Laporan Praktikum Pemrograman

Modul IV –PENGENALAN AWAL PEMROGRAMAN BERORIENTASI OBJEK

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

Raita Rahmatina/22103001006

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

Tanggal praktikum: 17 Mei 2023

raita221201@gmail.com

Teknik Komputer

Institut Teknologi dan Sains Nahdlatul Ulama Kalimantan

***Abstrak*— Pada praktikum keempat ini, membahas konsep dasar pemrograman berorientasi objek (OOP) dalam Bahasa PHP. Konsep yang dibahas mencakup tingkat visibilitas atau akses untuk properti dan metode kelas, *setters* dan *getters* untuk mengatur dan mengambil nilai properti, kata kunci statis untuk membuat properti atau metode terikat kelas, dan konstanta untuk membuat nilai tetap yang tidak dapat diubah saat *runtime*. Dalam pemrograman berorientasi objek, penting untuk memahami konsep dasar ini untuk membuat kode yang lebih efisien dan dapat dipelihara. Praktikum kali ini memberikan penjelasan lengkap dan contoh cara menerapkan setiap konsep, sehingga diharapkan mahasiswa memiliki pemahaman yang baik dan dapat menerapkan dalam kode yang dihasilkan.**

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

1. Pendahuluan

*Object oriented programming* atau OOP adalah konsep pemrograman yang berorientasi pada objek atau data untuk mengatur sebuah desain program. Secara garis besar, cara kerja OOP yaitu membangun program dari gabungan objek-objek yang saling berhubungan dan disusun menjadi satu kesatuan. 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* adalah teknik yang digunakan untuk mengatur *property* maupun *method* pada sebuah objek. Dengan menggunakan *Access Modifier* nantinya akan memberi batasan akses yang di inginkan terhadap suatu objek tertentu. *Visibility* juga 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*.

Pada praktikum kali ini akan mempraktikan mengenai 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 menerapkan konsep-konsep tersebut pada bahasa PHP secara praktis.

1. Tinjauan Pustaka
   1. Visibility

*Visibility* adalah teknik yang digunakan untuk mengatur property maupun *method* pada sebuah objek. Dengan menggunakan *Access Modifier* nantinya akan memberi batasan akses yang di inginkan terhadap suatu objek tertentu. *Visibility* juga merupakan salah satu konsep yang digunakan untuk mengatur aksesibilitas suatu *property* atau *method* pada suatu 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* dan *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*, pengguna 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* dan *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 pengguna 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

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.

Kemudian, setelah soal *pretest* terjawab, mahasiswa membuka aplikasi *xampp control panel*, menyalakan di bagian *Apache* sampai warna hijau, membuka *browser* dengan mengetikkan *local host* dan membuat folder pada file *htdocs xampp* lalu membukanya dengan aplikasi Visual Stuido Code. Terakhir, menjawab soal praktikum :

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 *coding* untuk soal di atas beserta 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*". Kelas abstrak tidak dapat diinstansiasi secara langsung, tetapi dapat diwariskan oleh kelas turunan.
2. Metode intro() dari kelas Mobil dideklarasikan abstrak sebelum definisi metode menggunakan kata kunci abstrak. Metode abstrak tidak memiliki implementasi di kelas abstrak, tetapi harus diimplementasikan di kelas turunan.
3. Kelas Audi, Volvo dan Citroen adalah kelas turunan dari kelas Mobil abstrak. Setiap kelas turunan harus mengimplementasikan metode abstract intro() yang didefinisikan di kelas abstrak Mobil.
4. Di kelas turunan (Audi, Volvo, Citroen) metode intro() ditambahkan dengan string tambahan "and my price is $this->price!". dalam slip pengembalian. Ini akan mengubah output untuk menyertakan harga informasi mobil.
5. Jika kita ingin membuat objek dari kelas turunan, kita dapat menggunakan sintaks *ClassName* (...) baru seperti pada contoh di atas. Setiap objek mewarisi properti dan metode dari kelas abstrak, tetapi output dari metode intro() dikustomisasi sesuai dengan penerapan kelas turunan. Konsep abstrak dapat digunakan untuk membuat garis besar untuk kelas otomatis yang mendefinisikan struktur dan metode yang harus dimiliki oleh kelas turunan, sementara setiap kelas turunan dapat mengimplementasikan metode abstrak sesuai dengan kebutuhan spesifiknya.
6. 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.

Praktikum selanjutnya diharapkan mahasaiswa dapat menyelesaikan penugasan secara lebih sederhana dan efektif dalam melaksanakan sistem pembelajaran yang ada.

VI. Daftar Pustaka

dewaweb.com

amuofficialnet.com

elib.unikom.ac.id

kodingin.com

php.net