Laporan Praktikum Pemrograman

Modul IV –PENGENALAN AWAL PEMROGRAMAN BERORIENTASI OBJEK

(VISIBILITY, SETTERS, GETTERS, STATIC KEYWORD, DAN CONSTANT)

Muhammad Yusron Nashrulloh/22103001005

Dosen : Achmad Arif Munaji, ST., M.Kom

Tanggal praktikum: 17 Mei 2023

myusron434@gmail.com

Teknik Komputer

Institut Teknologi dan Sains Nahdlatul Ulama Kalimantan

***Abstrak*—Pada modul praktikum keempat ini, akan dibahas mengenai konsep dasar dalam pemrograman berorientasi objek (OOP) pada bahasa pemrograman PHP. Beberapa konsep yang akan dibahas meliputi visibility atau tingkat akses pada properti dan metode dalam sebuah kelas, setters dan getters untuk mengatur dan mengambil nilai properti, static keyword untuk membuat properti atau metode yang terikat pada kelas, serta constant untuk membuat nilai tetap yang tidak dapat diubah selama runtime.**

**Dalam pemrograman berorientasi objek, penting untuk memahami konsep-konsep dasar tersebut agar dapat membuat kode yang lebih efisien dan mudah di-maintain. Dalam modul praktikum ini, akan disajikan penjelasan lengkap beserta contoh implementasi dari masing-masing konsep, sehingga diharapkan peserta dapat memahami dengan baik dan dapat mengimplementasikan dalam kode yang dibuat.**

***Kata kunci—Visibility, Setter & Getters, Static Keyword, dan Constant***

1. Pendahuluan

Pemrograman berorientasi objek (OOP) merupakan salah satu paradigma pemrograman yang sering digunakan untuk memudahkan pengembangan perangkat lunak. Dalam OOP, segala sesuatu dianggap sebagai objek yang memiliki karakteristik atau atribut serta dapat melakukan suatu tindakan atau fungsi. Bahasa pemrograman PHP sendiri sudah mendukung OOP sejak versi 5, sehingga programmer dapat menggunakan konsep OOP pada pembuatan aplikasi atau sistem dengan bahasa PHP. Konsep OOP pada bahasa PHP meliputi visibility, setters, getters, static keyword, dan constant.

Visibility merupakan salah satu konsep yang digunakan untuk mengatur aksesibilitas suatu property atau method pada suatu class. Terdapat 3 jenis visibility, yaitu public, private, dan protected. Setters dan getters adalah metode yang digunakan untuk mengakses property pada suatu class. Setters digunakan untuk mengubah nilai property sedangkan getters digunakan untuk mengambil nilai property. Static keyword digunakan untuk membuat property atau method yang dapat diakses tanpa harus membuat instance dari class tersebut. Sedangkan constant adalah nilai yang tetap dan tidak dapat diubah pada suatu class.

Dalam modul ini, akan membahas pengenalan awal tentang konsep-konsep dasar pada OOP pada bahasa pemrograman PHP. Khususnya mengenai visibility, setters, getters, static keyword, dan constant. Selain itu, kita juga akan mempelajari bagaimana cara mengimplementasikan konsep-konsep tersebut pada suatu program PHP secara praktis.

1. Tinjauan Pustaka
   1. Visibility

Dalam bahasa pemrograman PHP, visibility merujuk pada pengaturan aksesibilitas atau hak akses pada properti dan metode dalam sebuah class. Ada tiga tingkat visibility yang dapat diterapkan pada properti dan metode, yaitu:

Public: properti dan metode yang dideklarasikan sebagai public dapat diakses dari mana saja, baik dari dalam maupun luar class. Protected: properti dan metode yang dideklarasikan sebagai protected hanya dapat diakses oleh class itu sendiri atau oleh class turunan yang diturunkan dari class induknya. Private: properti dan metode yang dideklarasikan sebagai private hanya dapat diakses oleh class itu sendiri dan tidak dapat diakses dari luar class.

* 1. Setters & Getters

Setter dan getter adalah metode dalam bahasa pemrograman PHP untuk mengakses dan mengubah nilai dari properti yang bersifat private atau protected pada suatu class. Dengan menggunakan setter dan getter, Anda dapat membatasi akses langsung pada properti yang sensitif, sambil tetap memberikan kemudahan untuk mengubah dan membaca nilai dari properti tersebut.

Setter dan getter pada PHP biasanya didefinisikan dengan mengikuti konvensi penamaan yang sudah ditetapkan. Nama setter biasanya diawali dengan kata "set" dan diikuti dengan nama properti yang ingin diubah. Sedangkan, nama getter diawali dengan kata "get" dan diikuti dengan nama properti yang ingin dibaca.

* 1. Static Keyword & Constant

Static keyword dan constant adalah fitur-fitur dalam bahasa pemrograman PHP yang digunakan untuk membuat properti dan metode yang bersifat konstan dan tetap dalam nilai atau perubahan. Keduanya memungkinkan Anda untuk memperoleh nilai atau menggunakan metode tanpa harus membuat objek dari class yang bersangkutan.

Static keyword digunakan untuk membuat properti atau metode yang bersifat konstan pada class. Properti yang dideklarasikan sebagai static dapat diakses dari mana saja, tanpa harus membuat objek dari class tersebut. Demikian juga, metode yang dideklarasikan sebagai static dapat diakses tanpa harus membuat objek, sehingga memudahkan dalam penggunaan fungsionalitas tertentu tanpa perlu membuat instance class terlebih dahulu.

1. Langkah Praktikum

Urutan pelaksanaan pada praktikum ini yaitu diawali dengan menjawab soal pretest yaitu abstract dan interface dapat digunakan pada apa? abstract dan interface dapat digunakan pada Pemrograman berorientasi objek atau oop.

Lalu setelah soal pretest selesai dijawab, kemudian membuka aplikasi xampp control panel, lalu menyalakan di bagian Apache sampai warna hijau, setelah itu membuka browser dengan mengetikkan local host dan membuat folder pada file htdocs xampp lalu membukanya dengan aplikasi Visual Stuido Code. Kemudian mulai mengerjakan soal praktikum yaitu sebagai berikut :

<?php  
  
class Car {  
  public $name;  
  public function \_\_construct($name) {  
    $this->name = $name;  
  }  
 public function intro(){

  return "Choose Car quality! I'm an $this->name!";

}   
}  
class Audi extends Car {  
  public function intro(){  
    return "Choose German quality! I'm an $this->name!";   
  }  
}  
class Volvo extends Car {  
  public function intro(){  
    return "Proud to be Swedish! I'm a $this->name!";   
  }  
}  
  
class Citroen extends Car {  
  public function intro(){  
    return "French extravagance! I'm a $this->name!";   
  }  
}  
?>

1. Terapkan konsep abstract pada *class Car* diatas
2. Tetapkan *method* mana yang baiknya dijadikan *abstract* dan terapkan konsep *abstract* pada *method* tersebut.
3. Tambahkan 1 argument dengan nama *$price* yang nanti diisikan informasi mengenai harga mobil pada method abstract. Kemudian tambahkan " and my price is $this->price"; diakhir string pada return tiap method abstract turunan agar merubah hasil outputnya.
4. Buat file baru bernama interface\_car.php
5. Copy semua isi dari class Car yang sudah dibuat sebelumnya
6. Ubah dengan sedemikian rupa class Car agar bisa menerapkan konsep dari interface yang baik dan benar.
7. hasil dan analisis

Berikut merupakan codingan untuk soal tersebut dan penjelasannya :

<?php

abstract class Car {

    public $name;

    public $price;

    public function \_\_construct($name, $price) {

        $this->name = $name;

        $this->price = $price;

    }

    abstract public function intro();

}

class Audi extends Car {

    public function intro() {

        return "Choose German quality! I'm an $this->name and my price is $this->price!";

    }

}

class Volvo extends Car {

    public function intro() {

        return "Proud to be Swedish! I'm a $this->name and my price is $this->price!";

    }

}

class Citroen extends Car {

    public function intro() {

        return "French extravagance! I'm a $this->name and my price is $this->price!";

    }

}

$audi = new Audi("Audi A4", 50000);

$volvo = new Volvo("Volvo XC60", 40000);

$citroen = new Citroen("Citroen C4", 30000);

echo $audi->intro();     // Output: Choose German quality! I'm an Audi A4 and my price is 50000!

echo $volvo->intro();    // Output: Proud to be Swedish! I'm a Volvo XC60 and my price is 40000!

echo $citroen->intro();  // Output: French extravagance! I'm a Citroen C4 and my price is 30000!

?>

Analisis :

1. Class Car diperbarui menjadi abstract dengan kata kunci abstract sebelum kata kunci class. Class abstract tidak dapat diinstansiasi langsung, tetapi dapat diwarisi oleh class turunan.
2. Method intro() pada class Car dideklarasikan sebagai abstract dengan kata kunci abstract sebelum deklarasi method. Method abstract tidak memiliki implementasi di class abstrak, tetapi harus diimplementasikan di class turunan.
3. Class Audi, Volvo, dan Citroen merupakan class turunan dari class abstrak Car. Masing-masing class turunan harus mengimplementasikan method abstract intro() yang didefinisikan di class abstrak Car.
4. Pada class turunan (Audi, Volvo, Citroen), method intro() ditambahkan dengan ekstra string "and my price is $this->price!" pada return statement-nya. Ini akan mengubah hasil output agar mencakup informasi harga mobil.
5. Untuk menginstansiasi objek dari class turunan, kita dapat menggunakan sintaks new ClassName(...) seperti pada contoh di atas. Setiap objek mewarisi properti dan method dari class abstrak, tetapi hasil output method intro() akan disesuaikan dengan implementasi di class turunan.

Dengan menggunakan konsep abstrak, kita dapat membuat blueprint class Car yang menetapkan struktur dan method yang harus ada dalam class turunan, sementara setiap class turunan dapat mengimplementasikan method abstract sesuai dengan kebutuhan spesifiknya.

1. Kesimpulan

Pemrograman berorientasi objek (OOP) pada bahasa pemrograman PHP melibatkan beberapa konsep dasar seperti visibility, setters, getters, static keyword, dan constant. Konsep-konsep tersebut penting dalam OOP karena dapat membantu dalam pengaturan nilai properti, aksesibilitas metode, serta pengelolaan nilai tetap yang tidak dapat diubah.

VI. Daftar Pustaka

**[1]**[***https://www.php.net/manual/en/language.oop5.visibility.php***](https://www.php.net/manual/en/language.oop5.visibility.php)

diakses pada tanggal 20 Mei 2023

**{2]** [***https://www.php.net/manual/en/language.oop5.visibility.php***](https://www.php.net/manual/en/language.oop5.visibility.php)diakses pada tanggal 20 Mei 2023