MODUL PRAKTIKUM

PEMROGRAMAN BERORIENTASI OBJEK

DISUSUN OLEH :

MOH. ERKAMIM, S.Kom., M.Kom

PROGRAM STUDI D4 SISTEM INFORMASI KOTA CERDAS

FAKULTAS TEKNIK

UNIVERSITAS TUNAS PEMBANGUNAN

SURAKARTA

HALAMAN PENGESAHAN

Matakuliah	: Pemrograman Berorientasi Objek
Kode Dokumen	: VSI3318/02/2023
Jenis Bahan Ajar	: Modul Praktikum
Program Studi	: D4 Sistem Informasi Kota Cerdas
Fakultas	: Teknik
Perguruan Tinggi	: Universitas Tunas Pembangunan Surakarta
Penyusun	: Moh. Erkamim, S.Kom., M.Kom

Modul ini disusun sebagai bahan ajar atau petunjuk praktikum dalam mata kuliah **Pemrograman Berorientasi Objek.**

Disahkan
Surakarta, 4 September 2023
Menyetujui
Ketua Program Studi
D4 Sistem Informasi Kota Cerdas

Moh. Erkamim, S.Kom., M.Kom

Mengesahkan
Dekan Fakultas Teknik

Dr. Tri Hartanto, S.T., M.Sc

DAFTAR ISI

HALAMAN PENGESAHAN ii
DAFTAR ISI iii
TATA TERTIB iv
Modul 1 Pengenalan Pemrograman Berbasis Objek (PBO) 1
Modul 2 Dasar Pemrograman Berbasis Objek PHP 4
Modul 3 Pewarisan (*Inheritance*) 8
Modul 4 Modifier dalam PHP 12
Modul 5 Enkapsulasi 15
Modul 6 Polimorfisme 18
Modul 7 Abstraksi dalam PHP 21
Modul 8 Namespace dan Autoloading dalam PHP 23
Modul 9 Penanganan Kesalahan dengan Exception 26
Modul 10 Advanced OOP Features 29
Modul 11 Operasi Database CRUD 32
Modul 12 Proyek Akhir 35

TATA TERTIB
Praktikum Menggunakan Laboratorium Pemrograman
Program Studi Sistem Informasi Kota Cerdas
Fakultas Teknik Universitas Tunas Pembangunan Surakarta

Praktikum Laboratorium Pemrograman adalah salah satu kegiatan penting dalam Program Studi Sistem Informasi Kota Cerdas. Agar praktikum berjalan dengan lancar dan efisien, serta menjaga keselamatan dan kedisiplinan, berikut adalah tata tertib yang harus diikuti oleh seluruh mahasiswa yang mengikuti praktikum ini:

1. Pendaftaran dan Presensi:
   - ✓ Setiap mahasiswa wajib mendaftar sebelum memulai praktikum.
   - ✓ Mahasiswa harus mencatat kehadiran mereka dalam buku presensi yang telah disediakan.

2. Pakaian:
   - ✓ Menggunakan pakaian yang sesuai dengan aturan laboratorium.
   - ✓ Tidak diperkenankan mengenakan sandal jepit, sandal tidur, atau pakaian yang tidak pantas.

3. Perlengkapan:
   - ✓ Pastikan membawa alat tulis, buku catatan, dan laptop (jika diperlukan) sesuai dengan petunjuk dosen.
   - ✓ Dilarang membawa makanan atau minuman ke dalam laboratorium.

4. Kedisiplinan:
   - ✓ Mahasiswa diharapkan tiba tepat waktu sesuai jadwal praktikum.
   - ✓ Dilarang berbicara keras atau mengganggu mahasiswa lain selama praktikum berlangsung.
   - ✓ Matikan telepon seluler atau atur ke mode senyap selama praktikum.

5. Keselamatan:
   - ✓ Ikuti semua petunjuk keamanan yang diberikan oleh dosen.
   - ✓ Tidak diperkenankan mencolokkan atau mencabut kabel-kabel tanpa izin.
   - ✓ Hindari mengganggu peralatan laboratorium yang tidak relevan dengan praktikum yang sedang berlangsung.

6. Penggunaan Perangkat Keras dan Perangkat Lunak:
   - ✓ Gunakan perangkat keras dan perangkat lunak laboratorium dengan bijak.
   - ✓ Jangan mencoba mengakses atau mengubah perangkat keras atau perangkat lunak laboratorium tanpa izin.

7. Tugas dan Proyek:
   - ✓ Serahkan tugas atau proyek sesuai dengan jadwal yang telah ditentukan oleh dosen.
   - ✓ Jangan melakukan plagiarisme atau menyalin pekerjaan orang lain.

8. Perilaku Etis:
   - ✓ Hormati hak dan privasi mahasiswa dan dosen lain.
   - ✓ Jangan menyebarkan informasi pribadi mahasiswa atau dosen tanpa izin.

9. Keteraturan Ruangan:
   - ✓ Setelah selesai praktikum, pastikan meja dan area kerja Anda dalam keadaan bersih dan rapi.
   - ✓ Pastikan semua perangkat yang digunakan telah dimatikan sebelum meninggalkan laboratorium.

10. Sanksi:
    - ✓ Pelanggaran terhadap tata tertib ini dapat mengakibatkan sanksi, seperti penurunan nilai atau pembatalan hak untuk mengikuti praktikum.

Semua mahasiswa diharapkan mematuhi tata tertib ini untuk menjaga keamanan, ketertiban, dan kelancaran praktikum laboratorium pemrograman.

Modul 1 Pengenalan Pemrograman Berorientasi Objek (PBO)

1. Tujuan

a. Memahami konsep dasar PBO dan perbedaannya dengan pemrograman prosedural.
b. Mengenal dan mampu mendefinisikan class dan objek dalam PBO.
c. Memahami sifat (properties) dan metode (methods) yang digunakan dalam class.
d. Mampu membuat dan menggunakan objek dalam bahasa pemrograman.
e. Memahami penggunaan constructor dan destructor dalam class.
f. Mengerti penggunaan kata kunci "this" dalam PBO.

2. Dasar Teori

A. Konsep dasar PBO

Pemrograman Berorientasi Objek (PBO) adalah paradigma pemrograman di mana segala sesuatunya dianggap sebagai objek. Objek adalah entitas yang memiliki properti (atau atribut) dan perilaku (disebut metode).

Empat Pilar Pemrograman Berbasis Objek (PBO):

1) **Enkapsulasi:** Proses menyembunyikan detil implementasi dan hanya menunjukkan operasi yang relevan kepada pengguna.
2) **Pewarisan (Inheritance):** Mekanisme dimana satu kelas dapat mewarisi properti dan metode dari kelas lain.
3) **Polimorfisme:** Kemampuan objek untuk mengambil banyak bentuk.
4) **Abstraksi:** Fokus pada karakteristik esensial sebuah objek, mengabaikan yang tidak esensial.

B. Keuntungan PBO

1) **Reusabilitas Kode:** Kode yang sudah ada dapat digunakan kembali.
2) **Pemeliharaan Mudah:** Memudahkan dalam pengembangan dan perbaikan kode.
3) **Modularitas:** Kode lebih terorganisir, mudah dipahami, dan diuji.
4) **Fleksibilitas:** Dengan pewarisan, perubahan kecil pada kelas induk dapat menghasilkan perubahan besar pada kelas turunan.
5) **Peningkatan Kolaborasi:** Lebih mudah bagi tim untuk bekerja bersama karena mereka dapat fokus pada objek tertentu.

C. Class dan objek dalam PHP

1) **Class:** Rancangan atau *blueprint* dari objek, untuk mendefinisikan properti dan metode yang akan dimiliki objek.
2) **Objek:** Instansi dari *class*, objek memiliki properti dan perilaku yang didefinisikan oleh class.

```
class Mobil {
  // Properti
  public $warna;
```

```
    public $merk;

    // Metode
    public function hidupkanMesin() {
        echo "Mesin Hidup!";
    }
}
// Membuat objek dari class Mobil
$avanza = new Mobil();
```

D. Properti dan metode

1) **Properti:** Variabel yang ada di dalam class, merepresentasikan data/atribut dari objek.

```
// Properti
class Hewan {
    public $jenis;
    public $kaki;
}
```

2) **Metode:** Fungsi yang ada di dalam class merepresentasikan perilaku dari objek.

```
// Properti
class Hewan {
    public $jenis;
    public $kaki;

// Metode
    public function berjalan() {
        echo $this->jenis . " sedang berjalan dengan " . $this->kaki . " kaki.";
    }
}
$anjing = new Hewan();
$anjing->jenis = "Anjing";
$anjing->kaki = 4;
$anjing->berjalan();  // Output: Anjing sedang berjalan dengan 4 kaki.
```

3. Kegiatan Praktikum

✓ **Class dan Objek dalam PHP**

Buatlah sebuah class Pelajar:

```
class Pelajar {
}
```

Kemudian, buat sebuah objek dari class tersebut:

```
$andi = new Pelajar();
```

✓ **Properti dan Metode**

Tambahkan beberapa properti ke dalam class Pelajar seperti nama, umur, dan kelas.

```
class Pelajar {
   public $nama;
   public $umur;
   public $kelas;
}
```

Kemudian, tambahkan sebuah metode perkenalan yang akan mencetak data pelajar:

```
class Pelajar {
   public $nama;
   public $umur;
   public $kelas;

   public function perkenalan() {
   echo "Halo, nama saya " . $this->nama . ". Saya berumur " . $this->umur . " tahun dan saat ini saya duduk di kelas " . $this->kelas . ".<br>";
   }
}
```

Sekarang, buat sebuah objek dari class Pelajar, atur nilai properti, dan panggil metode perkenalan:

```
$andi = new Pelajar();
$andi->nama = "Andi";
$andi->umur = 16;
$andi->kelas = "10A";
$andi->perkenalan();
```

4. Tugas / Latihan

1. Buatlah class dengan nama "Buku" dengan properti "judul" dan "penulis". Tambahkan metode "cetakInfo" yang akan mencetak judul dan penulis buku tersebut.
2. Instansiasi objek dari class "Buku", berikan nilai pada properti, dan panggil metode "cetakInfo".

Modul 2 Dasar Pemrograman Berbasis Objek PHP

1. Tujuan

a. Memahami konsep dasar kelas dan objek dalam PHP.
b. Mengenal dan memahami fungsi dari properti dan metode dalam PBO PHP.
c. Mampu membuat dan menggunakan objek dalam PHP.
d. Memahami tujuan dan penggunaan dari constructor dan destructor.
e. Mengetahui cara kerja dan penggunaan dari this keyword.

2. Dasar Teori

A. Kelas dan Objek

1) **Kelas (Class):** Adalah *blueprint* atau cetakan untuk membuat objek. Menentukan apa yang harus dimiliki objek berdasarkan atribut dan perilaku.
2) **Objek:** Adalah instansi dari *class*, merupakan realisasi dari apa yang didefinisikan di dalam class.

B. Sifat (*Properties*) dan Metode (*Methods*)

1) ***Properties*:** Variabel yang terdapat di dalam class. Merepresentasikan data yang bisa dimiliki objek.
2) ***Methods*:** Fungsi yang terdapat di dalam class. Merepresentasikan perilaku dari objek.

C. Pembuatan dan Penggunaan Objek 3.4

Objek dibuat dengan menggunakan keyword new diikuti dengan nama class.
$objekBaru = new NamaKelas();

D. *Constructor* dan *Destructor*

1) **Constructor:** Metode khusus yang akan otomatis dijalankan ketika objek dari class dibuat. Biasanya digunakan untuk inisialisasi properti.

```
class Mobil {
   public function __construct() {
      echo "Mobil telah dibuat!";
   }
}
```

2) **Destructor:** Metode khusus yang akan otomatis dijalankan ketika objek dihapus atau keluar dari lingkup. Biasanya digunakan untuk cleanup atau pembebasan sumber daya.

```
class Mobil {
   public function __destruct() {
      echo "Mobil telah dihancurkan!";
   }
}
```

E. *this keyword*

Digunakan di dalam class untuk merujuk ke objek saat ini. Sangat berguna untuk mengakses properti dan metode dalam class yang sama.

```
class Buku {
   public $judul = "Belum Ada Judul";

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

3. Kegiatan Praktikum

1) **Pembuatan Kelas:** Buatlah kelas Mahasiswa dengan properti nama, nim, dan jurusan.
2) **Menambahkan Metode:** Tambahkan metode cetakData yang mencetak informasi mahasiswa.
3) **Membuat Objek:** Buat objek dari kelas Mahasiswa, lalu atur propertinya dan panggil metode cetakData.
4) **Menerapkan Constructor:** Modifikasi kelas Mahasiswa dengan menambahkan constructor yang menerima nama, nim, dan jurusan sebagai parameter.
5) **Penggunaan this keyword:** Modifikasi metode di kelas Mahasiswa dengan menggunakan this keyword untuk mengakses properti.

✓ **Pembuatan Kelas:**

```
class Mahasiswa {
   public $nama;
   public $nim;
   public $jurusan;
   public function cetakData() {
      echo "Nama: " . $this->nama . "<br>";
      echo "NIM: " . $this->nim . "<br>";
      echo "Jurusan: " . $this->jurusan . "<br>";
   }
}
```

✓ **Menambahkan Metode:**

Metode cetakData telah ditambahkan pada langkah pertama.

✓ **Membuat Objek:**

```
$mahasiswa1 = new Mahasiswa();
$mahasiswa1->nama = "Andi";
$mahasiswa1->nim = "12345678";
$mahasiswa1->jurusan = "Teknik Informatika";
$mahasiswa1->cetakData();
```

✓ **Menerapkan Constructor:**

```
class Mahasiswa {
    public $nama;
    public $nim;
    public $jurusan;
    public function __construct($nama, $nim, $jurusan) {
        $this->nama = $nama;
        $this->nim = $nim;
        $this->jurusan = $jurusan;
    }
    public function cetakData() {
        echo "Nama: " . $this->nama . "<br>";
        echo "NIM: " . $this->nim . "<br>";
        echo "Jurusan: " . $this->jurusan . "<br>";
    }
}
$mahasiswa2 = new Mahasiswa("Budi", "87654321", "Sistem Informasi");
$mahasiswa2->cetakData();
```

✓ **Penggunaan this keyword:**

Pada contoh kode di atas, kita sudah menggunakan this keyword di dalam metode cetakData dan __construct.

✓ **Contoh Source**

```
class Persegi {
```

```
    public $panjang;
    public $lebar;
    public function __construct($panjang, $lebar) {
        $this->panjang = $panjang;
        $this->lebar = $lebar;
    }
    public function hitungLuas() {
        return $this->panjang * $this->lebar;
    }
    public function hitungKeliling() {
        return 2 * ($this->panjang + $this->lebar);
    }
}
$persegi1 = new Persegi(10, 5);
echo "Luas Persegi: " . $persegi1->hitungLuas() . "<br>";
echo "Keliling Persegi: " . $persegi1->hitungKeliling() . "<br>";
```

4. Tugas

1) Buatlah class Persegi dengan properti panjang dan lebar. Tambahkan metode untuk menghitung luas dan keliling persegi tersebut.
2) Buatlah objek dari class Persegi, atur propertinya, dan tampilkan hasil perhitungan luas dan keliling dengan memanggil metodenya.
3) Modifikasi class Persegi dengan menambahkan constructor yang menerima panjang dan lebar sebagai parameter.

Modul 3 Pewarisan (*Inheritance*)

1. Tujuan

a. Mengerti dan memahami konsep pewarisan dalam pemrograman berorientasi objek.
b. Dapat menerapkan overriding metode dari superclass.
c. Memahami dan menggunakan keyword parent untuk merujuk ke superclass.
d. Mengetahui perbedaan antara single inheritance dan multiple inheritance serta bagaimana menerapkannya dalam kode program.

2. Dasar Teori

A. Konsep Pewarisan

Pewarisan adalah salah satu pilar dalam pemrograman berorientasi objek. Ia memungkinkan kita untuk mendefinisikan sebuah kelas baru berdasarkan kelas yang sudah ada sebelumnya. Kelas yang diwarisi disebut sebagai superclass atau parent class, sementara kelas yang mewarisi disebut subclass atau child class. Tujuan utama dari pewarisan adalah untuk mempromosikan reusability kode dan menetapkan hubungan antara superclass dan subclass.

B. Overriding Metode

Overriding adalah fitur dimana subclass memiliki metode dengan nama yang sama seperti yang ada pada superclass tetapi dengan implementasi yang berbeda. Ini memungkinkan subclass untuk menyediakan implementasi khusus dari metode yang sudah didefinisikan pada superclass.

C. Penggunaan keyword parent

Keyword parent digunakan untuk merujuk ke metode atau properti dari superclass. Ini sangat berguna terutama saat melakukan overriding metode tetapi masih ingin memanggil metode dari superclass.

D. Tipe Pewarisan: Single dan Multiple Inheritance

1) **Single Inheritance**: Sebuah kelas hanya dapat mewarisi dari satu kelas lain. PHP mendukung single inheritance.
2) **Multiple Inheritance**: Sebuah kelas dapat mewarisi fitur dari lebih dari satu kelas. PHP tidak mendukung multiple inheritance secara langsung, tetapi dapat disimulasikan dengan menggunakan trait.

3. Kegiatan Praktikum

1) Membuat Superclass

Pertama-tama, kita akan membuat sebuah superclass.

Buat file dengan nama Hewan.php.

```php
<?php
class Hewan {
    public $jenis;
    public function __construct($jenis) {
        $this->jenis = $jenis;
    }
    public function suara() {
        return "Suara hewan ini adalah...";
    }
}
?>
```

2) Membuat Subclass

Sekarang, mari kita buat sebuah subclass yang mewarisi dari class Hewan.

Buat file dengan nama Kucing.php.

```php
<?php
require_once "Hewan.php";
class Kucing extends Hewan {
    // Overriding Metode suara
    public function suara() {
        return "Meong!";
    }
    // Menggunakan keyword parent
    public function deskripsi() {
        return "Ini adalah jenis " . $this->jenis . " dan bersuara: " . parent::suara();
    }
}
?>
```

3) Melakukan Testing

Mari kita coba kode tersebut untuk memahami konsep pewarisan dan overriding.

Buat file dengan nama index.php.

```
<?php
require_once "Kucing.php";
$kucingAnggora = new Kucing("Anggora");
echo $kucingAnggora->suara(); // Harusnya menampilkan "Meong!"
echo "<br>";
echo $kucingAnggora->deskripsi(); // Deskripsi jenis kucing dan suaranya
?>
```

4) Eksplorasi Multiple Inheritance dengan Trait

Meskipun PHP tidak mendukung multiple inheritance, kita bisa menggunakan trait untuk mensimulasikannya.

Buat dua file trait, Trait1.php dan Trait2.php.

Trait1.php:

```
<?php
trait Trait1 {
    public function method1() {
        return "Ini adalah method dari Trait1";
    }
}
?>
```

Trait2.php:

```
<?php
trait Trait2 {
    public function method2() {
        return "Ini adalah method dari Trait2";
    }
}
?>
```

Sekarang, buat class Combination yang menggunakan kedua trait tersebut.

Combination.php:

```
<?php
require_once "Trait1.php";
require_once "Trait2.php";
class Combination {
   use Trait1, Trait2;
}
?>
```

Lakukan testing pada index.php:

```
<?php
require_once "Combination.php";
$obj = new Combination();
echo $obj->method1(); // Harusnya menampilkan "Ini adalah method dari Trait1"
echo "<br>";
echo $obj->method2(); // Harusnya menampilkan "Ini adalah method dari Trait2"
?>
```

4. Tugas

1. Buatlah kelas Mobil dengan metode berjalan dan berhenti. Kemudian buat *subclass* MobilSport yang melakukan *overriding* metode berjalan.
2. Dalam *subclass* MobilSport, gunakan *keyword parent* untuk memanggil metode berjalan dari superclass Mobil.
3. Ciptakan kelas lainnya dan coba implementasikan konsep *single inheritance*. Diskusikan mengapa Anda memilih struktur pewarisan tersebut.
4. Buatlah dua trait, dan ciptakan sebuah kelas yang menggunakan kedua trait tersebut untuk mensimulasikan *multiple inheritance*. Jelaskan fungsi dari setiap trait yang Anda buat.

Modul 4 Modifier dalam PHP

1. Tujuan

a. Memahami konsep modifier dalam pemrograman berorientasi objek PHP.
b. Mengerti perbedaan dan penggunaan dari modifier public, private, dan protected.
c. Mengetahui tujuan dan penerapan dari keyword final dan static.

2. Dasar Teori

A. Public, Private, dan Protected

Modifier adalah keyword yang menentukan tingkat akses dari properti dan metode dalam class.

1) **Public**: Properti atau metode yang didefinisikan sebagai public dapat diakses dari mana saja, termasuk di luar class.
2) **Private**: Properti atau metode yang didefinisikan sebagai private hanya dapat diakses dari dalam class itu sendiri.
3) **Protected**: Properti atau metode yang didefinisikan sebagai protected dapat diakses dari dalam class itu sendiri dan subclassnya.

B. Tujuan Penggunaan Modifier

Modifier digunakan untuk:

1) Mengendalikan akses ke properti dan metode dalam class.
2) Menyembunyikan detail implementasi dari pengguna class (encapsulation).
3) Membantu dalam pembuatan class yang lebih aman dan mudah dipelihara.

C. Penggunaan final dan static keyword

1) ***final***: *Keyword* ini digunakan sebelum deklarasi class atau metode untuk menunjukkan bahwa class atau metode tersebut tidak dapat diwariskan atau di-override oleh subclass.
2) ***static***: *Keyword* ini digunakan untuk mendefinisikan properti atau metode yang terkait dengan class, bukan instance dari class. Properti atau metode static dapat diakses tanpa perlu membuat instance dari class.

3. Kegiatan Praktikum

5) Eksplorasi Modifier Public, Private, dan Protected

Buat file dengan nama ***Person.php.***

```
<?php
class Person {
  public $name;
  private $age;
  protected $address;
```

```
    public function setAge($value) {
        $this->age = $value;
    }
    protected function getAddress() {
        return $this->address;
    }
    private function getSecret() {
        return "This is a secret!";
    }
}
?>
```

Lakukan testing pada ***testModifier.php.***

```
<?php
require_once "Person.php";
$person = new Person();
$person->name = "John"; // Bisa diakses karena public
$person->setAge(25);   // Bisa diakses karena public

// $person->age akan menghasilkan error karena private
// $person->getAddress() akan menghasilkan error karena protected
// $person->getSecret() akan menghasilkan error karena private

echo $person->name;    // Output: John
?>
```

6) Eksplorasi final dan static Keyword

Buat file dengan nama Vehicle.php.

```
<?php
class Vehicle {
    public static $count = 0;
    public function __construct() {
        self::$count++;
```

```
    }
    final public function startEngine() {
        return "Engine started!";
    }
}
class Car extends Vehicle {
    // public function startEngine() akan menghasilkan error karena metode tersebut final di parent class
}
?>
```

Lakukan testing pada ***testKeywords.php.***

```
<?php
require_once "Vehicle.php";
$car1 = new Car();
$car2 = new Car();
echo Vehicle::$count;   // Output: 2
?>
```

4. Tugas

1. Buatlah class BankAccount dengan properti accountNumber (public), balance (private), dan accountHolderName (protected). Lengkapi dengan metode untuk set dan get untuk setiap properti.
2. Eksplorasi pembuatan metode static yang bernama interestRate() dalam class BankAccount yang mengembalikan suku bunga tetap, misalnya 2.5%.
3. Ciptakan subclass dari BankAccount yang bernama PremiumAccount. Coba override metode interestRate() untuk mengembalikan suku bunga yang lebih tinggi, misalnya 3.5%. Jika Anda mengalami kesulitan, pikirkan apakah Anda harus menggunakan keyword final di superclass.
4. Buatlah class lain dengan penggunaan modifier yang berbeda, serta coba implementasikan keyword static dan final pada class tersebut. Jelaskan pilihan Anda.

Modul 5 Enkapsulasi

1. Tujuan

a. Memahami konsep enkapsulasi dalam pemrograman berorientasi objek.
b. Mengetahui cara kerja dan penggunaan modifier akses: public, private, dan protected.
c. Mengerti dan dapat menerapkan metode getter dan setter untuk memanage akses ke properti dalam sebuah class.

2. Dasar Teori

A. Modifier Akses: public, private, dan protected

Enkapsulasi memungkinkan objek untuk menyembunyikan data internalnya dari dunia luar dan hanya mengekspos data yang aman untuk publik. Modifier akses membantu dalam mewujudkan prinsip ini:

Public: Anggota class yang dideklarasikan sebagai public dapat diakses dari mana saja, termasuk di luar class.

Private: Anggota class yang dideklarasikan sebagai private hanya dapat diakses dari dalam class itu sendiri.

Protected: Anggota class yang dideklarasikan sebagai protected hanya dapat diakses dari dalam class itu sendiri dan oleh subclassnya.

B. Getter dan Setter

Getter dan setter adalah metode khusus yang digunakan untuk memperoleh dan mengatur nilai dari properti private dan protected. Ini memungkinkan kita untuk mengontrol akses dan modifikasi ke properti tersebut, dan juga memungkinkan validasi tambahan atau tindakan lain saat mengambil atau menetapkan nilai.

3. Kegiatan Praktikum

1. Eksplorasi Modifier Akses

Buat file dengan nama Mahasiswa.php.

php

Copy code

<?php

```
class Mahasiswa {
    private $NIM;
    protected $nama;
    public $prodi;

    public function setNIM($NIM) {
        $this->NIM = $NIM;
    }

    public function getNIM() {
        return $this->NIM;
    }

    protected function setNama($nama) {
        $this->nama = $nama;
    }

    public function getNama() {
        return $this->nama;
    }
}

?>
```

Lakukan testing pada testEnkapsulasi.php.

```
php
Copy code
<?php

require_once "Mahasiswa.php";

$student = new Mahasiswa();
```

```
$student->setNIM("12345678");
echo $student->getNIM();    // Output: 12345678

$student->prodi = "Teknik Informatika";
echo $student->prodi;      // Output: Teknik Informatika

// $student->setNama("Budi") akan menghasilkan error karena metode tersebut protected
// $student->NIM akan menghasilkan error karena properti tersebut private

?>
```

2. Implementasi Getter dan Setter

Pada class Mahasiswa yang telah Anda buat, Anda telah menggunakan getter dan setter untuk properti NIM. Sekarang, coba tambahkan validasi di setter untuk memastikan bahwa NIM harus terdiri dari 8 karakter.

4. Tugas

1. Buatlah class Buku dengan properti judul, pengarang, dan tahunTerbit. Gunakan modifier akses yang sesuai.
2. Lengkapi class Buku dengan metode getter dan setter untuk setiap properti. Pastikan pada setter tahunTerbit, Anda menambahkan validasi untuk memastikan tahun tidak lebih dari tahun saat ini.
3. Ciptakan beberapa instance dari class Buku dan cobalah mengakses dan memodifikasi propertinya menggunakan metode getter dan setter.
4. Refleksikan apa yang Anda pelajari dari tugas ini, terutama mengenai pentingnya enkapsulasi dan penggunaan getter dan setter dalam pemrograman berorientasi objek.

Modul 6 Polimorfisme

1. Tujuan

a. Memahami dan menerapkan konsep polimorfisme dalam pemrograman berorientasi objek.
b. Mengidentifikasi berbagai tipe polimorfisme dan bagaimana mereka diimplementasikan dalam PHP.
c. Mengimplementasikan polimorfisme dalam berbagai skenario praktikum.

2. Dasar Teori

A. Pengertian Polimorfisme

Polimorfisme berasal dari dua kata dalam bahasa Yunani, "poly" yang berarti banyak dan "morph" yang berarti bentuk. Dalam konteks pemrograman berorientasi objek, polimorfisme adalah kemampuan objek untuk diakses dalam bentuk-bentuk atau tipe-tipe berbeda.

B. Tipe-tipe Polimorfisme

1) ***Ad-hoc Polimorfisme***: Dikenal juga sebagai polimorfisme fungsi atau method overloading. Terjadi ketika dua fungsi memiliki nama yang sama tetapi parameter yang berbeda.
2) ***Sub-tipe Polimorfisme***: Objek dari subclass dapat dianggap sebagai objek dari superclass. Ini paling umum ditemukan dalam konsep pewarisan.
3) ***Parametric Polimorfisme***: Dikenal juga dengan generics atau templates di beberapa bahasa pemrograman. Mengizinkan kode untuk ditulis tanpa menentukan tipe data tertentu.

C. Contoh Polimorfisme di PHP

Dalam PHP, polimorfisme paling sering diwujudkan melalui pewarisan dan interface. Meskipun PHP tidak mendukung method overloading secara langsung seperti dalam bahasa lain, namun dapat diakali dengan beberapa teknik.

3. Kegiatan Praktikum

1. Implementasi Sub-tipe Polimorfisme

   Buat file dengan nama ***Hewan.php.***

```
<?php
    class Hewan {
      public function suara() {
        return "Suara hewan";
      }
    }
```

```
    class Kucing extends Hewan {
        public function suara() {
            return "Meong";
        }
    }
    class Anjing extends Hewan {
        public function suara() {
            return "Guk";
        }
    }
    function suaraHewan(Hewan $hewan) {
        return $hewan->suara();
    }
?>
```

Lakukan testing pada ***testPolimorfisme.php.***

```
<?php
    require_once "Hewan.php";
    $cat = new Kucing();
    $dog = new Anjing();

    echo suaraHewan($cat);    // Output: Meong
    echo suaraHewan($dog);    // Output: Guk
?>
```

Fungsi suaraHewan menerima parameter tipe Hewan tapi bisa menerima objek Kucing dan Anjing karena keduanya merupakan sub-tipe dari Hewan.

4. Tugas

1. Buatlah class BangunDatar dengan metode luas dan keliling. Kemudian buat subclass seperti Segitiga dan Persegi yang mengoverride metode tersebut sesuai dengan perhitungan matematika masing-masing bangun datar.
2. Buat fungsi infoBangunDatar yang menerima objek tipe BangunDatar dan mencetak informasi tentang luas dan keliling bangun datar tersebut.

3. Ciptakan beberapa instance dari subclass yang Anda buat dan coba panggil fungsi infoBangunDatar untuk masing-masing instance tersebut.
4. Refleksikan apa yang Anda pelajari dari tugas ini tentang bagaimana polimorfisme memungkinkan fleksibilitas dan generalisasi dalam pemrograman berorientasi objek.

Modul 7 Abstraksi dalam PHP

1. Tujuan

a. Memahami dan menerapkan konsep abstraksi dalam pemrograman berorientasi objek.
b. Mengenal dan membedakan antara abstract class dan interface dalam PHP.
c. Mengimplementasikan abstract class dalam skenario praktikum.

2. Dasar Teori

A. Abstract Class

Sebuah abstract class adalah class yang tidak bisa diinstansiasi. Ini biasanya digunakan sebagai blueprint bagi class lain. Class abstrak dapat memiliki metode abstrak dan metode non-abstrak. Metode abstrak adalah metode yang dideklarasikan di class abstrak tetapi tidak memiliki implementasi. Setiap class yang turunan dari class abstrak harus menyediakan implementasi untuk semua metode abstrak yang ada.

B. Implementasi Abstract Class

Pada PHP, sebuah class dideklarasikan sebagai abstrak dengan keyword abstract. Jika class abstrak memiliki metode abstrak, maka subclass yang mengextend class tersebut harus mengimplementasikan semua metode abstrak tersebut.

C. Perbedaan antara Interface dan Abstract Class

1) ***Abstract Class***: Dapat memiliki metode dengan implementasi (metode non-abstrak) dan metode tanpa implementasi (metode abstrak). Sebuah class hanya bisa mengextend satu class abstrak.
2) ***Interface:*** Semua metode di dalam interface adalah abstrak dan tidak memiliki implementasi. Sebuah class bisa mengimplementasikan banyak interface.

3. Kegiatan Praktikum

1. Implementasi Abstract Class

Buat file dengan nama ***BangunDatar.php.***

```
<?php
abstract class BangunDatar {
   abstract public function luas();
   abstract public function keliling();
}
class Persegi extends BangunDatar {
   private $sisi;
   public function __construct($sisi) {
      $this->sisi = $sisi;
```

```
    }
    public function luas() {
        return $this->sisi * $this->sisi;
    }
    public function keliling() {
        return 4 * $this->sisi;
    }
}
?>
```

Lakukan testing pada testAbstraksi.php.

```
<?php
require_once "BangunDatar.php";
$persegi = new Persegi(5);
echo "Luas Persegi: " . $persegi->luas() . " cm^2\n";      // Output: 25 cm^2
echo "Keliling Persegi: " . $persegi->keliling() . " cm";  // Output: 20 cm
?>
```

4. Tugas

1. Tambahkan class Lingkaran yang merupakan subclass dari BangunDatar dengan mengoverride metode luas dan keliling sesuai dengan perhitungan matematika lingkaran.
2. Buatlah fungsi informasiBangunDatar yang menerima objek tipe BangunDatar dan mencetak informasi tentang luas dan keliling bangun datar tersebut.
3. Ciptakan instance dari Persegi dan Lingkaran, lalu gunakan fungsi informasiBangunDatar untuk menampilkan informasi dari kedua instance tersebut.
4. Refleksikan apa yang Anda pelajari dari tugas ini mengenai kegunaan dan penerapan abstraksi dalam pemrograman berorientasi objek.

Modul 8 Namespace dan Autoloading dalam PHP

1. Tujuan

a. Memahami dan menerapkan konsep namespace dalam pemrograman PHP.
b. Mengetahui cara mengorganisir kode dengan autoloading.
c. Mengaplikasikan standar PSR-4 dalam autoloading.

2. Dasar Teori

A. Menggunakan Namespace

Namespace dalam PHP digunakan untuk menyelenggarakan dan mengelompokkan serangkaian kelas, fungsi, dan konstanta. Hal ini penting terutama dalam aplikasi skala besar untuk menghindari konflik nama dan memastikan modularitas kode.

B. Mengorganisir Kode dengan Autoloading

Autoloading memungkinkan PHP untuk memuat kelas secara otomatis tanpa harus memanggil include atau require untuk setiap file kelas. Dengan autoloading, Anda hanya perlu mendefinisikan aturan bagaimana kelas harus dimuat, dan PHP akan menanganinya untuk Anda.

C. PSR-4 Autoloading Standard

PSR-4 adalah salah satu standar yang disetujui oleh PHP-FIG yang mendefinisikan aturan untuk autoloading kelas dari jalur file. Pada dasarnya, PSR-4 mengatakan bahwa namespace dalam kelas harus sesuai dengan struktur direktori, dan nama kelas harus sesuai dengan nama file.

3. Kegiatan Praktikum

1. Menggunakan ***Namespace*** dan ***Autoloading***

   Asumsikan Anda memiliki struktur direktori sebagai berikut:

   markdown

   - src/
     - App/
       - Controllers/
         - UserController.php
       - Models/
         - UserModel.php

   File ***UserController.php:***

   ```
   <?php
       namespace App\Controllers;
   ```

```
    class UserController {
        // kode UserController
    }
?>
```

File ***UserModel.php:***

```
<?php
namespace App\Models;
    class UserModel {
        // kode UserModel
    }
?>
```

Untuk mengimplementasikan autoloading sesuai dengan standar PSR-4, Anda bisa menggunakan ***spl_autoload_register***.

File ***index.php:***

```
<?php
    spl_autoload_register(function ($className) {
        $className = str_replace("\\", DIRECTORY_SEPARATOR, $className);
        require_once 'src/' . $className . '.php';
    });

    use App\Controllers\UserController;
    use App\Models\UserModel;

    $userController = new UserController();
    $userModel = new UserModel();
?>
```

4. Tugas

1. Buatlah struktur direktori untuk sebuah aplikasi e-commerce dengan minimum terdapat direktori: Controllers, Models, dan Views.
2. Dalam masing-masing direktori, buatlah minimal dua file kelas. Setiap kelas harus memiliki namespace yang sesuai dengan struktur direktorinya.
3. Implementasikan autoloading sesuai dengan standar PSR-4 pada file utama Anda (misalnya index.php).
4. Di dalam file utama, buatlah objek dari setiap kelas yang Anda buat dan panggil beberapa metodenya.
5. Refleksikan apa yang Anda pelajari dari tugas ini mengenai bagaimana namespace dan autoloading membantu mengorganisir kode Anda dengan lebih rapi dan modular.

Modul 9 Penanganan Kesalahan dengan Exception

1. Tujuan

a. Memahami dan menerapkan konsep Exception dalam pemrograman PHP untuk penanganan kesalahan.
b. Mengetahui cara membuat dan menggunakan Exception kustom.
c. Mengaplikasikan struktur try, catch, throw, dan finally dalam kode program.

2. Dasar Teori

A. Dasar-dasar Exception

Exception adalah mekanisme khusus yang digunakan dalam pemrograman berorientasi objek untuk menangani kesalahan atau kondisi luar biasa. Dengan Exception, Anda dapat "menangkap" kesalahan dan memutuskan apa yang harus dilakukan selanjutnya, daripada membiarkan program berhenti secara tiba-tiba.

B. Membuat Exception Custom

Sementara PHP menyediakan sejumlah built-in exceptions, Anda juga dapat mendefinisikan exception kustom Anda sendiri untuk menangani kasus yang spesifik dengan aplikasi Anda. Exception kustom memperluas kelas dasar Exception.

C. Try, Catch, Throw, dan Finally

1) ***try***: Kode yang berpotensi menyebabkan exception diletakkan di dalam blok try.
2) ***catch***: Blok catch menangkap exception yang dilemparkan dari blok try. Dapat memiliki beberapa blok catch untuk menangani berbagai jenis exception.
3) ***throw***: Digunakan untuk melempar exception secara manual.
4) ***finally***: Blok ini akan selalu dijalankan setelah blok try dan catch, terlepas dari apakah exception dilemparkan atau tidak.

3. Kegiatan Praktikum

1. Menggunakan Exception Dasar

php

Copy code

```
<?php

function pembagian($num1, $num2) {
  if ($num2 == 0) {
    throw new Exception("Pembagi tidak boleh nol.");
  }
```

```
    return $num1 / $num2;
}

try {
    echo pembagian(10, 0);
} catch (Exception $e) {
    echo "Terjadi kesalahan: " . $e->getMessage();
} finally {
    echo "\nOperasi selesai.";
}

?>
```

2. Membuat dan Menggunakan Exception Kustom

```
php
Copy code
<?php

class PembagianDenganNolException extends Exception { }

function pembagian($num1, $num2) {
    if ($num2 == 0) {
        throw new PembagianDenganNolException("Pembagi tidak boleh nol.");
    }
    return $num1 / $num2;
}

try {
    echo pembagian(10, 0);
} catch (PembagianDenganNolException $e) {
    echo "Terjadi kesalahan: " . $e->getMessage();
} finally {
```

```
    echo "\nOperasi selesai.";
}

?>
```

4. Tugas

Buatlah fungsi kuadratAkar yang mengambil satu parameter dan mengembalikan kuadrat akar dari parameter tersebut. Jika parameter yang diberikan kurang dari 0, lemparkan exception kustom yang Anda buat.

Tulis kode untuk memanggil fungsi kuadratAkar dalam blok try dan tangani exception dengan blok catch.

Buatlah 3 exception kustom lainnya untuk situasi yang berbeda dalam konteks aplikasi Anda.

Refleksikan pengalaman Anda: Bagaimana mekanisme exception membantu Anda dalam menangani kesalahan dan memastikan kelancaran eksekusi kode?

Modul 10 Advanced OOP Features

1. Tujuan

a. Memahami konsep Late Static Binding dalam PHP.
b. Mengenal dan menerapkan Anonymous Classes.
c. Memahami perbedaan antara objek dan referensinya.

2. Dasar Teori

A. Late Static Binding

Late Static Binding memungkinkan referensi ke class yang diturunkan diakses dengan kata kunci static. Ini digunakan untuk menentukan class mana yang digunakan saat metode statis dari subclass memanggil metode yang sudah didefinisikan di parent class.

B. Anonymous Classes

Sebagai tambahan dari PHP 7, Anonymous Classes berguna ketika class sederhana hanya digunakan satu kali saat eksekusi dan tidak perlu didefinisikan sebagai class formal. Mereka dapat digunakan untuk implementasi cepat dari interface atau instansiasi objek sementara.

C. Objek dan Referensi

Dalam PHP, ketika Anda menetapkan satu objek ke variabel lain, Anda sebenarnya membuat referensi ke objek asli, bukan membuat salinan dari objek tersebut. Ini berarti bahwa perubahan pada satu variabel akan mempengaruhi variabel lainnya yang merujuk ke objek yang sama.

3. Kegiatan Praktikum

a. Late Static Binding

```
class BaseClass {
   public static function who() {
      echo __CLASS__;
   }
   public static function test() {
      static::who(); // Fungsi Late Static Binding disini
   }
}

class ChildClass extends BaseClass {
   public static function who() {
```

```
        echo 'Anak';
    }
}
ChildClass::test();  // Output: Anak
```

b. Anonymous Classes

```
interface Logger {
    public function log(string $msg);
}

$logger = new class implements Logger {
    public function log(string $msg) {
        echo $msg;
    }
};

$logger->log("Ini adalah pesan log dari Anonymous Class.");
```

c. Objek dan Referensi

```
class SimpleClass {
    public $data = "Original Data";
}

$instance = new SimpleClass();

$assignedReference   = $instance;
$assignedReference->data = "Data Baru";

echo $instance->data;  // Output: Data Baru
```

4. Tugas

1. Buatlah sebuah class Calculator dengan metode statis add, subtract, multiply, dan divide. Kemudian, turunkan class tersebut menjadi class ScientificCalculator dengan metode tambahan power. Gunakan konsep Late Static Binding untuk mengkalkulasi hasil dari fungsi power.
2. Implementasikan interface ImageProcessor dengan Anonymous Class yang memiliki metode resize, rotate, dan applyFilter. Anda tidak perlu menuliskan fungsi sebenarnya, cukup tuliskan pesan yang mengindikasikan metode apa yang sedang dijalankan.
3. Buatlah sebuah class Product dengan properti name dan price. Instansiasi objek dari class tersebut dan buatlah referensi ke objek yang sama. Ubah nilai dari salah satu objek dan verifikasi apakah objek lainnya terpengaruh.

Modul 11 Operasi Database CRUD

1. Tujuan

a. Mengaplikasikan konsep PBO dalam operasi database CRUD (Create, Read, Update, Delete).
b. Memahami dan menerapkan PDO untuk koneksi dan operasi database.
c. Menerapkan operasi CRUD dalam kode berbasis objek.

2. Dasar Teori

A. Konsep Dasar PBO untuk CRUD

Dalam pemrograman berorientasi objek (PBO), operasi database dapat dimodelkan sebagai kelas yang mewakili tabel-tabel database. Metode-metode dalam kelas ini dapat mewakili operasi CRUD. Misalnya, sebuah kelas 'User' bisa memiliki metode create, read, update, dan delete untuk mengelola data di tabel user.

B. Setting Koneksi Database dengan PDO

PHP Data Objects (PDO) adalah antarmuka database yang memberikan kumpulan fungsi untuk mengakses database dalam PHP. Menggunakan PDO memungkinkan pemrogram untuk membuat kode yang dapat berfungsi dengan banyak jenis database dengan sedikit perubahan.

Contoh koneksi dengan PDO:

```
$dsn = "mysql:host=localhost;dbname=test_db";
$pdo = new PDO($dsn, 'username', 'password');
```

C. Operasi Create dalam PBO

Dalam PBO, operasi Create dapat direpresentasikan sebagai metode dalam kelas. Metode ini dapat menerima parameter (seperti data yang akan ditambahkan) dan menggunakan PDO untuk menambahkannya ke database.

D. Operasi Read dalam PBO

Operasi Read mengambil data dari database. Metode ini mungkin mengembalikan satu baris, beberapa baris, atau semua baris dari tabel, tergantung pada kebutuhannya.

E. Operasi Update dalam PBO

Operasi Update memodifikasi baris yang ada di database. Metode ini biasanya memerlukan identifikasi baris yang akan diubah dan data baru yang akan ditempatkan di baris tersebut.

F. Operasi Delete dalam PBO

Operasi Delete menghapus baris dari tabel. Sama seperti operasi Update, operasi ini memerlukan identifikasi baris yang akan dihapus.

3. Kegiatan Praktikum

1. Membuat ***Koneksi Database***

```
<?php
try {
        $dsn = "mysql:host=localhost;dbname=test_db";
        $username = "root";
        $password = "password";

        $conn = new PDO($dsn, $username, $password);
        $conn->setAttribute(PDO::ATTR_ERRMODE, PDO::ERRMODE_EXCEPTION);
        echo "Connected successfully";
} catch(PDOException $e) {
          echo "Connection failed: " . $e->getMessage();
}
?>
```

2. Implementasi ***Operasi Create***

```
class User {
  private $conn;
  public function __construct($conn) {
    $this->conn = $conn;
  }
  public function create($name, $email) {
    try {
      $stmt = $this->conn->prepare("INSERT INTO users (name, email) VALUES (:name,
 :email)");
      $stmt->bindParam(':name', $name);
      $stmt->bindParam(':email', $email);
      $stmt->execute();
      echo "User added successfully!";
    } catch(PDOException $e) {
      echo "Error: " . $e->getMessage();
    }
  }
```

```
}
$user = new User($conn);
$user->create("John Doe", "john@example.com");
```

3. Implementasi ***Operasi Read***

```
public function read($id = null) {
    try {
        if ($id) {
            $stmt = $this->conn->prepare("SELECT * FROM users WHERE id = :id");
            $stmt->bindParam(':id', $id);
        } else {
            $stmt = $this->conn->prepare("SELECT * FROM users");
        }
        $stmt->execute();
        $results = $stmt->fetchAll(PDO::FETCH_ASSOC);
        return $results;
    } catch(PDOException $e) {
        echo "Error: " . $e->getMessage();
    }
}
// Example usage:
print_r($user->read());
```

4. Implementasi ***Operasi Update***

```
public function update($id, $name, $email) {
    try {
        $stmt = $this->conn->prepare("UPDATE users SET name = :name, email = :email
        WHERE id = :id");
        $stmt->bindParam(':id', $id);
        $stmt->bindParam(':name', $name);
        $stmt->bindParam(':email', $email);
        $stmt->execute();
        echo "User updated successfully!";
    } catch(PDOException $e) {
```

```
            echo "Error: " . $e->getMessage();
        }
    }
    // Example usage:
    $user->update(1, "Jane Doe", "jane@example.com");
```

5. Implementasi ***Operasi Delete***

```
    public function delete($id) {
        try {
            $stmt = $this->conn->prepare("DELETE FROM users WHERE id = :id");
            $stmt->bindParam(':id', $id);
            $stmt->execute();
            echo "User deleted successfully!";
        } catch(PDOException $e) {
            echo "Error: " . $e->getMessage();
        }
    }
    // Example usage:
    $user->delete(1);
```

Catatan: Sebelum menjalankan skrip ini, pastikan Anda memiliki tabel users dengan kolom id, name, dan email di database test_db. Jangan lupa untuk mengganti kredensial koneksi sesuai dengan setup Anda.

4. Tugas

1. Optimalisasi kode Anda dengan menambahkan fitur validasi input untuk setiap operasi CRUD.
2. Ekspansi kelas Anda dengan menambahkan metode lain yang mungkin relevan, seperti search untuk mencari user berdasarkan kriteria tertentu.
3. Buat kelas baru, misalnya Post, untuk mengelola post atau artikel yang dibuat oleh user. Implementasikan operasi CRUD di kelas ini.
4. Buat sistem sederhana di mana user dapat mendaftar, melihat profil mereka, mengedit profil, atau menghapus akun mereka, menggunakan operasi CRUD yang Anda telah buat

1. **Tujuan**
   a. Menerapkan semua konsep-konsep OOP yang telah dipelajari sepanjang praktikum.
   b. Mengembangkan sebuah aplikasi sederhana berbasis web yang interaktif, menggunakan PHP dengan pendekatan OOP.

2. **Petunjuk**
   A. Analisis Kebutuhan: Sebelum memulai pemrograman, buatlah dokumen analisis kebutuhan yang menjelaskan fitur-fitur apa saja yang ingin Anda implementasikan.
   B. Desain Database: Desain skema database yang akan mendukung fungsionalitas aplikasi Anda. Tentukan tabel-tabel yang dibutuhkan, kolom-kolom di dalamnya, dan relasi antar tabel.
   C. Pemrograman: Gunakan prinsip-prinsip PBO yang telah Anda pelajari, seperti enkapsulasi, pewarisan, polimorfisme, dan abstraksi. Pastikan untuk menggunakan fitur-fitur lanjutan seperti Namespace, Autoloading, dan Exception Handling.
   D. Testing: Sebelum menyerahkan proyek, lakukan pengujian untuk memastikan semua fitur berfungsi dengan benar. Perbaiki bug atau masalah yang mungkin muncul selama pengujian.
   E. Dokumentasi: Selain kode program, buatlah dokumentasi yang menjelaskan tentang aplikasi Anda, cara penggunaannya, serta struktur database dan kode.

3. **Tugas**
   1) Aplikasi: Kembangkan aplikasi web dengan menggunakan PHP dan database sesuai dengan tema yang Anda pilih.
   2) Laporan: Buat laporan yang berisi:
   - Deskripsi Aplikasi: Jelaskan secara singkat tentang aplikasi yang Anda buat.
   - Fitur-fitur Aplikasi: Daftar dan jelaskan fitur-fitur yang ada di aplikasi Anda.
   - Diagram Relasi Database: Sertakan diagram yang menunjukkan struktur dan relasi database Anda.
   - Kode Penting: Sertakan beberapa potongan kode yang menurut Anda penting atau merepresentasikan penggunaan prinsip PBO.
   - Kesimpulan: Tuliskan apa yang Anda pelajari selama pengerjaan proyek akhir ini dan bagaimana pengalaman Anda menerapkan konsep PBO dalam aplikasi nyata.

   3) Presentasi: Anda mungkin akan diminta untuk mempresentasikan aplikasi Anda di depan kelas atau tim pengajar. Siapkan slide presentasi yang menarik dan pastikan untuk menunjukkan demo dari aplikasi Anda.
   4) Evaluasi: Aplikasi Anda akan dievaluasi berdasarkan fungsionalitas, penerapan konsep PBO, kualitas kode, dan kelengkapan dokumentasi.