MODUL PRAKTIKUM

PEMROGRAMAN BERBASIS WEB LANJUTAN

DOSEN PENGAMPU

SUENDRI, M.Kom

PROGRAM STUDI SISTEM INFORMASI

FAKULTAS SAINS DAN TEKNOLOGI

UIN SUMATERA UTARA MEDAN

2020

KATA PENGANTAR

Puji dan syukur kepada Allah ‘Azza Wajalla yang telah melimpahkan rahmat dan kasih sayangNya sehingga Modul Praktikum ini selesai disusun dengan baik. Selawat dan salam juga teruntuk nabi junjungan alam, Muhammad Shollollohu ‘Alaihi Wasallam sebagai panutan manusia hingga akhir zaman. Modul Praktikum ini disusun sebagai kelengkapan matakuliah Pemrograman Berbasis Web Lanjutan pada Program Studi Sistem Informasi Universitas Islam Negeri (UIN) Sumatera Utara Medan. Modul Praktikum ini hanya untuk kepentingan perkuliahan, digunakan pada kalangan sendiri. Jika terdapat kutipan, seluruh referensi tersebut dicantumkan dengan lengkap pada daftar pustaka.

Penulis menyadari masih banyak perbaikan yang diperlukan dalam penyusunan, aturan penulisan dan tata letak dari Modul Praktikum ini, oleh kerena itu penulis mengharapkan kritik dan saran yang membangun. Penulis juga mengucapkan terimakasih kepada seluruh pihak yang dijadikan referensi dalam penulisan Modul Praktikum ini, baik dari sisi perangkat lunak, perangkat keras serta buah pikiran dari pakar-pakar terdahulu.

Medan, Maret 2020

Suendri, M.Kom

DAFTAR ISI

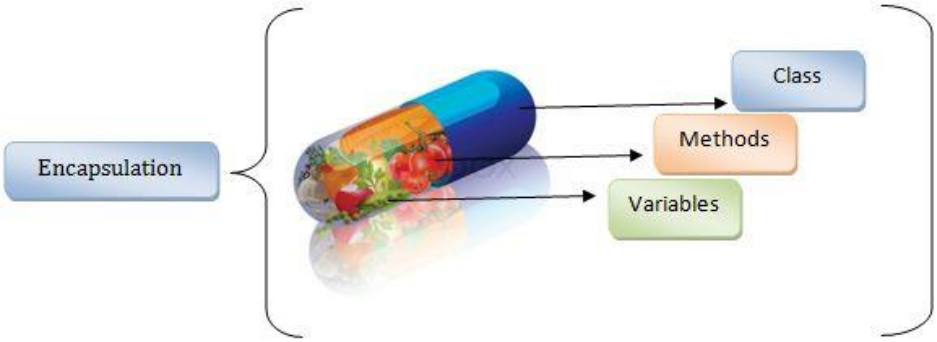
PRAKTIKUM 4
ENCAPSULATION

4.1 Tujuan

Setelah mempelajari modul ini anda diharapkan mampu :

a. Memahami metode *Encapsulation* (Pembungkusan) dalam konsep OOP pada bahasa pemrograman PHP.
b. Memahami dan mengetahui bagian dan fungsi yang membangun metode *encapsulation* pada bahasa pemrograman PHP.
c. Memahami dan merancang sistem yang dilengkapi dengan metode *encapsulation*.

4.2 Praktikum



Encapsulation (Pembungkusan) sebenarnya bukanlah proses yang baru dari materi yang sudah kita bahas pada praktikum sebelumnya. *Encapsulation* adalah proses membungkus *class* dan menjaga apa saja yang ada di dalam *class* tersebut agar tidak bisa diakses sembarangan dari *class* lainnya. Perintah yang sudah kita bahas yaitu penggunaan *Modifier* ***Private*** dan ***Protected*** pada *Property* maupun *Method.* Dengan *Modifier Private* maka *Property* dan *Method* hanya bisa diakses oleh *class* itu sendiri, sedangkan *Modifier Protected* hanya bisa diakses oleh *Class* itu sendiri dan Anak *Class*-nya. Berikut ini keuntungan dari penggunaan *encapsulation* pada konsep OOP menggunakan bahasa pemrograman PHP

a. Penyembunyian data, struktur dari data dan detail yang tidak perlu diketahui pengguna dapat disembunyikan.

b. Keamanan data, proses encapsulation membuat data lebih kuat dan aman, hal ini disebabkan data terlindungi dan tidak ada pengaruh dari luar.
c. Mengurangi kompleksitas karena data tersembunyi.
d. Dapat digunakan kembali, property dan method dengan telah dimiliki oleh Class bisa digunakan kembali oleh Anak Class tanpa perlu ditulis ulang.
e. Handal, data bisa dijadikan hanya bisa di baca (*read-only*) atau hanya ditulis (*write-only*).
f. Mudah untuk diuji, penggunaan method pada anak class, memastikan method pada class induk sudah benar.
g. Meningkatkan fleksibilitas, bisa diimplementasikan pada class lain tanpa merubah kode.

a. Private

Private hanya bisa diakses *class* itu sendri, berikut contoh penggunaan private:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     private string $nim;
6.     public string $nama;
7.
8.     public function setNim(string $a) {
9.         $this->nim = $a;
10.       }
11.
12.    public function setNama(string $nama) {
13.           $this->nama = $nama;
14.       }
15.
16.    public function getNim() {
17.           return $this->nim;
18.       }
19.
20.  }
21.
22.  $mhsw = new Mahasiswa;
23.  $mhsw->setNim("17021000");
24.  $mhsw->setNama("Kiki");
25.
26.  echo "<p>" . $mhsw->getNim() . "</p>";
27.  echo "<p>" . $mhsw->nama . "</p>";
```

Pada contoh di atas, `$mhsw->nama` bisa diakses langsung pada saat ditampilkan ke layar, tapi tidak sebaliknya dengan `$mhsw->nim` karena *modifier*-nya *private*, maka dibantu dengan method `$mhsw->getNim()`.

b. *Protected*

Protected bisa diakses oleh *class* itu sendiri dan turunannya, berikut ini contoh penggunaan private:

```
1. <?php
2.
3. class Mahasiswa {
4.
5.     private string $nim = "17021000";
6.     protected string $nama = "Kiki";
7.
8. }
9.
10.  class Nilai extends Mahasiswa {
11.
12.      public function getNim() {
13.          return $this->nim;
14.      }
15.
16.      public function getNama() {
17.          return $this->nama;
18.      }
19.  }
20.
21.  $mhsw = new Nilai();
22.
23.  echo "<p>" . $mhsw->getNim() . "</p>";
24.  echo "<p>" . $mhsw->getNama() . "</p>";
```

Pada contoh diatas `$mhsw->getNim()` tidak bisa dieksekusi karena `$nim` mempunyai *modifier private*, sedangkan `$mhsw->getNama()` bisa dieksekusi karena mempunyai *modifier protected*.

Pada dua contoh diatas digunakan pada *Property*, begitu juga penggunaan pada *Method* dengan cara yang sama.

4.3 Latihan

a. Jawablah quiz yang disedikan di *e-learning* pada pertemuan 4 untuk menguji kemampuan anda sampai dengan materi praktikum ini.