Laporan Praktikum Pemrograman

Modul II – Pengenalan Awal Pemograman Berorientasi Objek (Constructor, Object Type, Inheritance, dan Overriding)

Raita Rahmatina/22103001006

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

Tanggal praktikum: 27 Maret 2023

Raita221201@gmail.com

Teknik Komputer

Institut Teknologi dan Sains Nahdlatul Ulama Kalimantan

***Abstrak*—Pada praktikum kedua, dibahas konsep dasar pemrograman berorientasi objek dalam bahasa pemrograman PHP. Praktikum ini bertujuan untuk memperkenalkan konsep dasar pemrograman berorientasi objek termasuk konstruktor, tipe objek, pewarisan, dan penggantian dalam PHP. Dalam praktikum, mahasiswa akan mempelajari cara membuat konstruktor untuk kelas, menginisialisasi objek dengan konstruktor, dan menerapkan pewarisan di kelas untuk membuat subkelas dengan properti dan perilaku yang sama dengan kelas induk. Selain itu, mahasiswa juga belajar meng-override method yang ada di parent class dengan method baru di subclass. Semua konsep ini dipraktikkan dengan membuat kelas dan objek serta melakukan pemrograman berorientasi objek dalam bahasa pemrograman PHP. Setelah menyelesaikan praktikum ini, mahasiswa diharapkan mampu memahami dan menerapkan konsep dasar pemrograman berorientasi objek pada PHP.**

***Kata kunci— Constructor,Obejct Type, Inheritance & Overriding***

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. Dalam OOP, konsep dasar yang harus dipahami meliputi objek, kelas, konstruktor, tipe objek, pewarisan (inheritance), dan penggantian (overriding). Konsep-konsep ini sangat penting untuk dipahami karena dapat membantu pengembang dalam membangun perangkat lunak yang efisien dan terstruktur dengan baik.

Pada praktikum kedua ini, melakukan praktik lebih mendalam mengenai konsep-konsep tersebut dan menggunakan bahasa pemrograman PHP. Pada awalnya, akan menerapkan konstruktor, yaitu metode khusus dalam kelas yang digunakan untuk menginisialisasi objek. Selanjutnya, menerapkan tipe objek, yang merujuk pada jenis data yang dapat diatur oleh sebuah objek. Kemudian, menerapkan pewarisan (inheritance), yang memungkinkan kelas baru untuk mengambil sifat dan metode dari kelas yang sudah ada. Dan terakhir, menerapkan penggantian (overriding), yang memungkinkan kelas turunan untuk mengganti metode yang sudah ada dalam kelas dasarnya.

1. Tinjauan Pustaka
   1. Constructor

Constructor adalah *method* atau *function* yang otomatis akan dijalankan saat class diinstansiasi (dibuat sebuah objek), pada bagian *constructor* dapat dilakukan apapun yang bisa dilakukan dalam *method* (*function*) kecuali mengembalikan nilai (*return* *value*). Constructor juga merupakan suatu function khusus yang akan  
dieksekusi ketika suatu objek dibuat (instansiasi class).  
Umumnya constructor dibuat untuk memberikan suatu operasi awal yang harus dilakukan ketika sebuah objek dilahirkan (inisialisasi objek).

* 1. *Object Type*

*Object* merupakan insansiasi dari suatu *Class*.  
Sebuah class tidak dapat digunakan tanpa  
diinstansiasikan dulu (kecuali untuk Static *Class*).  
*Object* sebenarnya sebuah variabel. *Object* juga merupakan output dari *class*. dan object dapat menampilkan atau mengelola isi *class*. seluruh isi *class* akan kita instansiasikan menjadi *object*.

* 1. *Inheritance* dan *Overriding*

*Inheritance* atau Pewarisan adalah konsep OOP dimana sebuah *class* dapat menurunkan *property* dan *method* yang dimilikinya kepada class lain. Konsep inheritance dipakai untuk memanfaatkan fitur code reuse atau kode yang yang dapat digunakan kembali, yakni menghindari terjadinya duplikasi kode program. ***Overriding*** adalah sebuah teknik PHP dengan konsep OOP yang mengimplementasikan pembuatan **method** yang sama pada *class child* dengan method *parent*nya. Terkadang ada kebutuhan untuk membuat fungsi yang sama namun keadaan yang berbeda. Maka dengan *overriding* menjadi solusi pemecahan masalah tersebut. Syarat untuk menggunakan teknik [*overriding*](http://kodingin.com/overriding-pada-php-oop) yaitu harus mempunyai *class parent* dan *class child.*

1. Langkah Praktikum

Pelaksanaan praktikum ini diawali dengan membuka aplikasi *xampp control panel*, 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 Studio *Code*. Kemudian mulai mengerjakan soal praktikum yaitu sebagai berikut :

1. Terapkan *constructor* pada *class Fruit*
2. Buat class baru bernama DetailFruit yang menerapkan konsep *Object Type class Fruit* pada *method* detail() dan kemudian *method* tersebut akan menampilkan output “I love $name and its $color”.
3. Buat *Class Apple* dan *Class Banana* yang merupakan *Child* dari *Class Fruit*. Masing-masing *Child Class* mempunyai *property* $shape dan *method* detail() untuk menampilkan output: “I Love $name, its $color, and its $shape”. Terapkan juga konsep *Overriding* untuk membuat baris program menjadi lebih rapi dan tidak diulang-ulang.
4. Buat *Class* Kalkulator yang memiliki tiga *Child Class*, yaitu Tambah, Kurang, dan Bagi. *Parent Class* mempunyai *property* $angka1 dan $angka2. Semua *Class* mempunyai *method* hasil(). *Method* hasil() pada *Parent Class* hanya menampilkan angka saja tanpa dilakukan proses aritmatika, sedangkan *Child Class* melakukan proses aritmatika sesuai dengan nama *Class*-nya.
5. hasil dan analisis

Perintah atau soal yang dikerjakan pada praktikum modul 2 ini adalah sebagai berikut :

Terapkan *constructor* pada *class Fruit*

Buat class baru bernama DetailFruit yang menerapkan konsep *Object Type class Fruit* pada *method* detail() dan kemudian *method* tersebut akan menampilkan output “I love $name and its $color”

Buat *Class Apple* dan *Class Banana* yang merupakan *Child* dari *Class Fruit*. Masing-masing *Child Class* mempunyai *property* $shape dan *method* detail() untuk menampilkan output: “I Love $name, its $color, and its $shape”. Terapkan juga konsep *Overriding* untuk membuat baris program menjadi lebih rapi dan tidak diulang-ulang.

Buat *Class* Kalkulator yang memiliki tiga *Child Class*, yaitu Tambah, Kurang, dan Bagi. *Parent Class* mempunyai *property* $angka1 dan $angka2. Semua *Class* mempunyai *method* hasil(). *Method* hasil() pada *Parent Class* hanya menampilkan angka saja tanpa dilakukan proses aritmatika, sedangkan *Child Class* melakukan proses aritmatika sesuai dengan nama *Class*-nya.

Penjelasannya sebagai berikut :

1. Menerapkan *Construct* pada *Class Fruit.*

?php

class Fruit {

    public $name;

    public $color;

    function \_\_construct($name, $color) {

      $this->name = $name;

      $this->color = $color;

    }

    function get\_name() {

      return $this->name;

    }

    function get\_color() {

      return $this->color;

    }

  }

  $apple = new Fruit("Apple", "Red");

  echo $apple->get\_name();

  echo "<br>";

  echo $apple->get\_color();

1. Buat class baru bernama DetailFruit yang menerapkan konsep *Object Type class Fruit* pada *method* detail() dan kemudian *method* tersebut akan menampilkan output “I love $name and its $color”.

class DetailFruit {

    public $fruit;

    function \_\_construct($fruit) {

      $this->fruit = $fruit;

    }

    function detail() {

      echo "I love " . $this->fruit->get\_name() . " and its " . $this->fruit->get\_color();

    }

  }

  $apple = new Fruit("Apple", "Red");

  $detail\_fruit = new DetailFruit($apple);

  $detail\_fruit->detail();

1. Buat *Class Apple* dan *Class Banana* yang merupakan *Child* dari *Class Fruit*. Masing-masing *Child Class* mempunyai *property* $shape dan *method* detail() untuk menampilkan output: “I Love $name, its $color, and its $shape”. Terapkan juga konsep *Overriding.*

class Apple extends Fruit {

    public $shape;

    function \_\_construct($name, $color, $shape) {

      parent::\_\_construct($name, $color);

      $this->shape = $shape;

    }

    function detail() {

      echo "I love " . $this->name . ", its " . $this->color . ", and its " . $this->shape;

    }

  }

  class Banana extends Fruit {

    public $shape;

    function \_\_construct($name, $color, $shape) {

      parent::\_\_construct($name, $color);

      $this->shape = $shape;

    }

    function detail() {

      echo "I love " . $this->name . ", its " . $this->color . ", and its " . $this->shape;

    }

  }

  $apple = new Apple("Apple", "Red", "Round");

  $banana = new Banana("Banana", "Yellow", "Curved");

  $apple->detail();

  echo "<br>";

  $banana->detail();

1. Buat *Class* Kalkulator yang memiliki tiga *Child Class*, yaitu Tambah, Kurang, dan Bagi. *Parent Class* mempunyai *property* $angka1 dan $angka2. Semua *Class* mempunyai *method* hasil(). *Method* hasil() pada *Parent Class* hanya menampilkan angka saja tanpa dilakukan proses aritmatika, sedangkan *Child Class* melakukan proses aritmatika sesuai dengan nama *Class*-nya.

<?php

class Kalkulator {

    protected $angka1;

    protected $angka2;

    public function \_\_construct($angka1, $angka2) {

        $this->angka1 = $angka1;

        $this->angka2 = $angka2;

    }

    public function hasil() {

        echo "{$this->angka1} dan {$this->angka2}";

    }

}

class Tambah extends Kalkulator {

    public function hasil() {

        echo $this->angka1 + $this->angka2;

    }

}

class Kurang extends Kalkulator {

    public function hasil() {

        echo $this->angka1 - $this->angka2;

    }

}

class Bagi extends Kalkulator {

    public function hasil() {

        echo $this->angka1 / $this->angka2;

    }

}

$kalkulator = new Kalkulator(10, 5);

$kalkulator->hasil(); // output: 10 dan 5

$tambah = new Tambah(10, 5);

$tambah->hasil(); // output: 15

$kurang = new Kurang(10, 5);

$kurang->hasil(); // output: 5

$bagi = new Bagi(10, 5);

$bagi->hasil(); // output: 2

*Codingan* diatas merupakan bagian dari praktik menggunakan OOP dalam bahasa PHP, hasil dari codingan tersebut dapat dilihat pada *chrome* dengan hasil dari masing-masing codingan dari nomor tersebut penjelasannya sebagai berikut :

1. Pada praktik pertama akan menghasilkan penerapan *constructor* pada *class fruit* yang berfungsi sebagai inisialisasi nilai awal dari objek dan menetapkan nilai awal untuk properti objek.
2. Pada praktik kedua membuat Detailfruit yang menerapkan konsep *object type* dan *method* akan menghasilkan output “I Love $Apple and its $red.
3. Pada praktik ketiga masing-masing *child class* memiliki *property shape* dan *method* detail dengan menghasilkan output “I Love $Apple, its $Red and its $round dan “I Love $Banana, its $yellow and its $curved.
4. Pada praktik keempat codingan di atas, terdapat pembuatan class Kalkulator sebagai parent class, dan tiga child class yaitu Tambah, Kurang, dan Bagi. Pada class Kalkulator terdapat dua property yaitu $angka1 dan $angka2, dan method \_\_construct() untuk menginisialisasi nilai property tersebut. Kemudian terdapat method hasil() pada parent class yang hanya menampilkan nilai property $angka1 dan $angka2 saja tanpa melakukan proses aritmatika. *Pada masing-masing child class, yaitu Tambah, Kurang, dan* Bagi, terdapat method hasil() yang sesuai dengan nama class-nya. Ketika method hasil() dipanggil pada masing-masing child class, maka akan dilakukan proses aritmatika yang sesuai. Contoh penggunaannya adalah dengan membuat objek dari masing-masing child class, dan memanggil method hasil(). Sebagai contoh, dengan membuat objek $tambah dari class Tambah, dan memanggil method hasil(), maka akan dilakukan proses penjumlahan dari nilai property $angka1 dan $angka2 yang dimiliki oleh class Kalkulator.
5. Kesimpulan

Dari praktikum ini dapat disimpulkan bahwa pemrograman berorientasi objek (OOP) merupakan paradigma pemrograman yang sangat berguna untuk membuat program yang kompleks dan modular. Konsep dasar OOP, seperti Constructor, Object Type, Inheritance, dan Overriding pada PHP dapat membantu programmer dalam membuat kode yang lebih bersih, terstruktur, mudah dipahami. 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