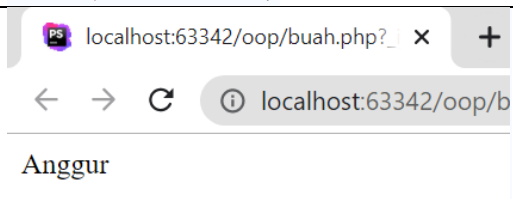
2. Di luar kelas dengan langsung mengubah nilai dari properti.

Nama File	: buah.php
Deskripsi	: Menjelaskan keyword \$this
<pre><?php class Buah{ public \$nama; } \$buah = new Buah(); \$buah->nama = "Anggur"; echo \$buah->nama;</pre>	
	

Jadi keyword this digunakan untuk merujuk pada objek saat ini. Sehingga dapat mengakses properti dirinya dari dalam method.

E. Method Konstruktur pada PHP

Konstruktur adalah sebuah method yang pertama kali dijalankan ketika sebuah objek diinisialisasi. Sebuah konstruktur menerima argumen yang dikirimkan saat objek diinisialisasi. Sehingga memungkinkan untuk mendefinisikan properti ketika objek dibuat.

Nama File	: apel.php
Deskripsi	: Menjelaskan method konstruktur
<pre><?php class Buah{ public \$nama; public function __construct(\$nama) { \$this->nama = \$nama; } } \$buah = new Buah("Anggur"); echo \$buah->nama;</pre>	
	

F. Apa Itu Access Modifier?

Properti dan metode dapat memiliki access modifier yang mengontrol di mana mereka dapat diakses. Pada bahasa pemrograman PHP terdapat tiga access modifier:

- **public** : properti atau metode dapat diakses dari mana saja. **Default.**
- **protected** : properti atau metode dapat diakses di dalam kelas dan kelas turunan.
- **private** : properti atau metode hanya dapat diakses di dalam kelas.

Nama File	: apel.php
Deskripsi	: Menjelaskan access modifier private
<pre><?php class Buah{ private \$nama; public function setName(\$nama){ \$this->nama = \$nama; } } \$buah = new Buah(); \$buah->setName("Anggur"); echo \$buah->nama;</pre>	

Pada contoh di atas, properti nama memiliki access modifier private. Sehingga hanya dapat diakses di dalam kelas. Percobaan akses terhadap properti nama dari luar kelas akan menghasilkan error.



G. Pewarisan Kelas

Suatu kelas dapat diturunkan dari kelas lain. Sehingga kelas anak mewarisi method dan properti dengan access modifier public dan protected dari kelas induk. Pewarisan didefinisikan menggunakan keyword extends. Lihat contoh berikut.

Nama File	: index.php
Deskripsi	: Pewarisan kelas
<pre><?php class Buah{ public function makan(){ echo "Buah sedang dimakan.
"; } } class Apel extends Buah{ private \$warna; private \$berat;</pre>	

```

    public function setBerat($berat) {
        $this->berat = $berat;
    }

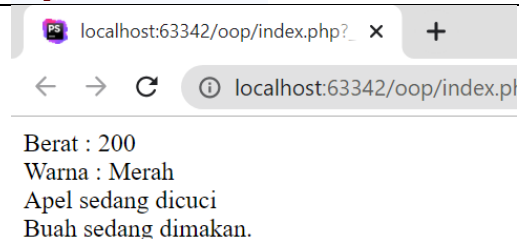
    public function setWarna($warna) {
        $this->warna = $warna;
    }

    public function info() {
        echo "Berat : {$this->berat} <br/>";
        echo "Warna : {$this->warna} <br/>";
    }

    public function cuci() {
        echo "Apel sedang dicuci <br/>";
    }
}

$apel1 = new Apel();
$apel1->setBerat(200);
$apel1->setWarna("Merah");
$apel1->info();
$apel1->cuci();
$apel1->makan();

```



localhost:63342/oop/index.php?_ x +

← → ↻ ⓘ localhost:63342/oop/index.pl

Berat : 200
Warna : Merah
Apel sedang dicuci
Buah sedang dimakan.

Pada contoh di atas. Dapat dilihat bahwa method makan() akan diwariskan dari kelas Buah ke kelas Apel. Sebuah method atau properti juga dapat di override di kelas anak. Memungkinkan untuk mengganti aksi dari method yang didefinisikan di kelas induk.

Nama File	:	index.php
Deskripsi	:	Pewarisan kelas
<pre> <?php class Buah{ public function makan() { echo "Buah sedang dimakan.
"; } } class Apel extends Buah{ private \$warna; private \$berat; public function setBerat(\$berat) { \$this->berat = \$berat; } public function setWarna(\$warna) { </pre>		

```
        $this->warna = $warna;
    }

    public function info() {
        echo "Berat : {$this->berat} <br/>";
        echo "Warna : {$this->warna} <br/>";
    }

    public function cuci() {
        echo "Apel sedang dicuci <br/>";
    }

    public function makan() {
        echo "Apel sedang dimakan. <br/>";
    }
}

$apel1 = new Apel();
$apel1->setBerat(200);
$apel1->setWarna("Merah");
$apel1->info();
$apel1->cuci();
$apel1->makan();
```

localhost:63342/oop/index.php?_ x +

← → ↻ ⓘ localhost:63342/oop/inde

Berat : 200
Warna : Merah
Apel sedang dicuci
Apel sedang dimakan.