A. PENGENALAN OOP

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.

Dalam pengertian sederhananya, OOP adalah konsep pembuatan program dengan memecah permasalahan program dengan menggunakan objek. Objek dapat diumpamakan dengan 'fungsi khusus' yang bisa berdiri sendiri. Untuk membuat sebuah aplikasi, berbagai objek akan saling bertukar data untuk mencapai hasil akhir.

Konsep Dasar OOP:

1. **Class** (Kelas) Dan **Object** (Objek)
   Class adalah 'cetak biru' atau 'blueprint' dari object. Class digunakan hanya untuk membuat kerangka dasar. Yang akan kita pakai nantinya adalah hasil cetakan dari class, yakni **Object**. Analoginya Seperti ini, Kita Ibaratkan Class adalah Mahasiswa . Kita tahu bahwa Mahasiswa memiliki data seperti Nim,Nama,Kelas. Kita sudah memiliki kerangkanya dan untuk menggunakan maka kita membuat sebuah Objek, sehingga kita dapat membuat Objek seperti Mahasiswa Asvarizal dengan data Nim : 15.11.9292 , Nama: Asvarizal Filcha, Kelas : 15-S1IF-12. Dikarenakan kelas adalah kerangka dasar maka kita dapat membuat objek yang berbeda – beda sesuai dengan kebutuhan.

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

   Struktur class yang dimaksud adalah property dan method. Dengan enkapsulasi, kita bisa membuat pembatasan akses kepada **property** dan **method**, sehingga hanya property dan method tertentu saja yang bisa diakses dari luar class. Enkapsulasi juga `dikenal dengan istilah '**information hiding**'.

3. **Inheritance** (Turunan)
   Turunan merupakan suatu karakteristik OOP dimana class (parent class/ base class) dapat diturunkan ke class lain (child class / derived class) sehingga class anak dapat memiliki data atau perilaku class orang tua nya.

4. **Polymorphism** (Banyak Bentuk)
   Dari segi bahasa, Polimorfisme (bahasa inggris: Polymorphism) berasal dari dua kata bahasa latin yakni poly dan morph. Poly berarti banyak, dan morph berarti bentuk. Polimorfisme berarti banyak bentuk (wikipedia).

   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.

B. LET'S START CODE OOP PHP

1. Class

Class bisa di ibaratkan **pembungkus untuk property dan method dalam OOP**. Berikut contoh penulisan class :

```php
<?php
    class Mahasiswa{
        // Property dan Method ditulis disini
    }
?>
```

Pada kode diatas, kita membuat class bernama Mahasiswa yang akan menjadi tempat property dan method yang akan dibuat, cara penulisan class adalah dengan menggunakan huruf besar diawal.

2. Object

Setelah membuat class, kita akan menggunakan nya dengan membuat object. Object digunakan untuk suatu object berdasarkan class. Contoh class adalah Mahasiswa, maka object adalah mahasiswa itu, contohnya nama mahasiswa.

```php
<?php
    class Mahasiswa{
        // Property dan Method ditulis disini
    }
    $rizal = new Mahasiswa(); // Object
?>
```

Cara menuliskan objek adalah dengan sintak **new** dan dilanjutkan nama kelas.

3. Access Modifier

Access Modifier adalah hak akses yang ada di OOP. Ada 3 Access Modifier yaitu :

a. Public

Ketika property atau method memiliki akses Public, maka property atau method tersebut **bisa di akses dari luar class atau bisa di akses langsung oleh object**.

b. Protected

Ketika property atau method memiliki akses Protected, maka property atau method tersebut **hanya bisa di akses dari luar class atau bisa di akses langsung class turunannya,** property atau method tidak dapat diakses langsung dari object atau luar class.

c. Private

Ketika property atau method memiliki akses Private, maka property atau method tersebut **hanya bisa di akses dalam class itu sendiri dan tidak bisa di akses dari luar class atau dari class turunannya**.

4. Property

Property adalah **variable yang ada didalam class**. Kalau di ibaratkan sebagai Mahasiswa, property adalah segala sesuatu yang berhubungan dengan mahasiswa, seperti Nama, Nim, Kelas, Jenis Kelamin, Jurusan dan sebagainya.

Cara penulisan property adalah dengan menuliskan Access Modifier lalu membuat nama variable (property). Contoh :

```php
<?php
    class Mahasiswa{
        public $nama = 'Asvarizal Filcha'; // Property
        public $jurusan; // Property

    }
    $rizal = new Mahasiswa(); // Object
    echo $rizal->nama;
?>
```

$nama dan **$jurusan** adalah property dengan hak akses public, artinya bisa langsung di akses melalui object. **echo $rizal->nama;** ini adalah contoh menampilkan data dari property melaui object maka hasilnya :

Asvarizal Filcha

5. Method

Method adalah **fungsi yang ada dalam class.** Method bila di ibaratkan dari class Mahasiswa, method adalah apa saja yang dapat dilakukan oleh mahasiswa, contohnya seperti MelihatBiodataMahasiswa, MembayarSPP, MengambilKRS dan sebagainya.

Cara penulisan method adalah dengan menuliskan Access Modifier lalu membuat nama fungsi. Contoh :

```php
<?php
    class Mahasiswa{
        public $nama = 'Asvarizal Filcha'; // Property
        public $jurusan = 'Informatika'; // Property

        public function bayarSPP(){ // Method
            echo "Anda membayar SPP";
        }

    }
    $rizal = new Mahasiswa(); // Object
    $rizal->bayarSPP();
?>
```

bayarSPP() adalah method dengan hak akses public, artinya bisa langsung di akses melalui object. **$rizal->bayarSPP();** ini adalah contoh memangil method melaui object maka hasilnya :

Anda membayar SPP

6. **Variable this**

Variable this yang dituliskan $this adalah **sebuah variable yang menandakan kalau property atau method berada dalam class tersebut.** Contoh:

```php
<?php
    class Mahasiswa{
        public $nama = 'Asvarizal Filcha'; // Property

        public function tampilkanNama(){ // Method
            echo $this->nama; // Contoh dari penggunaan $this
        }

    }
    $rizal = new Mahasiswa(); // Object
    $rizal->tampilkanNama();
?>
```

Hasil :

Asvarizal Filcha

7. **Setter and Getter**

Setter adalah **method yang digunakan untuk memasukkan data kedalam property**, sedangkan Getter adalah **method yang digunakan untuk mendapakan data dari property.** Fungi dari setter dan getter adalah untuk mengolah property yang memiliki hak akses private atatu protected. Contoh setter dan getter :

```php
<?php
    class Mahasiswa{
        private $nama; // Property

        public function setNama($nama){ // Contoh Setter
            $this->nama = $nama;
        }

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

    }
    $rizal = new Mahasiswa(); // Object
    $rizal->setNama("Asvarizal"); // Memanggil Setter
    echo $rizal->getNama(); // Memanggil getter
?>
```

Hasil :

Asvarizal

8. **Constructor** dan **Destructor**

Constructor dan Destructor adalah method bawaan dari OOP PHP. Constructor adalah **fungsi yang paling pertama kali dijalan ketika object dibuat**. Sedangkan Destructor adalah **fungsi yang terakhir kali dijalankan ketika semua kode sudah dijalankan**. Constructor dan Destructor **harus memiliki hak akses Public.** Adapun cara penulisan Constructor dan Destructor adalah dengan menuliskan __construct() dan __ destruct(). Contoh :

```php
<?php
    class Mahasiswa{

        public function __construct(){ // Contoh Constructor
            echo "Fungsi Construct Terpanggil";
        }

        public function __destruct(){ // Contoh Destructor
            echo "Fungsi Destruct Terpanggil";
        }

    }
    // function __construct() terpanggil ketika objek ini dibuat
    $rizal = new Mahasiswa(); // Object
    echo "<br>Batas <br>";
    // function __destruct() terpanggil ketika file berakhir
?>
```

Hasil :

Fungsi Construct Terpanggil
Batas
Fungsi Destruct Terpanggil

9. **Inheritance**

Inheritance/Pewarisan merupakan salah satu konsep OOP, yang dimaksud pewarisan adalah **class anak / sub class dapat memiliki sifat dari parent class / base class.** Di ibaratkan manusia, maka anak akan memiliki bawaan dari orang tuanya, baik itu sifat maupun penampilan, Jika orang tua nya memiliki rambut hitam maka kemungkinan besar anaknya memiliki rambut hitam.

Dalam pewarisan, **property atau method harus memiliki hak akses public atau protected agar bisa diturunkan kepada class anak**. Pewarisan dilakukan dengan menuliskan sintaks **extends** di class anak, Contoh :

```php
<?php
    class OrangTua{
        protected $rambut = "hitam";
        public function warnaRambut(){
            echo "Warna rambutnya adalah ". $this->rambut;
        }
    }
    class Anak extends OrangTua{ // Contoh pewarisan
    }
    $rizal = new Anak();
    // class Anak bisa memiliki fungsi warnaRambut dari class OrangTua
    echo $rizal->warnaRambut();
?>
```

Hasil :

Warna rambutnya adalah hitam

Apabila tidak di override/overriding (Menulis property atau method ulang) maka akan menjalankan method bawaan dari class OrangTua. Berikut contoh override, dengan menambahkan property dan method dengan nama yang sama tapi isi berbeda pada class Anak.

```
1.  class Anak extends OrangTua{ // Contoh pewarisan
2.      //Contoh dari overriding property dari class OrangTua
3.      protected $rambut = "merah";
4.
5.      //Contoh dari overriding method dari class OrangTua
6.      public function warnaRambut(){
7.          echo "Ternyata rambutnya adalah ". $this->rambut;
8.      }
9.  }
```

Hasil :

Ternyata rambutnya adalah merah

10. **Abstract Class**

Abstract class sama seperti class biasa, bedanya **abstract class tidak bisa langsung dijadikan object melainkan harus class anak yang menjadi object**.
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**. Contoh penggunaan Abstrack class :

```
1.  <?php
2.      abstract class MakhlukHidup{ // contoh Abstract Class
3.          protected $nafas = "hidung";
4.          protected $jenisMakhluk = "Manusia";
5.          public function bernafas(){
6.              echo $this->jenisMakhluk. " bernafas melalui ". $this->nafas;
7.          }
8.      }
9.
10.     class Manusia extends MakhlukHidup{ // Contoh pewarisan
11.
12.     }
13.
14.     $orang = new Manusia();
15.     // class Manusia memiliki fungsi bernafas dari class abstract MakhlukHidup
16.     echo $orang->bernafas();
17. ?>
```

Hasil :

Manusia bernafas melalui hidung

11. Interface

Interface adalah **suatu aturan method atau property yang harus ada dalam suatu class,** class harus melakukan override dari interface yang digunakan, cara menggunaka interface adalah dengan sintaks **implements** dan untuk membuat interface dengan sintaks **interface** berikut contoh penggunaan interface :

```php
<?php

    interface Praktikum{ // Contoh penulisan interface
        public function berdoa(); // method yang harus di override
    }

    // Contoh penggunaan interface dengan implements
    class Mahasiswa implements Praktikum{
        public function berdoa(){
            echo "Sebelum memulai praktikum harus berdoa";
        }
    }

    $mhs = new Mahasiswa();
    echo $mhs->berdoa(); // Pemanggilan fungsi berdoa();
?>
```

Hasil :

Sebelum memulai praktikum harus berdoa

Apabila Mahasiswa tidak membuat method berdoa() maka akan terjadi error.

12. Scope Resolution Operator

Scope Resolution Operator adalah pemanggilan property/method secara langsung dengan nama class tanpa membuat objek, contoh pemanggilan nya adalah nama class dilanjutkan double colon lalu nama method atua property contoh **Mahasiswa::berdoa()** lebih lengkapnya seperti contoh berikut :

```php
<?php

    class Mahasiswa{
        public function berdoa(){
            echo "Sebelum memulai praktikum harus berdoa";
        }
    }

    // Contoh penggunaan Scope Resolution Operator
    Mahasiswa::berdoa();
?>
```

Hasil :

Sebelum memulai praktikum harus berdoa

13. Static property dan static method

Static property dan static method adalah **property (variabel) dan method (function) yang melekat kepada class, bukan kepada objek**. Konsep static property memang 'agak keluar' dari konsep objek sebagai tempat melakukan proses, karena sebenarnya class hanya merupakan 'blueprint' saja.

Untuk membuat static property dan static method, kita menambahkan keyword 'static' setelah penulisan hak akses property atau method, seperti contoh berikut:

```php
<?php
    class Mahasiswa{
        public static $agama;
        public static function berdoa(){
            echo "Sebelum memulai praktikum harus berdoa Menurut Agama ";
        }
    }

    // set static property
    Mahasiswa::$agama = "Islam";
    // Memanggil static Method
    Mahasiswa::berdoa();
    // get static property
    echo Mahasiswa::$agama;
?>
```

Hasil :

Sebelum memulai praktikum harus berdoa Menurut Agama Islam

Dikarenakan static melekat pada class bukan objek maka pemanggilannya melalui class, dalam static method tidak dapat menggunakan **$this** dikarenakan tidak menggunakan objek dalam pemanggilannya.

14. Self keyword

Self keyword adalah keyword yang berfungsi untuk memanggil property/method milik dirinya sendiri, sederhananya **self keyword memanggil property atau method pada class dimana self keyword itu digunakan**, contoh penggunaan :

```php
<?php
    class OrangTua{
        public function tempat(){
            echo "Ini Fungsi Dari Class Orang Tua <br>";
        }
        public function tampil(){
            self::tempat(); // Contoh Penggunaan self
            $this->tempat(); // Contoh Penggunaan this
        }
    }
    class Anak extends OrangTua{
        public function tempat(){
            echo "Ini Fungsi Dari Class Anak <br>";
        }
    }

    $manusia = new Anak();
    //menampilkan method tampil() melalui objek dari class anak
    $manusia->tampil();
?>
```

Hasil :

Ini Fungsi Dari Class Orang Tua
Ini Fungsi Dari Class Anak

15. Parent keyword

Parent keyword adalah **keyword untuk memanggil property atau method milik parent class**, sehingga walau pun property atau method sudah di override di class anak, tetap bisa menggunakan class asli. Berikut contoh penggunaan parent keyword :

```php
<?php
    class OrangTua{
        public function tempat(){
            echo "Ini Fungsi Dari Class Orang Tua <br>";
        }
    }
    class Anak extends OrangTua{
        public function tampil(){
            parent::tempat(); // Contoh Penggunaan parent
            $this->tempat(); // Contoh Penggunaan this
        }
        public function tempat(){
            echo "Ini Fungsi Dari Class Anak <br>";
        }
    }

    $manusia = new Anak();
    //menampilkan method tampil() melalui objek dari class anak
    $manusia->tampil();
?>
```

Hasil :

Ini Fungsi Dari Class Orang Tua
Ini Fungsi Dari Class Anak

DAFTAR PUSTAKA

1. http://en.wikipedia.org/wiki/Object-oriented_programming
2. https://www.duniailkom.com
3. Buku “Bikin Framework PHP Sendiri dengan Teknik OOP & MVC”, David Naista.