**Pertemuan 3**

**Pemrograman Berorientasi Objek**

1. **Pengertian**

Pemrograman Berbasis Objek (PBO) adalah konsep dimana Property / Variable dan juga Method / Fungsi di bungkus dalam sebuah Class, yang kemudian akan di terapkan pada Objek – objek yang di deklarasikan

Contoh program class di PHP

<?php

class Manusia{

// property declaration

public $nama;

// method declaration

public function cetak() {

echo $this->nama;

}

}

$amir = new Manusia;

$hasan = new Manusia;

$fitri = new Manusia;

?>

Class Manusia ini adalah nama Class – nya, $nama adalah property yang dimiliki class Manusia, sedangkan cetak() adalah fungsi yang juga dimiliki class Manusia. Dan setelah Class, Property, juga Method telah di definisikan, langkah selanjutnya adalah menginisialisasi – kannya ke dalam bentuk Objek.

1. **Pengertian Class dan Object**

Class adalah kerangka dasar untuk mebuat Object, maksudnya adalah, pada Class ini lah property dan juga method di deklarasikan. Misal untuk membuat sebuah Mobil, maka anda perlu menggambarkan beberapa property yang dimiliki Mobil nantinya seperti roda, pintu mobil, dan juga spion mobil. Selain itu anda juga harus mendefinisikan aksi – aksi (method) apa saja yang dapat dilakukan oleh mobil ini, yakni berjalan, mengerem dan membunyikan klakson.

Setelah gambaran awal dari property dan method ini sudah di definisikan, barulah dibuat Object – nya.

Pembuatan Class dalam PHP dimulai dengan mendeklarasikan keyword class, di ikuti dengan penulisan nama class, dan terakhir di ikuti oleh tanda kurung kurawal.

Object adalah hasil nyata dari sebuah kerangka Class yang digambarkan. Kalau jika ada sebuah Class yang menggambarkan Manusia, maka seseorang bernama Budi / Arif / Fitri adalah Object – nya. Nah dalam gambaran Class Manusia ini terdapat Property seperti Mata, Telinga, dan Rambut dan juga terdapat Method seperti Berjalan, Melihat, Menggigit dan juga Mendengar.

Dalam PHP untuk membuat Object, anda harus mendefinisikannya dengan keyword New, dan proses ini perwujudan Object dari Class dikenal dengan istilah instance / instantiation.

Contoh lengkap:

<?php

class Manusia{

// property declaration

public $mata = 'hitam';

// method declaration

public function berjalan() {

echo 'berjalan';

}

}

$Amir = new Manusia;

$Hasan = new Manusia;

?>

<p>Hai, nama saya Hasan, saya bermata <?= $Hasan->mata ?>, saya dapat <?= $Hasan->berjalan() ?>.</p>

<p>Hai, nama saya Amir, saya bermata <?= $Amir->mata ?>, saya dapat <?= $Hasan->berjalan() ?>.</p>

1. **Method dan property**

Property adalah Asset yang dimiliki oleh Class, yakni sebuah variable yang dapat anda setting sendiri nilainya. Property ini berisi data, dan proses pembuatannya cukup mudah sama seperti pembuatan variable pada umumnya.

Namun bila pada Class pertama – pertama anda harus mendefinisikan status hak akses dari Property, apakah akan di jadikan sebagai Public / Protected / Private.

Setelah itu di ikuti dengan nama Variable, konsep penama’annya sama seperti proses pembuatan Variable pada umumnya.

<?php

class Manusia{

// property declaration

public $telinga = 'caplang';

public $usia = 22;

}

$Amir = new Manusia;

?>

<p>Hai, nama saya Amir, saya memiliki telinga <?= $Amir->telinga ?>, usia saya <?= $Amir->usia ?> tahun. </p>

Method adalah Fungsi / Aksi yang dimiliki oleh Class, Method ini baru dapat dijalankan ketika namanya dipanggil. Proses pembuatannya pun sama seperti konsep pembuatan fungsi biasa pada PHP, dimana anda harus mendefinisikan fungsinya menggunakan keyword function yang selanjutnya di ikuti dengan nama fungsi tersebut.

<?php

class Manusia{

// method declaration

public function berjalan() {

echo 'berjalan';

}

}

$Amir = new Manusia;

?>

<p>Saya Amir, Saya dapat <?= $Amir->berjalan() ?>.</p>

1. **Enkapsulasi (pembungkusan)**

Enkapsulasi / Pengkapsulan adalah konsep melindungi kode sumber yang ada pada Class dari pengaksesannya dari luar Class. Dengan proses Pengkapsulan ini, seorang Developer dapat menentukan apakah Property dan juga Method yang dibuat dapat digunakan di luar Class, atau hanya dapat digunakan oleh Class itu sendiri dan juga turunannya ?

Proses Pengkapsulan terbagi ke dalam 3 cara, yakni Public, Protected, dan juga Private. 3 Keyword ini di definisikan sebelum pendefinisian Property dan Method. Sekarang mari kita bahas fungsi dari ketiga keyword ini:

1. Hak akses public

Hak Akses Public, dengan keyword ini, berarti anda mendefinisikan supaya Property dan juga Method dapat di akses oleh Class turunan dan juga di luar Class. Contoh :

<?php

class Hamba{

// Pembuatan Property

public $nama = '';

// Pembuatan Method

public function jalan(){

return 'Menggunakan kaki, secara bergantian.';

}

}

// Intansiasi Object

$Manusia = new Hamba;

?>

<p>Hai, nama saya <?= $Manusia->nama = 'Amir'; ?></p>

<p>Saya berjalan dengan <?= $Manusia->jalan(); ?> </p>

Maka ketika Property dan juga Method tersebut di akses, tidak akan muncul error apapun

1. Hak akses protected

Hak Akses Protected, jika sebuah Property / Method di definisikan sebagai Protected, maka Property dan Method tersebut hanya dapat di akses dari dalam Class itu sendiri dan juga Class turunan.

Selain dapat di akses dari dalam Class, Property / Method yang di definisikan sebagai Protected juga dapat di akses oleh Class yang menjadi turunannya, atau biasa disebut Class Anak / Child Class

<?php

class OrangTua{

// Pembuatan Property Protected

protected $kadar\_gula = 'Tinggi';

// Pembuatan Method Protected

protected function ciriFisik(){

return 'Cara Jalan Cepat';

}

// Mengakses Property $kadar\_gula yang di definisikan sebagai Protected

public function kadarGulaSaya(){

return $this->kadar\_gula;

}

// Mengakses Method ciriFisik() yang di definisikan sebagai Protected

public function ciriFisikSaya(){

return $this->ciriFisik();

}

}

// Intansiasi Object Class Parent

$Amir = new OrangTua;

?>

<h3>Parent</h3>

<p>Kadar Gula saya <?= $Amir->kadarGulaSaya() ?></p>

<p>Saya memiliki <?= $Amir->ciriFisikSaya() ?></p>

1. Hak akses private

hak akses Private, dimana jika sebuah Property / Method di definisikan sebagai Private, maka Property / Method tersebut hanya dapat di akses dari dalam Class yang mendefinisikannya dan akan menghasilkan error apabila di akses dari luar Class / Class turunan – nya.

<?php

class Mobil {

private $nomor\_seri = 'A77FFH';

private function speed(){

return '500km/jam';

}

public function nomor\_seri(){

return $this->nomor\_seri;

}

}

$Nissan = new Mobil;

?>

<p>Nomor seri Mobil Nissan = <?= $Nissan->nomor\_seri() ?></p>

1. **Inheritance (Pewarisan)**

Inheritance (Pewarisan) Adalah konsep penurunan sifat (maksudnya Property dan Method jika dalam OOP) yang dimiliki oleh sebuah Class kepada Class lain yang menjadi Class turunannya. Class yang menurunkan Property/Method ini disebut Parent Class / Class Induk, sedangkan Class yang menerima pewarisan Property/Method ini disebut Child Class / Class Anak.

Agar Class Parent dapat menurunkan Property/Methodnya, maka Property/Method itu harus di definisikan sebagai Public / Protected, sedangkan syarat bagi sebuah Class menjadi Class Anak, maka dalam pendefinisian Class ini harus ditambah keyword extends.

<?php

// Pembuatan Class Parent

class ClassInduk {

// Pendefinisian Property

public $properti;

// Pendefinisian Method

public function metode(){

}

}

// Pembuatan Class Anak

class ClassAnak extends ClassInduk{

// Pendefinisian Method pada Class Anak

public function cetakProperty(){

return $this->properti();

}

}

?>

1. **Static Property dan Static Method**

Property dan juga Method yang dibuat di dalam Class tidak hanya dapat diakses melalui Objek saja. Namun anda dapat juga mengaksesnya dengan cara menulis nama Class yang mendeklarasikannya kemudian menambahkan sebuah operator khusus seperti ini ‘::’, barulah anda tulis nama Property / Method tersebut.

Cara seperti ini dikenal dengan nama Static Property / Static Method, dimana anda mencetak nilainya menggunakan nama Class. Sedangkan operator yang berada di tengah – tengah ‘::’ disebut operator Static.

Untuk membuat sebuah Property / Method menjadi Static, anda hanya perlu mendefiniskan keyword static sesudah keyword hak akses Property/Method (public / protected / private), dan sebelum nama Property / Method.

Dengan menggunakan konsep Static ini kita akan lebih mudah dalam mencetak nilai yang ada di dalam Property/Method, karena tidak perlu meng – intansiasi kannya ke dalam bentuk Object. Namun pilihan tetap di tangan anda, ingin menggunakan konsep Static ini, atau tetap ingin membuat sebuah Object untuk mencetak Property / Method yang ada di dalam Class.

<?php

// Pembuatan Class Manusia

class Manusia{

// Pendefinisian Static Property $nama

public static $nama = 'Wahyu';

// Pendefinisian Static Property Method berjalan()

public static function berjalan(){

return 'menggunakan kaki kanan dan kiri secara ber - iringan';

}

}

?>

<p>Hai, nama saya <?= Manusia::$nama ?>, saya berjalan dengan <?= Manusia::berjalan() ?>.</p>

Ada perbedaan dalam hal pengaksesan Property / Method dari dalam Class. Bila secara normal kita menggunakan variable $this untuk mengaksesnya, namun dalam konsep Static kita harus menggantinya dengan keyword ‘self::’, seperti ini :

<?php

// Pembuatan Class Manusia

class Manusia{

// Pendefinisian Static Property $nama

public static $nama = 'Wahyu';

// Pendefinisian Static Property Method berjalan()

public static function namaSaya(){

// Memanggil Property $nama yang di definisikan secara Static

return self::$nama;

}

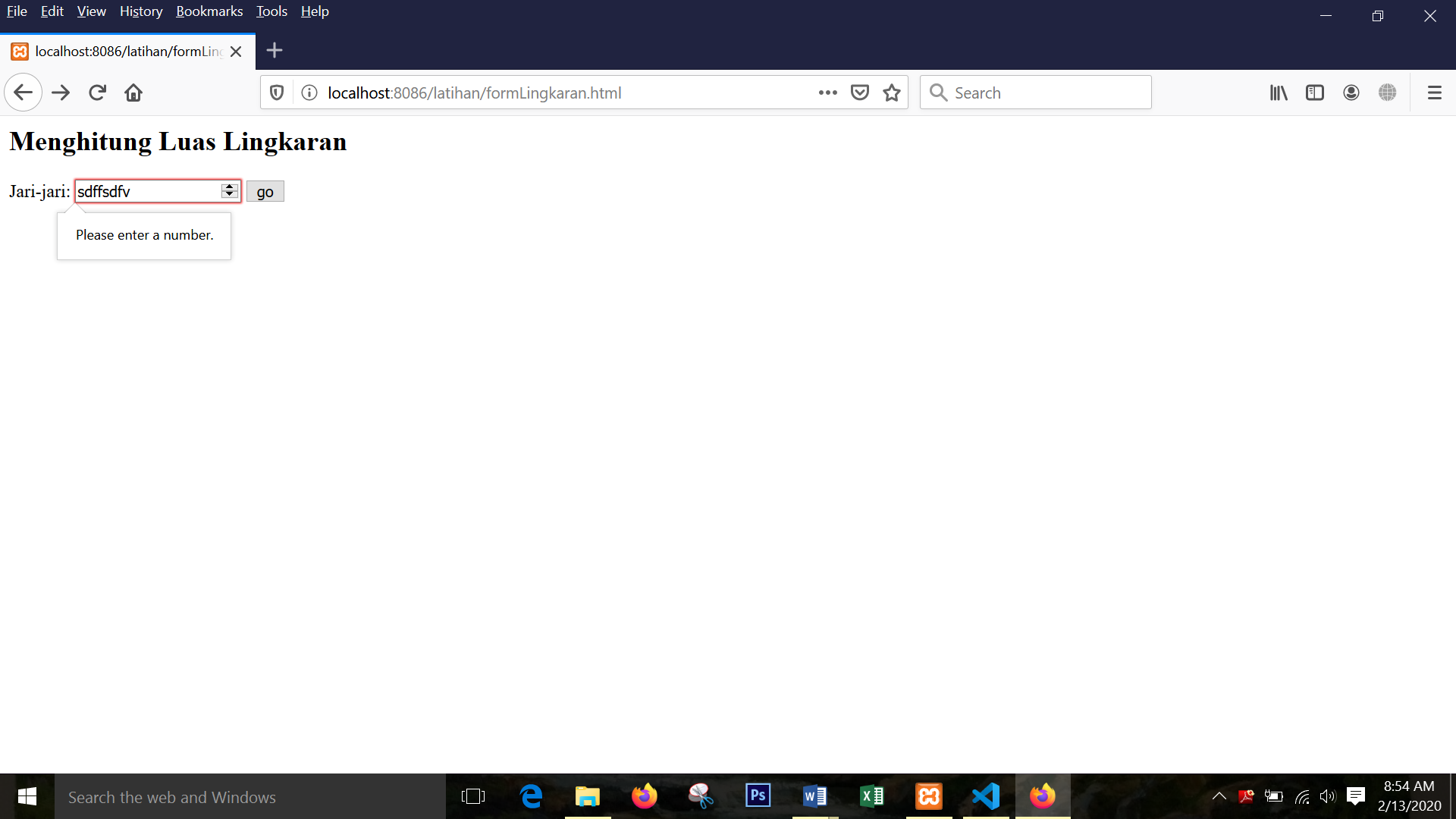
}

?>

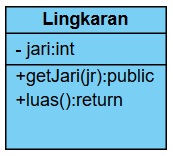
<p>Hai, nama saya <?= Manusia::namaSaya() ?>.</p>

Pada kode sumber di atas saya meletakkan Property $nama di dalam Method namaSaya(), dan tidak terjadi error, karena Property $nama sudah saya definisikan sebagai static. Namun jika Property $nama bukanlah bersifat static, maka akan memunculkan pesan error karena akan di anggap sebagai Property yang disiapkan untuk Object.

1. **Soal Latihan**
   1. Buatlah file **formLingkaran.html** seperti gambar di bawah ini, dengan validasinya



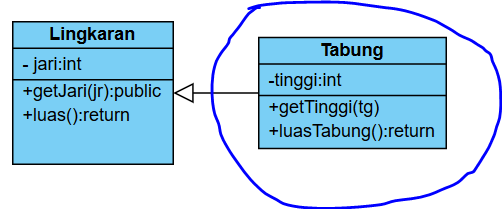
* 1. Buatlah file **Lingkaran.php** sebagai file class lingkaran dengan konsep class seperti di bawah ini:



* 1. Buatlah file **hitungLingkaran.php** untuk menampung nilai jari-jari dari formLingkaran.html dan menghitung luasnya dengan memanfaatkan class lingkaran.php hingga hasilnya sebagai berikut:

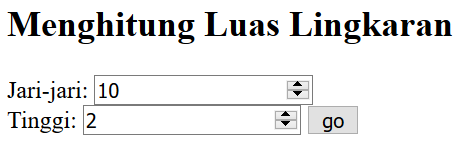


* 1. Buatlah file **Tabung.php** sebagai class anak yang menginduk ke class lingkaran dengan konsep seperti dibawah ini:

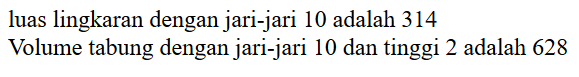


* 1. Tambahkan pada FormLingkaran.html elemen input untuk menampung tinggi dan tambahkan program pada hitungLingkaran.php untuk menampilkan luas lingkaran dan volume tabung.

1. Tampilan form:



1. Tampilan output



1. Perubahan Source code **hitungLingkaran.php**



* 1. Soal kasus

Buatlah program sederhana berbasis OOP untuk menghitung luas persegi dan volume balok.

