# MODUL PRAKTIUM

**PEMROGRAMAN BERORIENTASI OBJECT**

**KOM63033**

****

**TIM PENYUSUN**

**(“ASISTEN PRAKTIKUM”)**

**20191 - SEMESTER GANJIL 2019/2020**

**PROGRAM STUDI ILMU KOMPUTER**

**JURUSAN MATEMATIKA**

**FAKULTAS MATEMATIKA DAN ILMU PENGETAHUAN ALAM**

**UNIVERSITAS HALU OLEO**

**KENDARI**

**2019**

# Praktikum 1

1. 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.

1. Fungsi Pemrograman Berbasis Objek dalam PHP

Manfaat dan tujuan dari OOP itu sendiri adalah untuk mempermudah pengembangan suatu program, didalam OOP itu sendiri memiliki konsep yang dikenal beberapa istilah umum, yaitu Attribute.

1. Konsep OOP

* Abstrak Class

1. Kelas merupakan deskripsi abstrak informasi dan tingkah laku dari sekumpulan data.
2. Kelas dapat diilustrasikan sebagai suatu cetak biru(blueprint) atau prototipe yang digunakan untuk menciptakan objek.
3. Kelas merupakan tipe data bagi objek yang mengenkapsulasi data dan operasi pada data dalam suatu unit tunggal.
4. Kelas mendefinisikan suatu struktur yang terdiri atas data kelas (data field), prosedur atau fungsi (method), dan sifat kelas (property).

* Encapsulation

1. Istilah enkapsulasi sebenarnya adalah kombinasi data dan fungsionalitas dalam sebuah unit tunggal sebagai bentuk untuk menyembunyikan detail informasi.
2. Proses enkapsulasi memudahkan kita untuk menggunakan sebuah objek dari suatu kelas karena kita tidak perlu mengetahui segala hal secara rinci.
3. Enkapsulasi menekankan pada antarmuka suatu kelas, atau dengan kata lain bagaimana menggunakan objek kelas tertentu.
4. Contoh: kelas mobil menyediakan antarmuka fungsi untuk menjalankan mobil tersebut, tanpa kita perlu tahu komposisi bahan bakar, udara dan kalor yang diperlukan untuk proses tersebut.

* Inheritance

1. Kita dapat mendefinisikan suatu kelas baru dengan mewarisi sifat dari kelas lain yang sudah ada.
2. Penurunan sifat ini bisa dilakukan secara bertingkattingkat, sehingga semakin ke bawah kelas tersebut menjadi semakin spesifik.
3. Sub kelas memungkinkan kita untuk melakukan spesifikasi detail dan perilaku khusus dari kelas supernya.
4. Dengan konsep pewarisan, seorang programmer dapat menggunakan kode yang telah ditulisnya pada kelas super berulang kali pada kelas-kelas turunannya tanpa harus menulis ulang semua kodekode itu.

* Polimorfisme

1. Polimorfisme merupakan kemampuan objekobjek yang berbeda kelas namun terkait dalam pewarisan untuk merespon secara berbeda terhadap suatu pesan yang sama.
2. Polimorfisme juga dapat dikatakan kemampuan sebuah objek untuk memutuskan method mana yang akan diterapkan padanya, tergantung letak objek tersebut pada jenjang pewarisan.
3. Method overriding.
4. Method name overloading.

# Praktikum 2

1. Pengertian Class dalam Pemrograman Berbasis Objek

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.

<?php

class laptop {

   // isi dari class laptop...

}

?>

1. Property dalam Pemrograman Berbasis Objek

**Property** (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah **class**.Melanjutkan analogi tentang laptop, **property** dari laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain.

<?php

classlaptop {

   var$pemilik;

   var$merk;

   var$ukuran\_layar;

// lanjutan isi dari class leptop

?>

1. 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**, dan berbagai tindakan lain.

<?php

classlaptop {

var $on

  functionhidupkan\_laptop() {

 //... isi dari method matikan\_laptop }

}?>

1. Object dalam Pemrograman Berbasis Objek

**Object** atau **Objek** adalah hasil cetak dari **class**, atau hasil ‘konkrit’ dari **class**. Jika menggunakan analogi **class laptop**, maka objek dari **class laptop** bisa berupa: laptop\_andi, laptop\_anto, , dan lain-lain. Objek dari **class laptop**akan memiliki seluruh ciri-ciri **laptop**, yaitu property dan method-nya.

<?php

classlaptop {

   //... isi dari class laptop

   }

$laptop\_andi= newlaptop();

$laptop\_andi->merk;

?>

?>

# Praktikum 3

1. Array dengan cara biasa

Array adalah suatu tipe data variabel yang mampu menyimpan banyak data atau nilai.

Karena array memiliki banyak data, maka isi dari array di tandai dengan angka pada tiap-tiap isi dari array. angka pada masing-masing isi array di mulai dari 0,1,2,3 sampai seterusnya. Ada dua cara penamaan atau pemanggilan isi array. yaitu dengan cara memanggil menggunakan angka default array, dan bisa juga dengan memberikan penamaan pada masing-masing isi array.

<?php

//membuat array yang berisi nama buah-buahan

$buah = array('semangka','jeruk','apel','anggur');

//menampilkan data array dengan nomor urut 2

echo $buah[2];

?>

1. Array Membuat dan memberi penamaan pada isi array

<?php

//penamaan isi array variabel buah

$buah = array(

'semangka' => "isinya merah",

'jeruk' => "rasanya manis",

'apel' => "warnanya merah",

'anggur' => "harganya mahal"

);

// menampilkan isi array yang bernama jeruk

echo $buah['jeruk'];

?>

1. Contoh menerapkan array dengan for

<?php

//membuat array yang berisi nama buah-buahan

$buah = array('semangka','jeruk','apel','anggur');

//count() untuk menghitung isi array.

for($x=0;$x<count($buah);$x++){

echo $buah[$x]."<br/>";

}

?>

1. Tugas (Implementasikan Array dalam OOP)

# Praktikum 4

* 1. Pengertian Enkapsulasi (Encapsulation)

Enkapsulasi (encapsulation) adalah sebuah metoda untuk mengatur struktur class dengan cara menyembunyikan alur kerja dari class tersebut.

* 1. Hak Akses: **Public**

Ketika sebuah property atau method dinyatakan sebagai **public**, maka seluruh kode program di luar class bisa mengaksesnya, termasuk class turunan.

<?php

// buat class laptop

classlaptop {

   // buat public property

   public$pemilik;

   // buat public method

   publicfunctionhidupkan\_laptop() {

     return"Hidupkan Laptop";

   }

}

// buat objek dari class laptop (instansiasi)

$laptop\_anto= newlaptop();

// set property

$laptop\_anto->pemilik="Anto";

// tampilkan property

echo$laptop\_anto->pemilik; // Anto

// tampilkan method

echo$laptop\_anto->hidupkan\_laptop(); // "Hidupkan Laptop"

?>

* 1. 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.

Apabila kita mencoba mengakses **protected**property atau **protected**method dari luar class, akan menghasilkan **error**, seperti contoh berikut ini:

<?php

// buat class laptop

classlaptop {

  // buat protected method

   protectedfunctionhidupkan\_laptop() {

      return"Hidupkan Laptop";

   }

}

// buat objek dari class laptop (instansiasi)

$laptop\_anto= newlaptop();

// jalankan protected method akan menghasilkan error

echo$laptop\_anto->hidupkan\_laptop();

// Fatal error: Call to protected method laptop::hidupkan\_laptop()

?>

Agar bisa diakses ***method*** hidupkan\_laptop() yang tidak bisa diakses secara langsung. Saya menambahkan method paksa\_hidup() yang secara internal akan mengakses method hidupkan\_laptop().

<?php

// buat class laptop

classlaptop {

  // buat protected method

   protectedfunctionhidupkan\_laptop() {

      return"Hidupkan Laptop";

   }

publicfunctionpaksa\_hidup() {

      return$this->hidupkan\_laptop();

   }

}

// buat objek dari class laptop (instansiasi)

$laptop\_anto= newlaptop();

// jalankan method paksa\_hidup()

echo$laptop\_anto->paksa\_hidup(); // "Hidupkan Laptop"

?>

* 1. 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.

<?php

// buat class komputer

classkomputer {

   // property dengan hak akses protected

   private$jenis\_processor= "Intel Core i7-4790 3.6Ghz";

   publicfunctiontampilkan\_processor() {

     return$this->jenis\_processor;

   }

}

// buat class laptop

classlaptop extendskomputer{

   publicfunctiontampilkan\_processor() {

     return$this->jenis\_processor;

   }

}

// buat objek dari class laptop (instansiasi)

$komputer\_baru= newkomputer();

$laptop\_baru= newlaptop();

// jalankan method dari class komputer

echo$komputer\_baru->tampilkan\_processor(); // "Intel Core i7-4790 3.6Ghz"

// jalankan method dari class laptop (error)

echo$laptop\_baru->tampilkan\_processor();

// Notice: Undefined property: laptop::$jenis\_processor

?>

Akses level **private** sering digunakan untuk menyembunyikan property dan method agar tidak bisa diakses di luar class.

# Praktikum 5

* 1. **Constructor**

(bahasa indonesia: konstruktor) adalah method khusus yang akan dijalankan secara otomatis pada saat sebuah objek dibuat (instansiasi), yakni ketika perintah “**new**” dijalankan.

<?php

// buat class laptop

classlaptop {

   private$pemilik= "Andi";

   private$merk= "Lenovo";

   publicfunction\_\_construct(){

     echo"Ini berasal dari Constructor Laptop";

   }

   publicfunctionhidupkan\_laptop(){

     return"Hidupkan Laptop $this->merk punya $this->pemilik";

// buat objek dari class laptop (instansiasi)

$laptop\_andi= newlaptop();

echo"<br />";

echo$laptop\_andi->hidupkan\_laptop();

?>

* 1. Contoh constructor yang sering digunakan untuk membuat objek dengan nilai awal. Konsep ini sering digunakan dalam pemrograman objek.

<?php

// buat class laptop

classlaptop {

   // constructor sebagai pembuat nilai awal

   publicfunction\_\_construct($pemilik, $merk) {

     $this->pemilik = $pemilik;

     $this->merk = $merk;

   }

   publicfunctionhidupkan\_laptop() {

     return"Hidupkan Laptop $this->merk punya $this->pemilik";

   }

}

// buat objek dari class laptop (instansiasi)

$laptop\_andi= newlaptop("Andi", "Lenovo");

echo$laptop\_andi->hidupkan\_laptop();

echo"<br />";

$laptop\_anto= newlaptop("Anto", "Acer");

echo$laptop\_anto->hidupkan\_laptop();

?>

Di dalam PHP, **constructor** dan **destructor**harus memiliki hak akses **public**. Jika anda mengubah hak akses **constructor** atau **destructor**menjadi **protected** atau **private**, PHP akan mengeluarkan error

# Praktikum 6

* 1. **Inheritance**

**Inheritance** atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ **property**dan **method** yang dimilikinya kepada **class**lain. Konsep **inheritance**digunakan untuk memanfaatkan fitur ‘code reuse’ untuk menghindari duplikasi kode program.

<?php

// buat class induk: komputer

classkomputer {

   public$merk;

   public$processor;

   public$memory;

   publicfunctionbeli\_komputer() {

     return"Beli komputer baru";

   }

}

// turunkan class komputer ke laptop

classlaptop extendskomputer {

   publicfunctionlihat\_spec() {

     return"merk: $this->merk, processor: $this->processor,

     memory: $this->memory";

   }

}

// buat objek dari class laptop (instansiasi)

$laptop\_baru= newlaptop();

// isi property objek

$laptop\_baru->merk = "acer";

$laptop\_baru->processor ="intel core i5";

$laptop\_baru->memory = "2 GB";

//panggil method objek

echo$laptop\_baru->beli\_komputer();

echo"<br />";

echo$laptop\_baru->lihat\_spec();

?>

# Praktikum 7

1. Abstrak Class dan Abstract Method

**Abstract Class** adalah sebuah class yang tidak bisa di-instansiasi (tidak bisa dibuat menjadi objek) dan berperan sebagai ‘kerangka dasar’ bagi class turunannya. Di dalam abstract class umumnya akan memiliki abstract method.

Abstract class digunakan di dalam **inheritance** (pewarisan class) untuk ‘memaksakan’ implementasi method yang sama bagi seluruh class yang diturunkan dari abstract class. Abstract class digunakan untuk membuat struktur logika penurunan di dalam pemrograman objek.

**Abstract Method** adalah sebuah ‘method dasar’ yang harus diimplementasikan ulang di dalam class anak (child class).Abstract method ditulis tanpa isi dari method, melainkan hanya ‘**signature’**-nya saja.**Signature** dari sebuah method adalah bagian method yang terdiri dari nama method dan parameternya (jika ada).

* Cara Membuat Abstract Class

Karena kita ingin membuat class komputer sebagai **abstract class**, maka berikut adalah cara penulisannya di dalam PHP:

<?php

abstractclasskomputer {

   // isi dari class komputer

}

?>

* Cara Membuat Abstract Method

abstractpublicfunctionlihat\_spec();

abstractpublicfunctionlihat\_spec($merk);

Kenapa kita tidak perlu membuat isi dari method? Ini karena jika sebuah method dinyatakan sebagai **abstract method**, isi dari method tersebut akan dibuat dalam class turunan. Abstract method**harus** berada di dalam abstract class.

berikut adalah cara penulisan abstract method lihat\_spec() di dalam abstract classkomputer:

<?php

abstractclasskomputer {

   abstractpublicfunctionlihat\_spec();

}

?>

* Abstract Class Tidak Bisa Diinstansiasi

abstractclasskomputer {

   abstractpublicfunctionlihat\_spec($pemilik);

   }

}

$komputer\_baru=newkomputer();

// Fatal error: Cannot instantiate abstract class komputer

?>

* Abstract Class Bisa Memiliki Property dan Method ‘biasa’

<?php

// buat abstract class

abstractclasskomputer{

   // buat abstract method

   abstractpublicfunctionlihat\_spec($pemilik);

   // buat method ‘biasa’

   publicfunctionhidupkan\_komputer(){

     echo"Hidupkan Komputer";

   }

}

?>

* Class Turunan Harus Mengimplementasikan Abstract Method

<?php

// buat abstract class

abstractclasskomputer{

   // buat abstract method

   abstractpublicfunctionlihat\_spec();

}

classlaptop extendskomputer{

// implementasi abstract method

   publicfunctionlihat\_spec(){

     return"Lihat Spec Laptop...";

   }

// method 'biasa'

   publicfunctionbeli\_laptop(){

     return"Beli Laptop...";

   }

}

// buat objek dari class laptop

$laptop\_baru= newlaptop();

echo$laptop\_baru->lihat\_spec();

// Lihat Spec Laptop...

echo"<br />";

echo$laptop\_baru->beli\_laptop();

// Beli Laptop...

?>

# Praktikum 8

1. Pengertian Polimorfisme

Di dalam pemrograman objek, **polimorfisme** adalah konsep dimana terdapat banyak class yang memiliki signature method yang sama. Implementasi dari method-method tersebut diserahkan kepada tiap class, akan tetapi cara pemanggilan method harus sama. Agar kita dapat ‘memaksakan’ signature method yang sama pada banyak class, class tersebut harus diturunkan dari sebuah **abstract class** atau **object interface**.

1. Cara membuat polimorfisme

<?php

abstractclasskomputer{

   abstractpublicfunctionbooting\_os();

}

classlaptop extendskomputer{

   publicfunctionbooting\_os(){

     return"Proses Booting Sistem Operasi Laptop";

   }

}

classpc extendskomputer{

   publicfunctionbooting\_os(){

     return"Proses Booting Sistem Operasi PC";

   }

}

classchromebook extendskomputer{

publicfunctionbooting\_os(){

     return"Proses Booting Sistem Operasi Chromebook";

   }

}

// buat objek dari class diatas

$laptop\_baru= newlaptop();

$pc\_baru= newpc();

$chromebook\_baru= newchromebook();

// buat fungsi untuk memproses objek

functionbooting\_os\_komputer($objek\_komputer){

   return$objek\_komputer->booting\_os();

}

echobooting\_os\_komputer($laptop\_baru);

echo"<br />";

echobooting\_os\_komputer($pc\_baru);

echo"<br />";

echobooting\_os\_komputer($chromebook\_baru);

?>

# Praktikum 9

1. Pengertian CRUD

CRUD adalah akronim untuk Create, Read, Update, dan Delete.Operasi CRUD adalah manipulasi data dari database.

1. Membuat CRUD sederhana

* Membuat file config.php

<?php

/\*\*

\* using mysqli\_connect for database connection

\*/

$databaseHost = 'localhost';

$databaseName = 'crud\_db';

$databaseUsername = 'root';

$databasePassword = '';

$mysqli = mysqli\_connect($databaseHost, $databaseUsername, $databasePassword, $databaseName);

?>

* Membuat file form add data dan query insert

<form action="add.php" method="post" name="form1">

<table width="25%" border="0">

<tr>

<td>Name</td>

<td><input type="text" name="name"></td>

</tr>

<tr>

<td>Email</td>

<td><input type="text" name="email"></td>

</tr>

<tr>

<td>Mobile</td>

<td><input type="text" name="mobile"></td>

</tr>

<tr>

<td></td>

<td><input type="submit" name="Submit" value="Add"></td>

</tr>

</table>

</form>

// Insert user data into table

$result = mysqli\_query($mysqli, "INSERT INTO users(name,email,mobile) VALUES('$name','$email','$mobile')");

* Form edit data dan query update

// update user data

$result = mysqli\_query($mysqli, "UPDATE users SET name='$name',email='$email',mobile='$mobile' WHERE id=$id");

// Fetech user data based on id

$result = mysqli\_query($mysqli, "SELECT \* FROM users WHERE id=$id");

//form

<form name="update\_user" method="post" action="edit.php">

<table border="0">

<tr>

<td>Name</td>

<td><input type="text" name="name" value=<?php echo $name;?>></td>

</tr>

* Query delete

// Get id from URL to delete that user

$id = $\_GET['id'];

// Delete user row from table based on given id

$result = mysqli\_query($mysqli, "DELETE FROM users WHERE id=$id");

# Praktikum 9

1. Pengertian dan konsep MVC

MVC adalah konsep arsitektur dalam pembangunan aplikasi berbasis web yang membagi aplikasi web menjadi 3 bagian besar. Yang mana setiap bagian memiliki tugas-tugas serta tanggung jawab masing-masing. Tiga bagian tersebut adalah: model, view dan controller.

* **Model**: Bertugas untuk mengatur, menyiapkan, memanipulasi dan mengorganisasikan data (dari database) sesuai dengan instruksi dari controller.
* **View**: Bertugas untuk menyajikan informasi (yang mudah dimengerti) kepada user sesuai dengan instruksi dari controller.
* **Controller**: Bertugas untuk mengatur apa yang harus dilakukan model, dan view mana yang harus ditampilkan berdasarkan permintaan dari user. Namun, terkadang permintaan dari user tidak selalu memerlukan aksi dari model. Misalnya seperti menampilkan halaman form untuk registrasi user.

**Contoh kasus :**

* 1. Ketika anda melihat layar komputer, di sana ditampilkan form untuk mengisi data diri pengunjung. Kita bisa simpulkan bahwa di sini, browser (sebagai client) meminta aksi kepada server (yang ditangani oleh controller) untuk menampilkan halaman form input data. Lalu controller memutuskan dan mengerti bahwasanya ia hanya perlu menampilkan view. Maka controller memanggil dan mengembalikan view (atau halaman) yang diminta.
  2. Anda kemudian mengisi data. Lalu menekan tombol submit. Di sini browser (sebagai client) mengirimkan data anda kepada server. Data itu ditangkap oleh controller dan controller tahu apa yang harus dia lakukan. Yaitu memanggil model dan memberi data tersebut untuk kemudian disimpan (oleh model) ke dalam database. Setelah proses penyimpanan selesai, controller memanggil dan mengembalikan view kepada user sebagai informasi bahwa data telah masuk.
  3. Jika ada pengunjung baru, maka kembali ke step 1

1. Praktikum Pengenalan Framework berbasis MVC

* CodeIgniter
* Yii
* Laravel