

Laporan Praktikum Pengembangan Aplikasi Berbasis Internet



**Nama : Erik Dito Tampubolon
NIM : 13321030
Program Studi : D3 Teknologi Komputer**

**INSTITUT TEKNOLOGI DEL
FAKULTAS VOKASI**

Pengembangan Aplikasi Berbasis Internet

Minggu/Sesi	:	11 / 2
Kode Mata Kuliah	:	1331204
Nama Mata Kuliah	:	Pengembangan Aplikasi Berbasis Internet
Panduan Kuliah	:	Panduan ini dibuat untuk mengarahkan mahasiswa memahami mengenai Praktikum Pengembangan Aplikasi Berbasis Internet.
Setoran	:	Laporan praktikum
Batas Waktu Setoran	:	-
Topik	:	OOP menggunakan PHP.
Tujuan	:	Students Know the programming paradigm on OOP in php.

OOP dengan 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.

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 tetap bisa membuat situs web dengan PHP tanpa menggunakan objek sama sekali.

Dalam studi pemrograman, pembuatan program 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.

Pengertian Class, Object, Property dan Method

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.

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.

Pengertian Enkapsulasi Objek (Public, Protected dan Private)

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.

Enkapsulasi Objek: Public, Protected dan Private

Untuk membatasi hak akses kepada property dan method di dalam sebuah class, Objek 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.

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.

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.

OOP dengan PHP

Video 1 (OOP DASAR pada PHP #1 – Pendahuluan)

URL project github: https://github.com/ErikDitoTampubolon/php_oop.git

Object-oriented programming atau OOP adalah suatu metode pemrograman yang berorientasi pada objek. Program-program yang telah ada merupakan gabungan dari beberapa komponen-komponen kecil yang sudah ada sebelumnya. Hal itu dapat mempermudah pekerjaan seorang programmer dalam melakukan pengembangan program.

Objek-objek yang saling berkaitan dan disusun kedalam satu kelompok ini disebut dengan class. Nantinya, objek-objek tersebut akan saling berinteraksi untuk menyelesaikan masalah program yang rumit. Jika sebelumnya developer harus berfokus pada logic yang akan dimanipulasi, dengan OOP, developer dapat lebih terfokus pada objeknya saja untuk dimanipulasi.

Karakteristik Procedural Programming:

- Instruksi dilakukan step by step
- Memecah program menjadi bagian-bagian kecil (modularisasi)
- Bagian kecil tersebut disebut prosedur, subroutine atau function.
- Linear / Top-to-Bottom
- Bahasa pemrograman yang menggunakan procedural programming:
- Fortran, ALGOL, COBOL, Pascal, C, PHP, Javascript

Kelebihan Procedural Programming

- To-the-point
- Simplicity dan kemudahan implementasi (untuk compiler dan interpreter)
- Mudah ditelusuri
- Membutuhkan lebih sedikit memory (dibandingkan dengan OOP)

Karakteristik Object Oriented Programming

- Menyusun semua kode program dan struktur data sebagai objek.
- Objek adalah unit dasar dari program
- Objek menyimpan data dan perilaku

- Objek biasa saling berinteraksi
- Java, Ruby, Python, C++, Javascript, PHP5

Konsep OOP pada PHP

Basic

- Class dan Object
- Property dan Method
- Constructor
- Object Type
- Inheritance (pewarisan)
- Visibility / Access Modifier
- Setter dan Getter
- Static Method

Advanced

- Abstract dan Interface
- Interceptor
- Object Cloning
- Callbacks dan Closures
- Namespaces dan Autoloading

Video 2 (OOP DASAR pada PHP #2 - Class dan Object)

Class

- Template atau blueprint untuk membuat instance sebuah object
- Class mendefinisikan object
- Menyimpan data dan perilaku yang disebut dengan property dan method

Membuat class

- Diawali dengan menuliskan keyword class, diikuti nama dan dibatasi dengan {} untuk menyimpan property dan method
- Aturan penamaan class sama seperti variable.

Contoh script class

```
<?php
class Coba {
    public $a; //property

    //method
    public function b() {
    }
}
```

Object

- Instance yang didefinisikan oleh class
- Banyak object dapat dibuat menggunakan satu class
- Object dibuat dengan menggunakan keyword new

Contoh script object

```
<?php
class Coba {
}

//Membuat Object Instance Dari Class
$a = new Coba();
$b = new Coba();
```


Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Coba {

}

$a = new Coba();
$b = new Coba();
$c = new Coba();

var_dump($a);
var_dump($b);
var_dump($c);

?>
```

Hasil:

← → ↻ localhost:8080/PABI/php_oop/coba.php

C:\xampp\htdocs\PABI\php_oop\coba.php:14:
object(Coba)[1]

C:\xampp\htdocs\PABI\php_oop\coba.php:15:
object(Coba)[2]

C:\xampp\htdocs\PABI\php_oop\coba.php:16:
object(Coba)[3]

Video 3 (OOP DASAR pada PHP #3 - Property dan Method)

Property

- Merepresentasikan data / keadaan dari sebuah object
- Variabel yang ada di dalam object (member variable)
- Sama seperti variable di dalam PHP, ditambah dengan visibility di depannya

Method

- Merepresentasikan perilaku dari sebuah object
- Function yang ada di dalam object
- Sama seperti function di dalam PHP, ditambah dengan visibility di depannya

Contoh Kasus Studi Mobil

- Daftar property:
- Nama
- Merk
- Warna
- kecepatanMaksimal
- jumlahPenumpang

Daftar method:

- tambahKecepatan
- kurangiKecepatan
- gantiTransmisi
- belokKiri
- belokKanan

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer
```

```

// Jualan Produk
// Komik
// Game
class Produk {
    public $judul = "judul",
           $penulis = "penulis",
           $penerbit = "penerbit",
           $harga = 0;

    public function getLabel() {
        return "$this->penulis, $this->penerbit, $this->harga";
    }
}

// $produk1 = new Produk();
// $produk1->judul = "Naruto";
// var_dump($produk1);

// $produk2 = new Produk();
// $produk2->judul = "Uncharted";
// $produk2->tambahProperty = "tambah";
// var_dump($produk2);

$produk3 = new Produk();
$produk3->judul = "Naruto";
$produk3->penulis= "Masashi Kishimoto";
$produk3->penerbit= "Shonen Jump";
$produk3->harga= "30000";

$produk4 = new Produk();
$produk4->judul = "Uncharted";
$produk4->penulis= "Neil Druckman";
$produk4->penerbit= "Sony Computer";
$produk4->harga= "250000";

echo "Komik : " . $produk3->getLabel();
echo "<br>";
echo "Game : " . $produk4->getLabel();

?>

```

Hasil:

← → ↺ ⓘ localhost:8080/PABI/php_oop/produk.php

Komik : Masashi Kishimoto, Shonen Jump, 30000
 Game : Neil Druckman, Sony Computer, 250000

Video 4 (OOP DASAR pada PHP #4 - Constructor)

Constructor merupakan sebuah method yang spesial yang ada di dalam sebuah class. Constructor disebut spesial karena method otomatis dijalankan ketika sebuah class di instansiasi. Jadi ketika kita membuat sebuah objek menggunakan keyword “new”, secara otomatis ada sebuah method yang dijalankan.

Bukan hanya di PHP sebenarnya fitur constructor ini, namun ada di bahasa lain namun istilahnya saja yang berbeda. Kita ambil contoh pada bahasa program Golang dimana method yang di jalankan pertama kali ini main() atau init().

Cara Membuat Constructor

Format penulisan constructor pada prefix / awalan nama method menggunakan tanda underscore dua kali(__). Contohnya :

```
function __construct(){  
    // isinya  
}
```

Program:

```
<?php  
// Nama      : Erik Dito Tampubolon  
// NIM       : 13321030  
// Prodi     : D3 Teknologi Komputer  
  
// Jualan Produk  
// Komik  
// Game  
class Produk {  
    public $judul,  
           $penulis,  
           $penerbit,  
           $harga;  
  
    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =  
"penerbit", $harga = 0) {  
        $this->judul = $judul;  
        $this->penulis = $penulis;  
        $this->penerbit = $penerbit;  
        $this->harga = $harga;  
    }  
  
    public function getLabel() {  
        return "$this->penulis, $this->penerbit";  
    }  
}
```

```

}

$produk1 = new Produk("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000);
$produk2 = new Produk("Uncharted", "Neil Druckman", "Sony Computer", 250000);
$produk3 = new Produk("Dragon Ball");

echo "Komik : " . $produk1->getLabel();
echo "<br>";
echo "Game : " . $produk2->getLabel();
echo "<br>";
var_dump($produk3);

?>

```

Hasil:

← → ↻ ⓘ localhost:8080/PABI/php_oop/constructor.php

Komik : Masashi Kishimoto, Shonen Jump
Game : Neil Druckman, Sony Computer

C:\xampp\htdocs\PABI\php_oop\constructor.php:36:

```

object(Produk)[3]
  public 'judul' => string 'Dragon Ball' (length=11)
  public 'penulis' => string 'penulis' (length=7)
  public 'penerbit' => string 'penerbit' (length=8)
  public 'harga' => int 0

```

Video 5 (OOP DASAR pada PHP #5 - Object Type)

Tipe data dikatakan juga sebagai tipe dari isi variabel. Ada yang berbentuk angka yang disebut juga integer, dan yang berbentuk kalimat atau kata disebut dengan string dan tipe data lainnya. Berikut ini adalah beberapa tipe data pada PHP.

Tipe data pada PHP:

- String
- Integer
- Float
- Boolean
- Array
- Object
- NULL

String

Tipe data string adalah tipe data yang berbentuk text dan untuk cara penulisan tipe data string di letakkan di tengah-tengah tanda petik. di awali dengan tanda petik dan di akhiri dengan tanda petik juga

Integer

Tipe data integer adalah tipe data yang berbentuk angka yang berbentuk bilangan asli atau bilangan bulat.

Float

Tipe data float atau di sebut juga tipe data double adalah tipe data yang berisi bilangan desimal.

Array

Array adalah sebuah tipe data yang menyimpan banyak isi di dalam sebuah variabel.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

// Jualan Produk
// Komik
// Game
```

```

class Produk {
    public $judul,
           $penulis,
           $penerbit,
           $harga;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0) {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }
}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

$produk1 = new Produk("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000);
$produk2 = new Produk("Uncharted", "Neil Druckman", "Sony Computer", 250000);
$produk3 = new Produk("Dragon Ball");

echo "Komik : " . $produk1->getLabel();
echo "<br>";
echo "Game : " . $produk2->getLabel();
echo "<br>";

$infoProduk1 = new CetakInfoProduk();
echo $infoProduk1->cetak($produk1);

?>

```

Hasil:

← → ↻ ⓘ localhost:8080/PABI/php_oop/object-type.php

Komik : Masashi Kishimoto, Shonen Jump
 Game : Neil Druckman, Sony Computer
 Naruto | Masashi Kishimoto, Shonen Jump (Rp. 30000)

Video 6 (OOP DASAR pada PHP #6 - Inheritance (Bagian 1))

Inheritance adalah konsep OOP dimana sebuah class dapat menurunkan property dan method yang dimilikinya kepada class lain. Konsep inheritance dipakai untuk memanfaatkan fitur code reuse, yakni menghindari terjadinya 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.

Inheritance:

- Menciptakan hierarki antar kelas (parent and child)
- Child class, mewarisi semua property dan method dari parent-nya (yang visible)
- Child class, memperluas (extends) fungsionalitas dari parent-nya

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Produk {
    public $judul,
           $penulis,
           $penerbit,
           $harga,
           $jmHalaman,
           $waktuMain,
           $tipe;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
    "penerbit", $harga = 0,
        $jmHalaman = 0, $waktuMain = 0, $tipe ) {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
        $this->jmHalaman = $jmHalaman;
        $this->waktuMain = $waktuMain;
        $this->tipe = $tipe;
    }
}
```



```

public function getLabel() {
    return "{$this->penulis, $this->penerbit}";
}

public function getInfoLengkap() {
    // Komik : Naruto | Mashashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
    $str = "{$this->tipe} : {$this->judul} | {$this->getLabel()} (Rp. {$this-
>harga})";
    ";
    if($this->tipe == "Komik") {
        $str .= "- {$this->jmHalaman} Halaman.";
    }else if
    ($this->tipe == "Game") {
        $str .= " ~ {$this->waktuMain} Jam.";
    }

    return $str;
}
}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

$produk1 = new Produk("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100, 0,
    "Komik");
$produk2 = new Produk("Uncharted", "Neil Druckman", "Sony Computer", 250000, 0, 50,
    "Game");

echo $produk1->getInfoLengkap();
echo "<br>";
echo $produk2->getInfoLengkap();
?>

```

Hasil:

← → ↻ ⓘ localhost:8080/PABI/php_oop/inheritance-problem.php

Komik : Naruto | Masashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
 Game : Uncharted | Neil Druckman, Sony Computer (Rp. 250000) ~ 50 Jam.

Video 7 (OOP DASAR pada PHP #7 - Inheritance (Bagian 2))

Contoh class mobil

Daftar Property:

- Nama
- Merk
- Warna
- KecepatanMaksimal
- JumlahPenumpang

Daftar Mobil:

- TambahKecepatan
- KurangiKecepatan
- GantiTransmisi
- BelokKiri
- BelokKanan

Kelas mobil ini bisa kita instansiasi untuk berbagai macam mobil. Tapi ada beberapa mobil yang memiliki property dan method yang khusus, selain dari property dan method di atas. Misalkan:

Mobil Sport:

- Turbo

Method:

- JalankanTurbo

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Produk {
    public $judul,
           $penulis,
           $penerbit,
```

```

        $harga,
        $jmHalaman,
        $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0,
        $jmHalaman = 0, $waktuMain = 0) {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
        $this->jmHalaman = $jmHalaman;
        $this->waktuMain = $waktuMain;
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    public function getInfoLengkap() {
        // Komik : Naruto | Masashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }
}

class Komik extends Produk {
    public function getInfoProduk() {
        $str = "Komik : {$this->judul} | {$this->getLabel()} (Rp. {$this->harga}) -
        {$this->jmHalaman} Halaman.";
        return $str;
    }
}

class Game extends Produk {
    public function getInfoProduk() {
        $str = "Game : {$this->judul} | {$this->getLabel()} (Rp. {$this->harga}) ~
        {$this->waktuMain} Jam.";
        return $str;
    }
}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100, 0);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 0, 50);

```

```
echo $produk1->getInfoProduk();  
echo "<br>";  
echo $produk2->getInfoProduk();  
?>
```

Hasil:

← → ↻ ⓘ localhost:8080/PABI/php_oop/inheritance.php

Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.
Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.

Video 8 (OOP DASAR pada PHP #8 - Overriding)

Overriding adalah sebuah teknik PHP dengan konsep OOP yang mengimplementasikan pembuatan method yang sama pada class child dengan method parentnya. Terkadang ada kebutuhan untuk membuat fungsi yang sama namun keadaan yang berbeda. Maka dengan overriding menjadi solusi pemecahan masalah tersebut. Syarat untuk menggunakan teknik overriding yaitu harus mempunyai class parent dan class child.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Produk {
    public $judul,
           $penulis,
           $penerbit,
           $harga,
           $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
    "penerbit", $harga = 0, $waktuMain = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
        $this->waktuMain = $waktuMain;
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    public function getInfoProduk() {
        // Komik : Naruto | Mashashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
        $str = "{$this->judul} | {$this->getLabel()} (Rp. $this->harga)";

        return $str;
    }
}

class Komik extends Produk {
    public $jmHalaman;
```

```

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfoProduk() {

        $str = "Komik : " . parent::getInfoProduk() . " - {$this->jmHalaman}
Halaman.";
        return $str;
    }
}

class Game extends Produk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfoProduk() {
        $str = "Game : " . parent::getInfoProduk() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

echo $produk1->getInfoProduk();
echo "<br>";
echo $produk2->getInfoProduk();
?>

```

Hasil:

← → ↻ ⓘ localhost:8080/PABl/php_oop/overriding.php

Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.

Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.

Video 9 (OOP DASAR pada PHP #9 - Visibility)

Visibility

- Konsep yang digunakan untuk mengatur akses dari property dan method pada sebuah objek.
- Ada 3 keyword visibility: public, protected dan private.

Visibility

- Public, dapat digunakan di mana saja, bahkan di luar kelas.
- Protected, hanya dapat digunakan di dalam sebuah kelas beserta turunannya.
- Private, hanya dapat digunakan di dalam sebuah kelas tertentu saja.

Kenapa:

- Hanya memperlihatkan aspek dari class yang dibutuhkan oleh “client”.
- Menentukan kebutuhan yang jelas untuk object.
- Memberikan kendali pada kode untuk menghindari “bug”.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Produk {
    public $judul,
           $penulis,
           $penerbit;

    protected $diskon = 0;

    private $harga;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function getHarga() {
        return $this->harga - ($this->harga * $this->diskon / 100);
    }
}
```



```

    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    public function getInfoProduk() {
        // Komik : Naruto | Mashashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }
}

class Komik extends Produk {
    public $jmHalaman;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfoProduk() {

        $str = "Komik : " . parent::getInfoProduk() . " - {$this->jmHalaman}
Halaman.";
        return $str;
    }
}

class Game extends Produk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function setDiskon( $diskon ){
        $this->diskon = $diskon;
    }

    public function getInfoProduk() {
        $str = "Game : " . parent::getInfoProduk() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}

```

```

}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

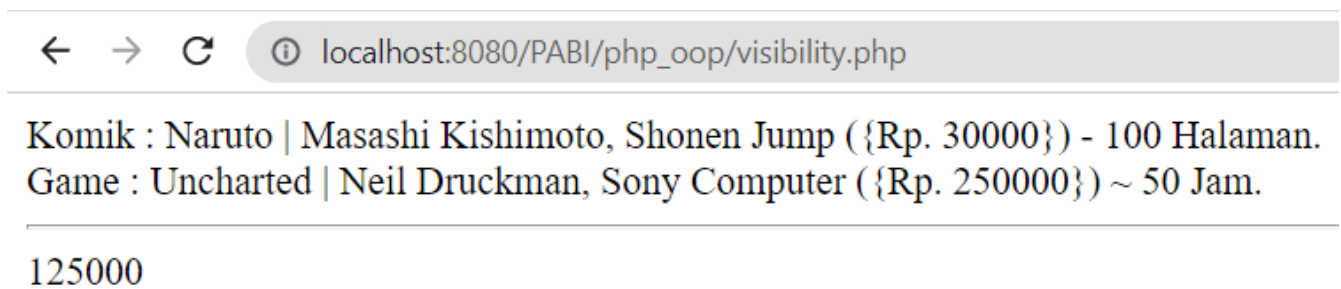
$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

echo $produk1->getInfoProduk();
echo "<br>";
echo $produk2->getInfoProduk();
echo "<hr>";

$produk2->setDiskon(50);
echo $produk2->getharga();
?>

```

Hasil:



Video 10 (OOP DASAR pada PHP #10 - Setter dan Getter)

Visibility

- Public, dapat digunakan di mana saja, bahkan di luar kelas.
- Protected, hanya dapat digunakan di dalam sebuah kelas beserta turunannya.
- Private, hanya dapat digunakan di dalam sebuah kelas tertentu saja.

Kita menggunakan setter dan getter ini untuk menghindari ketika kita membuat sebuah property dengan visibility public, karena sebaiknya kita tidak membiarkan bagian dari luar class kita bisa mengakses property secara langsung. Untuk itu kita harus mengubah visibility nya menjadi protected atau private, tergantung dari design kita.

Setelah kita mengubah visibility nya menjadi protected atau private itu artinya kita tidak lagi punya akses secara langsung jika kita mau mendapatkan isi dari sebuah property atau bahkan men-set nilai baru ke sebuah property, untuk itulah kita membutuhkan sebuah method. Method untuk bisa membaca atau melihat isi dari property ataupun mengubah isinya.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Produk {
    private $judul,
            $penulis,
            $penerbit,
            $harga,
            $diskon = 0;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function setJudul($judul) {
        $this->judul = $judul;
    }
}
```

```

    public function getJudul() {
        return $this->judul;
    }

    public function setPenulis($penulis) {
        $this->penulis = $penulis;
    }

    public function getPenulis() {
        return $this->penulis;
    }

    public function setPenerbit($penerbit) {
        $this->penerbit = $penerbit;
    }

    public function getPenerbit() {
        return $this->penerbit;
    }

    public function setDiskon( $diskon ){
        $this->diskon = $diskon;
    }

    public function getDiskon() {
        $this->diskon = $this->diskon;
    }

    public function setHarga() {
        $this->harga = $harga;
    }

    public function getHarga() {
        return $this->harga - ($this->harga * $this->diskon / 100);
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    public function getInfoProduk() {
        // Komik : Naruto | Mashashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
        $str = "{$this->judul} | {$this->getLabel()} (Rp. $this->harga)";

        return $str;
    }
}

class Komik extends Produk {
    public $jmHalaman;

```

```

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfoProduk() {

        $str = "Komik : " . parent::getInfoProduk() . " - {$this->jmHalaman}
Halaman.";
        return $str;
    }
}

class Game extends Produk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfoProduk() {
        $str = "Game : " . parent::getInfoProduk() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}

class CetakInfoProduk {
    public function cetak( Produk $produk ) {
        $str = "{$produk->judul} | {$produk->getLabel()} (Rp. {$produk->harga})";
        return $str;
    }
}

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

echo $produk1->getInfoProduk();
echo "<br>";
echo $produk2->getInfoProduk();
echo "<hr>";

$produk2->setDiskon(50);
echo $produk2->getharga();
echo "<hr>";

```

```
$produk1->setPenulis("Erik Tampubolon");  
echo $produk1->getPenulis();  
?>
```

Hasil:

← → ↻ ⓘ localhost:8080/PABl/php_oop/setter-dan-getter.php

Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.
Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.

125000

Erik Tampubolon

Video 11 (OOP DASAR pada PHP #11 - Static Keyword)

Dengan static keyword, kita bisa membuat static property dan static method, supaya kita bisa akses pada konteks class. 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.

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.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

// class ContohStatic {
//     public static $angka = 1;

//     public static function halo() {
//         return "Halo." . self::$angka++. "kali.";
//     }
// }

// echo ContohStatic::$angka;
// echo "<br>";
// echo ContohStatic::halo();
// echo "<hr>";
// echo contohStatic::halo();

class Contoh {
    public static $angka = 1;
    public function halo() {
        return "Halo." . self::$angka++ . "kali. <br>";
    }
}

$obj = new Contoh;
```

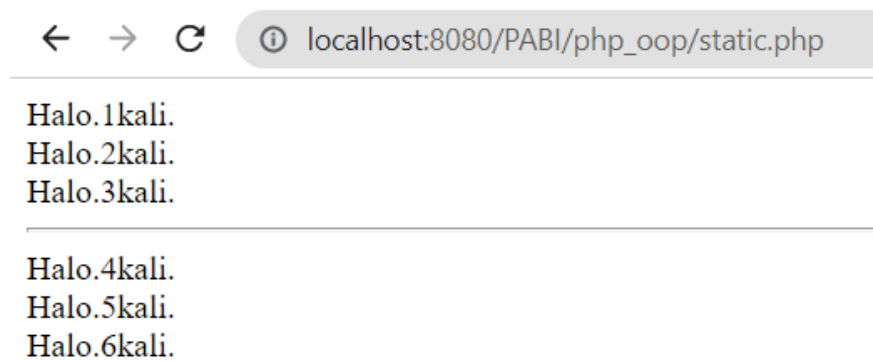
```
echo $obj->halo();
echo $obj->halo();
echo $obj->halo();

echo "<hr>";

$obj2 = new Contoh;
echo $obj2->halo();
echo $obj2->halo();
echo $obj2->halo();

?>
```

Hasil:



← → ↻ ⓘ localhost:8080/PABI/php_oop/static.php

Halo.1kali.
Halo.2kali.
Halo.3kali.

Halo.4kali.
Halo.5kali.
Halo.6kali.

https://drive.google.com/drive/folders/10N5WgluuS0MZf4GEgIe_lUB0PkdWM6_R

Video 12 (OOP DASAR pada PHP #12 - Constant)

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. 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 berperilaku sebagai static property. Class constant juga terikat kepada class, bukan objek.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

define('NAMA', 'Erik Tampubolon');
echo NAMA;

echo "<br>";

const UMUR= 19;
echo UMUR;
echo "<br>";

class Coba {
    const NAMA = 'Erik Tampubolon';
}

echo Coba::NAMA;

function coba() {
    return __FUNCTION__;
}

echo "<br>";
echo coba();

// class Coba {
//     public $kelas = __CLASS__;
// }

$obj = new Coba();
// echo $obj->kelas;
?>
```

Hasil:

← → ↻ ⓘ localhost:8080/PABl/php_oop/constant.php

Erik Tampubolon

19

Erik Tampubolon

coba

Video 13 (OOP DASAR pada PHP #13 - Abstract Class (bagian 1))

Abstract Class (1)

- Sebuah kelas yang tidak dapat di instansiasi
- Kelas “abstrak”
- Mendefinisikan interface untuk kelas lain yang menjadi turunannya
- Berperan sebagai “kerangka dasar” untuk kelas turunannya
- Memiliki minimal 1 method abstrak
- Digunakan “pewarisan” / inheritance untuk “memaksakan” implementasi method abstrak yang sama untuk semua kelas turunannya.

Abstract Class (1)

- Semua kelas turunan, harus mengimplementasikan method abstrak yang ada di kelas abstraknya
- Kelas abstrak boleh memiliki property/method reguler
- Kelas abstrak boleh memiliki property/static method

Contoh Kelas Abstrak

- Class mobil Extends Kendaraan
- Class laptop Extends Komputer
- Class email Extends Komunikasi

Kenapa menggunakan kelas abstrak

- Merepresentasikan ide atau konsep dasar
- “Composition over Inheritance”
- Salah satu cara menerapkan Polimorphism
- Sentralisasi logic
- Mempermudah pengerjaan tim

Video 14 (OOP DASAR pada PHP #14 - Abstract Class (bagian 2))

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi      : D3 Teknologi Komputer

abstract class Produk {
    private $judul,
            $penulis,
            $penerbit,
            $harga,
            $diskon = 0;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function setJudul($judul) {
        $this->judul = $judul;
    }

    public function getJudul() {
        return $this->judul;
    }

    public function setPenulis($penulis) {
        $this->penulis = $penulis;
    }

    public function getPenulis() {
        return $this->penulis;
    }

    public function setPenerbit($penerbit) {
        $this->penerbit = $penerbit;
    }

    public function getPenerbit() {
        return $this->penerbit;
    }

    public function setDiskon( $diskon ){
        $this->diskon = $diskon;
    }
}
```

```

    }

    public function getDiskon() {
        $this->diskon = $this->diskon;
    }

    public function setHarga() {
        $this->harga = $harga;
    }

    public function getHarga() {
        return $this->harga - ($this->harga * $this->diskon / 100);
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    abstract public function getInfoProduk();

    public function getInfo() {
        // Komik : Naruto | Mashashi Kishimoto, Shonen Jump (Rp. 30000) - 100 Halaman.
        $str = "{$this->judul} | {$this->getLabel()} (Rp. $this->harga)";

        return $str;
    }
}

class Komik extends Produk {
    public $jmHalaman;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfoProduk() {

        $str = "Komik : " . $this->getInfo() . " - {$this->jmHalaman} Halaman.";
        return $str;
    }
}

class Game extends Produk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {

```

```

        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfoProduk() {
        $str = "Game : " . $this->getInfo() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}

class CetakInfoProduk {
    public $daftarProduk = array();

    public function tambahProduk(Produk $produk) {
        $this->daftarProduk[] = $produk;
    }

    public function cetak() {
        $str = "DAFTAR PRODUK : <br>";

        foreach ($this->daftarProduk as $p) {
            $str .= "- {$p->getInfoProduk()} <br>";
        }

        return $str;
    }
}

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

$cetakProduk = new CetakInfoProduk();
$cetakProduk->tambahProduk( $produk1);
$cetakProduk->tambahProduk( $produk2);

echo $cetakProduk->cetak();

?>

```

Hasil:

←
→
↻
🔍 localhost:8080/PABl/php_oop/abstract-class.php

DAFTAR PRODUK :

- Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.
- Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.

Video 15 (OOP DASAR pada PHP #15 - Interface)

Interface (1)

- Kelas Abstrak yang sama sekali tidak memiliki implementasi.
- Murni merupakan template untuk kelas turunannya.
- Tidak boleh memiliki property, hanya deklarasi method saja.
- Semua method harus dideklarasikan dengan visibility public.
- Boleh mendeklarasikan __construct().

Interface (2)

- Dengan menggunakan type-hinting dapat melakukan “Dependency Injection”.
- Pada akhirnya akan mencapai Polymorphism.

Program:

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

interface infoProduk {
    public function getInfoProduk();
}

abstract class Produk {
    protected $judul,
               $penulis,
               $penerbit,
               $harga,
               $diskon = 0;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
    "penerbit", $harga = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function setJudul($judul) {
        $this->judul = $judul;
    }
}
```

```

    }

    public function getJudul() {
        return $this->judul;
    }

    public function setPenulis($penulis) {
        $this->penulis = $penulis;
    }

    public function getPenulis() {
        return $this->penulis;
    }

    public function setPenerbit($penerbit) {
        $this->penerbit = $penerbit;
    }

    public function getPenerbit() {
        return $this->penerbit;
    }

    public function setDiskon( $diskon ){
        $this->diskon = $diskon;
    }

    public function getDiskon() {
        $this->diskon = $this->diskon;
    }

    public function setHarga() {
        $this->harga = $harga;
    }

    public function getHarga() {
        return $this->harga - ($this->harga * $this->diskon / 100);
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    abstract public function getInfo();
}

class Komik extends Produk implements InfoProduk {
    public $jmHalaman;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {

```



```

        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {

        $str = "Komik : " . $this->getInfo() . " - {$this->jmHalaman} Halaman.";
        return $str;
    }
}

class Game extends Produk implements InfoProduk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {
        $str = "Game : " . $this->getInfo() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}

class CetakInfoProduk {
    public $daftarProduk = array();

    public function tambahProduk(Produk $produk) {
        $this->daftarProduk[] = $produk;
    }

    public function cetak() {
        $str = "DAFTAR PRODUK : <br>";

        foreach ($this->daftarProduk as $p) {

```

```

        $str .= "- {$p->getInfoProduk()} <br>";
    }

    return $str;
}
}

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

$cetakProduk = new CetakInfoProduk();
$cetakProduk->tambahProduk( $produk1);
$cetakProduk->tambahProduk( $produk2);

echo $cetakProduk->cetak();

?>

```

Hasil:

localhost:8080/PABl/php_oop/interface.php

DAFTAR PRODUK :

- Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.
- Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.

Video 16 (OOP DASAR pada PHP #16 - Autoloading)

Autoloading merupakan salah satu konsep OOP di dalam pemrograman PHP yang digunakan untuk memanggil file lain secara otomatis. Autoloading ini sebagai pengganti ketika menggunakan perintah (keyword) `require`, `require_once` maupun menggunakan `include`.

Fungsi Autoloading

Autoloading berfungsi untuk memanggil file yang di pisahkan, latihan sebelumnya kita membuat class dalam satu file. Nah untuk menangani banyak module maka sebaiknya setiap class terhadap object harus di pisahkan.

Struktur File:

Name	Date modified
App	21/04/2022 12:15
index	21/04/2022 12:45

Didalam File App:

Name	Date modified
Produk	21/04/2022 12:44
init	21/04/2022 12:18

Didalam File Produk:

Name	Date modified	Type
CetakInfoProduk	21/04/2022 11:20	PHP File
Coba	21/04/2022 12:44	PHP File
Game	21/04/2022 11:20	PHP File
InfoProduk	21/04/2022 11:17	PHP File
Komik	21/04/2022 11:19	PHP File
Produk	21/04/2022 11:18	PHP File

Program:

Index.php

```
<?php  
  
require_once 'App/init.php';
```

```

$produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
$produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

$cetakProduk = new CetakInfoProduk();
$cetakProduk->tambahProduk( $produk1);
$cetakProduk->tambahProduk( $produk2);

echo $cetakProduk->cetak();

echo "<hr>";

new Coba();

```

init.php

```

<?php

require_once 'Produk/InfoProduk.php';
require_once 'Produk/Produk.php';
require_once 'Produk/Komik.php';
require_once 'Produk/Game.php';
require_once 'Produk/CetakInfoProduk.php';

spl_autoload_register(function ( $class ){
    require_once 'Produk/' . $class . '.php';
});

```

CetakInfoProduk.php

```

<?php

class CetakInfoProduk {
    public $daftarProduk = array();

    public function tambahProduk(Produk $produk) {
        $this->daftarProduk[] = $produk;
    }

    public function cetak() {
        $str = "DAFTAR PRODUK : <br>";

        foreach ($this->daftarProduk as $p) {
            $str .= "- {$p->getInfoProduk()} <br>";
        }

        return $str;
    }
}

```

Coba.php

```
<?php

class Coba {
    public function __construct() {
        echo "Ini adalah kelas Coba";
    }
}
```

Game.php

```
<?php

class Game extends Produk implements InfoProduk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {
        $str = "Game : " . $this->getInfo() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}
```

InfoProduk.php

```
<?php

interface infoProduk {
    public function getInfoProduk();
}
```

Komik.php

```
<?php

class Komik extends Produk implements InfoProduk {
    public $jmHalaman;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {

        $str = "Komik : " . $this->getInfo() . " - {$this->jmHalaman} Halaman.";
        return $str;
    }
}
```

Produk.php

```
<?php

abstract class Produk {
    protected $judul,
                $penulis,
                $penerbit,
                $harga,
                $diskon = 0;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0)
    {
        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function setJudul($judul) {
        $this->judul = $judul;
    }
}
```

```
public function getJudul() {  
    return $this->judul;  
}  
  
public function setPenulis($penulis) {  
    $this->penulis = $penulis;  
}  
  
public function getPenulis() {  
    return $this->penulis;  
}  
  
public function setPenerbit($penerbit) {  
    $this->penerbit = $penerbit;  
}  
  
public function getPenerbit() {  
    return $this->penerbit;  
}  
  
public function setDiskon( $diskon ){  
    $this->diskon = $diskon;  
}  
  
public function getDiskon() {  
    $this->diskon = $this->diskon;  
}  
  
public function setHarga() {  
    $this->harga = $harga;  
}  
  
public function getHarga() {  
    return $this->harga - ($this->harga * $this->diskon / 100);  
}  
  
public function getLabel() {  
    return "$this->penulis, $this->penerbit";  
}  
  
abstract public function getInfo();  
}
```

Hasil:

← → ↻ ⓘ localhost:8080/PABl/php_oop/autoloading/

DAFTAR PRODUK :

- Komik : Naruto | Masashi Kishimoto, Shonen Jump ({Rp. 30000}) - 100 Halaman.
 - Game : Uncharted | Neil Druckman, Sony Computer ({Rp. 250000}) ~ 50 Jam.
-

Ini adalah kelas Coba

Video 17 (OOP DASAR pada PHP #17 - Namespace)

Namespace Pada OOP merupakan sebuah fitur yang terdapat pada PHP yang berfungsi untuk menghindari terjadinya sebuah kesalahan atau error ketika kita membuat sebuah nama class yang sama, dan Namespace juga berfungsi untuk merubah nama class yang sama, dan kita dapat merubahnya dengan nama yang berbeda

Membuat dan menggunakan Namespace

Namespace terletak pada bagian paling atas dalam script PHP setelah tanda pembuat PHP. Format penulisan namespace seperti di bawah ini:

```
Namespace Nama\SubNama
```

Struktur File:

Name	Date modified
App	21/04/2022 12:47
index	22/04/2022 8:51

Didalam File App:

Name	Date modified
Produk	21/04/2022 12:54
init	21/04/2022 13:20

Didalam File Produk:

Name	Date modified
Service	21/04/2022 12:58
CetakInfoProduk	21/04/2022 11:20
Game	21/04/2022 11:20
InfoProduk	21/04/2022 11:17
Komik	21/04/2022 11:19
Produk	21/04/2022 11:18
User	21/04/2022 13:02

Didalam File Service:

Name	Date modified
User	21/04/2022 13:02

Program:

Index.php

```
<?php

require_once 'App/init.php';

// $produk1 = new Komik("Naruto", "Masashi Kishimoto", "Shonen Jump", 30000, 100);
// $produk2 = new Game("Uncharted", "Neil Druckman", "Sony Computer", 250000, 50);

// $cetakProduk = new CetakInfoProduk();
// $cetakProduk->tambahProduk( $produk1);
// $cetakProduk->tambahProduk( $produk2);

// echo $cetakProduk->cetak();

// echo "<hr>";

use App\Service\User as ServiceUser;
use App\Produk\User as ProdukUser;

new ServiceUser();
echo "<br>";
new ProdukUser();
```

init.php

```
<?php

require_once 'Produk/InfoProduk.php';
require_once 'Produk/Produk.php';
require_once 'Produk/Komik.php';
require_once 'Produk/Game.php';
require_once 'Produk/CetakInfoProduk.php';
require_once 'Produk/User.php';
require_once 'Produk/Service/User.php';

spl_autoload_register(function ( $class ){
    // App\Produk\User = ["App", "Produk", "User"]
    $class = explode( '\\', $class );
    $class = end($class);
    require_once 'Produk/' . $class . '.php';
});

spl_autoload_register(function ( $class ){
    $class = explode( '\\', $class );
    $class = end($class);
    require_once 'Service/' . $class . '.php';
});
```

```
});
```

CetakInfoProduk.php

```
<?php

class CetakInfoProduk {
    public $daftarProduk = array();

    public function tambahProduk(Produk $produk) {
        $this->daftarProduk[] = $produk;
    }

    public function cetak() {
        $str = "DAFTAR PRODUK : <br>";

        foreach ($this->daftarProduk as $p) {
            $str .= "- {$p->getInfoProduk()} <br>";
        }

        return $str;
    }
}
```

Game.php

```
<?php

class Game extends Produk implements InfoProduk {
    public $waktuMain;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit =
"penerbit", $harga = 0, $waktuMain = 0)
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->waktuMain = $waktuMain;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {
        $str = "Game : " . $this->getInfo() . " ~ {$this->waktuMain} Jam.";
        return $str;
    }
}
```

InfoProduk.php

```
<?php

interface infoProduk {
    public function getInfoProduk();
}
```

Komik.php

```
<?php

class Komik extends Produk implements InfoProduk {
    public $jmHalaman;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0, $jmHalaman = 0 )
    {
        parent::__construct( $judul, $penulis, $penerbit, $harga);

        $this->jmHalaman = $jmHalaman;
    }

    public function getInfo() {
        $str = "{$this->judul} | {$this->getLabel()} (Rp. {$this->harga})";

        return $str;
    }

    public function getInfoProduk() {

        $str = "Komik : " . $this->getInfo() . " - {$this->jmHalaman} Halaman.";
        return $str;
    }
}
```

Produk.php

```
<?php

abstract class Produk {
    protected $judul,
                $penulis,
                $penerbit,
                $harga,
                $diskon = 0;

    public function __construct( $judul = "judul", $penulis = "penulis", $penerbit = "penerbit", $harga = 0)
    {
```

```

        $this->judul = $judul;
        $this->penulis = $penulis;
        $this->penerbit = $penerbit;
        $this->harga = $harga;
    }

    public function setJudul($judul) {
        $this->judul = $judul;
    }

    public function getJudul() {
        return $this->judul;
    }

    public function setPenulis($penulis) {
        $this->penulis = $penulis;
    }

    public function getPenulis() {
        return $this->penulis;
    }

    public function setPenerbit($penerbit) {
        $this->penerbit = $penerbit;
    }

    public function getPenerbit() {
        return $this->penerbit;
    }

    public function setDiskon( $diskon ){
        $this->diskon = $diskon;
    }

    public function getDiskon() {
        $this->diskon = $this->diskon;
    }

    public function setHarga() {
        $this->harga = $harga;
    }

    public function getHarga() {
        return $this->harga - ($this->harga * $this->diskon / 100);
    }

    public function getLabel() {
        return "$this->penulis, $this->penerbit";
    }

    abstract public function getInfo();

```

```
}
```

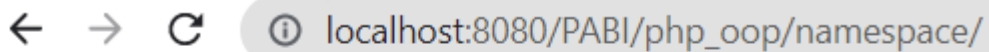
User.php

```
<?php namespace App\Produk;  
  
class User {  
    public function __construct() {  
        echo "Ini adalah class " . __CLASS__;  
    }  
}
```

Service/User.php

```
<?php namespace App\Service;  
  
class User {  
    public function __construct() {  
        echo "Ini adalah class " . __CLASS__;  
    }  
}
```

Hasil:



Ini adalah class App\Service\User
Ini adalah class App\Produk\User

Membuat Aplikasi MVC dengan PHP

MVC adalah singkatan dari “Model View Controller” merupakan suatu konsep yang sangat populer dalam pembangunan website dan aplikasi. MVC memisahkan pengembangan aplikasi berdasarkan 3 jenis komponen utama yaitu manipulasi data, user interface, dan bagian yang menjadi kontrol aplikasi. Komponen-komponen utama tersebut membangun suatu MVC pattern atau bagian yang diberi nama Model, View dan Controller.

Dengan menggunakan konsep MVC, suatu aplikasi dapat dikembangkan sesuai dengan kemampuan PIC-nya, Developer yang menangani bagian Model dan Controller, sedangkan Web Designer yang menangani bagian View, sehingga penggunaan arsitektur MVC dapat meningkatkan maintainability dan pengorganisasian kode. Meskipun begitu, namun tetap dibutuhkan komunikasi yang baik Developer dan Web Designer dalam menangani variabel dan parameter data yang ada

Model

Model adalah bagian yang berhubungan langsung dengan database, model bertugas untuk memanipulasi data (select, insert, update, delete) serta menangani validasi dari bagian Controller, namun tidak dapat berhubungan langsung dengan bagian View.

View

View adalah bagian yang menangani Presentation Logic. Pada suatu aplikasi web bagian ini merupakan template yang berupa file HTML, View ini diatur oleh bagian Controller. Bagian View berfungsi untuk menerima dan mempresentasikan data kepada user, atau View ini bisa dibilang sebagai interface aplikasi. Bagian ini tidak memiliki akses langsung terhadap database atau bagian Model.

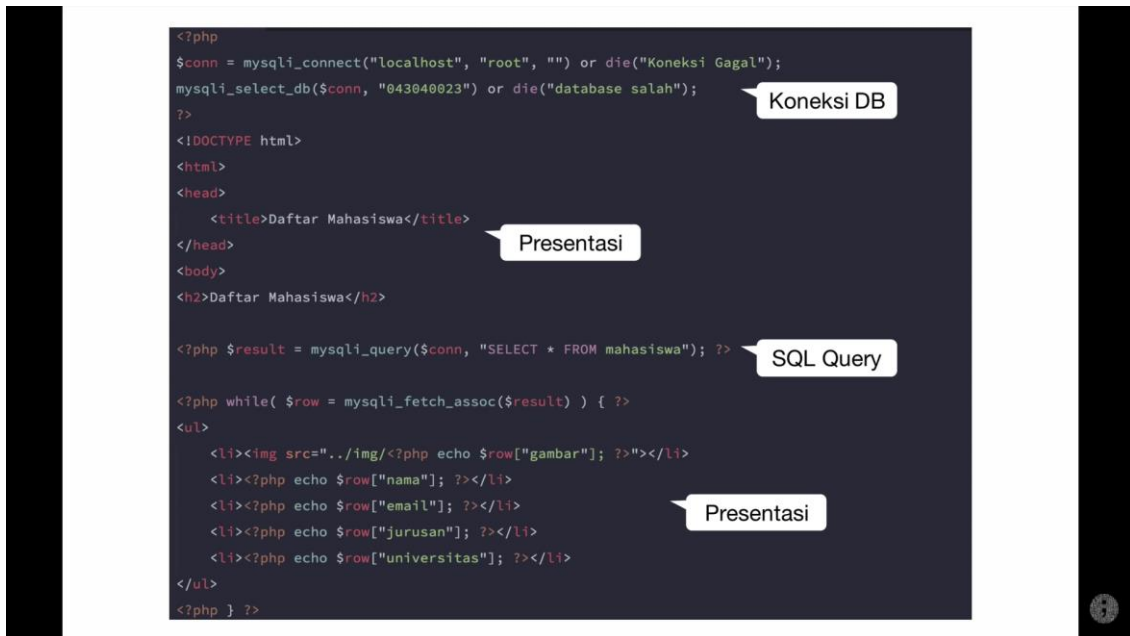
Controller

Controller adalah bagian yang mengatur hubungan antara bagian Model dan bagian View. Controller berfungsi untuk menerima sebuah request data dari user, kemudian menentukan apa yang akan diproses oleh aplikasi tersebut. Selain itu, bagian Controller juga mengatur routing URL yang akan digunakan.

Membuat Aplikasi MVC dengan PHP

Video 1 (Membuat Aplikasi MVC dengan PHP #1 Pendahuluan)

MVC adalah sebuah pola arsitektur pada perancangan perangkat lunak berorientasi objek. Salah satu tujuan MVC adalah memisahkan antara tampilan, data dan proses.

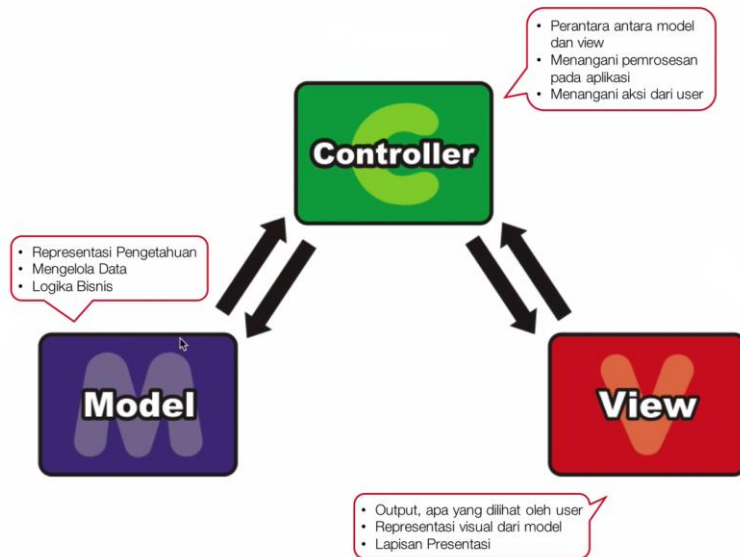


Kenapa MVC

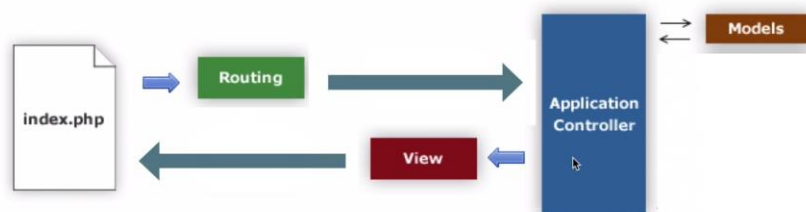
- Organisasi dan Struktur Kode
- Pemisahan logic dan tampilan
- Perawatan Kode
- Implementasi konsep OOP yang sudah dipelajari
- Digunakan oleh banyak Web Application Framework

Framework MVC

Bahasa	Framework
PHP	CodeIgniter, Laravel, Yii
Java	Spring MVC, JSF, Struts
Python	Django, CherryPy
Ruby	Ruby in Rails, Sinatra
Javascript	AngularJS, React, Backbone.js



APPLICATION FLOW YANG AKAN KITA BUAT



Video 2 (Membuat Aplikasi MVC dengan PHP #2 Persiapan)

Struktur File:

Name	Date modified
app	21/04/2022 21:17
public	21/04/2022 21:14

Didalam File app:

Name	Date modified
controllers	21/04/2022 21:11
core	21/04/2022 21:18
models	21/04/2022 21:12
views	21/04/2022 21:12
init	21/04/2022 21:18

Didalam File core:

Name	Date modified
App	21/04/2022 22:36
Controller	21/04/2022 21:19

Didalam File views:

Name	Date modified
home	21/04/2022 21:12

Didalam File public:

Name	Date modified
css	21/04/2022 21:10
img	21/04/2022 21:10
js	21/04/2022 21:10
index	21/04/2022 21:19

Program:

index.php

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
```

```
// Prodi    :    D3 Teknologi Komputer

require_once '../app/init.php';

$app = new App;
```

init.php

```
<?php
// Nama      :    Erik Dito Tampubolon
// NIM       :    13321030
// Prodi     :    D3 Teknologi Komputer

require_once 'core/App.php';
require_once 'core/Controller.php';
```

App.php

```
<?php
// Nama      :    Erik Dito Tampubolon
// NIM       :    13321030
// Prodi     :    D3 Teknologi Komputer

class App {
    public function __construct() {
        echo 'OK!';
    }
}
```

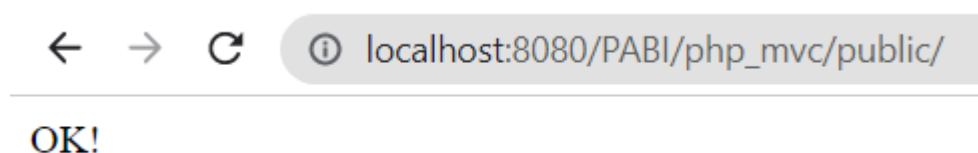
Controller.php

```
<?php
// Nama      :    Erik Dito Tampubolon
// NIM       :    13321030
// Prodi     :    D3 Teknologi Komputer

class Controller {

}
```

Hasil:



Video 3 (Membuat Aplikasi MVC dengan PHP #3 Routing)

Program:

App.php

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class App {
    public function __construct() {
        $url = $this->parseURL();
        var_dump($url);
    }

    public function parseURL(){
        if ( isset($_GET['url']) ) {
            $url = rtrim($_GET['url'], '/');
            $url = filter_var($url, FILTER_SANITIZE_URL);
            $url = explode('/', $url);
            return $url;
        }
    }
}
```

app/.htaccess

```
Options -Indexes
```

public/.htaccess

```
Options -Multiviews

RewriteEngine On
RewriteCond %{REQUEST_FILENAME} !-d
RewriteCond %{REQUEST_FILENAME} !-f
RewriteRule ^(.*)$ index.php?url=$1 [L]
```

Hasil:

← → ↻ localhost:8080/PABI/php_mvc/public/about/page/satu/dua

C:\xampp\htdocs\PABI\php_mvc\app\core\App.php:9:

array (size=4)

0 => string 'about' (length=5)

1 => string 'page' (length=4)

2 => string 'satu' (length=4)

3 => string 'dua' (length=3)

Video 4 (Membuat Aplikasi MVC dengan PHP #4 Controller)

Program:

App.php

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class App {
    protected $controller = 'Home';
    protected $method = 'index';
    protected $params = [];

    public function __construct() {
        $url = $this->parseURL();

        // Controller
        if (file_exists('../app/controllers/' . $url[0] . '.php')) {
            $this->controller = $url[0];
            unset($url[0]);
        }

        require_once '../app/controllers/' . $this->controller . '.php';
        $this->controller = new $this->controller;

        // Method
        if (isset($url[1])) {
            if (method_exists($this->controller, $url[1])) {
                $this->method = $url[1];
                unset($url[1]);
            }
        }

        // Parameters
        if (!empty($url)) {
            $this->params = array_values($url);
        }

        // Jalankan Controller & method, serta kirimkan parameter
        call_user_func_array([$this->controller, $this->method], $this->params);
    }

    public function parseURL(){
        if (isset($_GET['url'])) {
            $url = rtrim($_GET['url'], '/');
        }
    }
}
```

```

        $url = filter_var($url, FILTER_SANITIZE_URL);
        $url = explode('/', $url);
        return $url;
    }
}
}

```

About.php

```

<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class About {

    public function index($nama = 'Erik Tampubolon', $pekerjaan = 'Mahasiswa' ) {
        echo "Hallo, nama saya $nama, saya adalah seorang $pekerjaan";
    }

    public function page() {
        echo 'About/page';
    }
}

```

Home.php

```

<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Home extends Controller {
    public function index() {
        $this->view('templates/header');
        $this->view('home/index');
        $this->view('templates/footer');
    }
}

```

Hasil (1):

← → ↻ ⓘ localhost:8080/PABI/php_mvc/public/about

Hallo, nama saya Erik Tampubolon, saya adalah seorang Mahasiswa

Hasil (2):

← → ↻ ⓘ localhost:8080/PABI/php_mvc/public/about/Erik/Gamer

Hallo, nama saya Erik, saya adalah seorang Gamer

Video 5 (Membuat Aplikasi MVC dengan PHP #5 View)

Program:

About.php

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class About extends Controller {

    public function index($nama = 'Erik Tampubolon', $pekerjaan = 'Mahasiswa', $umur =
19) {
        $data['nama'] = $nama;
        $data['pekerjaan'] = $pekerjaan;
        $data['umur'] = $umur;
        $data['judul'] = 'About Me';
        $this->view('templates/header');
        $this->view('about/index', $data);
        $this->view('templates/footer');
    }

    public function page() {
        $data['judul'] = 'Pages';
        $this->view('templates/header', $data);
        $this->view('about/page');
        $this->view('templates/footer');
    }
}
```

Home.php

```
<?php
// Nama      : Erik Dito Tampubolon
// NIM       : 13321030
// Prodi     : D3 Teknologi Komputer

class Home extends Controller {
    public function index() {
        $data['judul'] = 'Home';
        $this->view('templates/header');
        $this->view('home/index');
        $this->view('templates/footer');
    }
}
```

app/views/home/about/index.php

```
<h1>About Me</h1>
    <p>Hallo, Nama <?= $data['nama']?>, Umur Saya <?= $data['umur']?>, Saya Adalah
Seorang <?= $data['pekerjaan']?> </p>
```

app/views/templates/header.php

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <title>Halaman <?= $data['judul']; ?> ?Home</title>
</head>
<body>
```

app/views/templates/footer.php

```
</body>
</html>
```

app/views/home/page.php

```
<h1>My Pages</h1>
```

app/views/home/index.php

```
<h1>Selamat Datang di Website Erik Dito</h1>
```