MODUL

PEMOGRAMAN WEB II

Oleh:

CHALIFA CHAZAR

MODUL 9

Kelas dan Objek

Tujuan:

Mahasiswa memahami penggunaan model pemograman berorintasi objek (OOP - *Object Oriented Programming*) untuk menyelesaikan permasalahan tertentu dalam kode PHP.

Pustaka:

Raharjo, B. (2015): *Mudah Belajar PHP Teknik Penggunaan Fitur-Fitur Baru Dalam PHP 5*. Informatika. Bandung

Prasetio, A. (2015): *Buku Pintar Webmaster*.

www.w3schools.com

Dalam PHP, terdapat dua model pemograman, yaitu model pemograman prosedural dan model pemograman berorientasi objek (OOP - *Object Oriented Programming*). Pada modul-modul sebelumnya kita menggunakan model pemograman prosedural untuk menuliskan contoh-contoh kode program.

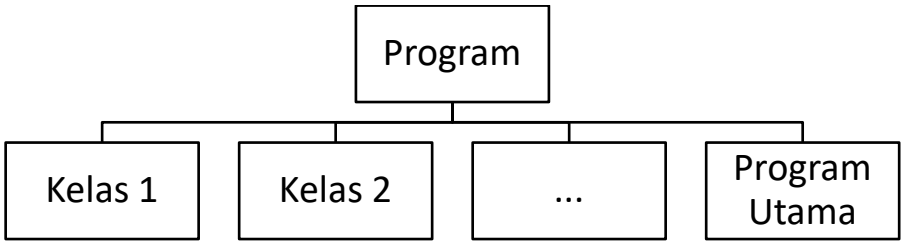
1. Pengertian Kelas dan Objek

Inti dari pemograman berorientasi objek adalah pembentukan kelas. Kelas merupakan model atau abstraksi yang merepresentasikan objek-objek yang ada di dalam dunia nyata. Kelas juga dapat didefinisikan sebagai kerangka atau cetak-biru (*blueprint*) dari suatu objek tertentu.

Setiap objek memiliki data dan kemampuan. Contohnya, objek mobil. Mobil memiliki data berupa tahun pembuatan, merek, tipe, warna, nomor polisi, dan lain-lain. Selain data, mobil juga memiliki kemampuan seperti, maju, berhenti, mundur, belok kanan, belok kiri, dan lain-lain.

Dalam konteks pemograman, data yang dimiliki oleh suatu objek disebut sebagai properti (atau atribut) dan kemampuan objek disebut sebagai metode. Properti direpresentasikan menggunakan variabel, sedangkan metode direpresentasikan menggunakan fungsi.

Proses pembentukan kelas dalam hal ini sebenarnya adalah proses memodelkan data dan kemampuan dari suatu objek ke dalam bentuk kode. Pemograman berorientasi objek merupakan pemograman yang tersusun oleh banyak kelas, yang selanjutnya diinstansiasi menjadi objek.



Esensinya, kelas adalah tipe data bentukan. Setiap kita mendefinisikan kelas maka kita membentuk suatu tipe data baru yang masih bersifat abstrak. Contoh nyata dari kelas disebut instace, atau dikenal juga dengan istilah objek. Perhatikan contoh berikut ini.

```
<?php
//contoh kelas
class Mobil {
	public $tahun;
	public $merek;
	public $tipe;
	public $warna;

	public function nyalakanMesin() {
		// ..............
	}
	public function matikanMesin() {
		// ..............
	}
	public function maju() {
		// ..............
	}
	public function berhenti() {
		// ..............
	}
}

//contoh objek
$mobil1 = new Mobil();
$mobil1->tahun = 2016;
$mobil1->merek = "Toyota Inova";
$mobil1->tipe = "GX";
$mobil1->warna = "Grey";
$mobil1->nyalakanMesin();
$mobil1->maju();
$mobil1->berhenti();
?>
```

Pada kode di atas, **`Mobil`** adalah kelas. Perintah **`new Mobil()`** akan menciptakan *instance* (objek) dari kelas **`Mobil`** dan memasukan objek tersebut ke variabel bertipe referensi dengan nama **`$mobil1`**. Maka, **`$mobil1`** dapat dikatakan sebagai objek.

2. Prinsip Dasar dalam Pemograman Berorientasi Objek

Suatu bahasa pemograman yang mendukung pemograman berorientasi objek harus dapat mengimplementasikan 3 prinsip dasar dari konsep-konsep berikut ini.

- Pembungkusan (*encapsulation*)
- Pewarisan (*inheritance*)
- Polimorfisme (*polymorphism*)

2.1 Pembungkusan

Pembukusan adalah konsep penggabungan properti dan metode ke dalam suatu entitas tunggal yang disebut kelas. Saat kita membuat kelas, maka kita sedang mengimplementasikan konsep pembukusan.

Dalam mendefinisikan suatu kelas, terdapat suatu ketentuan yang mengatur tingkat akses dari properti atau metode. Tingkat akases ini akan menentukan bagaimana properti atau metode dapat diakses dari lingkungan di luar kelas. Dalam PHP, penentuan tingkat akses (*access-specifier*) dibagi menjadi 3 jenis, yaitu: *private*, *protected*, dan *public*.

Tingkat Akses `pivate`

Properti dan metode di dalam suatu kelas yang didefinisikan dengan tingkat akses `private` hanya dapat diakses oleh kelas yang bersangkutan. Kelas lain (meskipun kelas turunan) dan kode lain di luar kelas tidak diizinkan untuk mengakses properti dan metode tersebut.

```
<?php
class A {
  private $a1;
}
$obj = new A();
$obj->a1 = 2;
//obj tidak dapat mengakses class A()
?>
```

Tingkat Akses `protected`

Properti dan metode di dalam suatu kelas yang didefinisikan dengan tingkat akses `protected` akan dapat diakses oleh kelas yang bersangkutan dan kelas-kelas lain yang masih merupakan kelas turunan. Kode luar dan kelas lain yang bukan kelas turunan tetap tidak diizinkan untuk mengakses data dan fungsi tersebut.

```
<?php
class A {
  protected $a2;
}
class B extends A {
  public setA($nilai) {
     $this->a2 = 2;
     //dapat mengakses $a2 karena kelas B merupakan turunan dari kelas A
  }
}

$obj = new A();
$obj->a2 = 2;
//obj tidak dapat mengakses class A()
?>
```

Tingkat Akses `public`

Properti dan metode di dalam suatu kelas yang didefinisikan dengan tingkat akses `public` akan dapat diakses oleh seluruh kelas yang ada, baik di dalam kelas itu sendiri, kelas turunan, maupun kelas lain yang bukan turunan, bahkan oleh kode (non-kelas) lainnya.

Perhatikan contoh berikut ini.

```
<?php
class A {
  public $a3;
}

//mengakses $a3 dari kelas turunan
class B extends A {
  public function setA($nilai) {
     $this->a3 = nilai;
  }
}

//mengakses $a3 dari kelas lain
class C {
  public function setA($nilai) {
     A::$a3 = nilai;
  }
}

// mengkases $a3 dari kode
$obj = new A();
$obj->a3 = 2;
?>
```

2.2. Pewarisan

Pewarisan adalah proses pembentukan kelas baru yang diturunkan dari kelas-kelas lain yang sebelumnya sudah ada. Kelas turunan secara otomatis akan mewarisi sifat-sifat yang dimiliki oleh kelas induk.

2.3. Polimorfisme

Polimorfisme adalah suatu kejadian dimana suatu objek yang didefinisikan menggunakan kelas induk dapat berperan sebagai objek dari kelas-kelas turunan. Dengan demikian objek tersebut memiliki banyak bentuk (dapat melakukan banyak hal yang berbeda melalui cara yang sama).

3. Mendefinisikan Kelas

Kelas didefinisikan menggunakan kata kunci `class`, dan diikuti dengan nama kelasnya. Badan kelas dibuat menggunakan tanda { dan }.

Perhatikan contoh berikut ini.

```
class NamaKelas {
  private $propertiPrivate1;
  ...
  protected $propertiProtected1;
  ...
  public $propertiPublic1;
  ...
  public function NamaMetode1($param,...) {
    //isi metode
  }
  ...
}
```

Contoh kelas sederhana

```
<?php
//mendefinisikan kelas
class Balok {
       //properti
       public $panjang;
       public $lebar;
       public $tinggi;
}

//membuat objek dari kelas Balok
$obj = new Balok();

//mengisi nilai ke dalam properti objek
$obj->panjang = 6;
$obj->lebar = 4;
$obj->tinggi =2;

//menghitung volume
$volume = $obj->panjang * $obj->lebar * $obj->tinggi;

//menampilkan volume
echo "Volume Balok: " . $volume;
?>
```

4. Mendefinisikan Metode

Seperti yang tela disebutkan bahwa metode merepresentasikan kemampuan atau operasi yang dapat dilakukan oleh suatu objek. Pada contoh sebelumnya operasi perhitungan volume masih dilakukan di luar kelas. Operasi tersebut sebenarnya bisa dilakukan di dalam kelas, yaitu dengan mendefinisikannya sebagai metode.

Perhatikan contoh berikut ini.

```
<?php
//mendefinisikan kelas
class Balok {
	//properti
	public $panjang;
	public $lebar;
	public $tinggi;

	//mendefinisikan metode
	public function volume() {
		return $this->panjang * $this->lebar * $this->tinggi;
	}
}

//membuat objek dari kelas Balok
$obj = new Balok();

//mengisi nilai ke dalam properti objek
$obj->panjang = 6;
$obj->lebar = 4;
$obj->tinggi =2;

//menghitung volume
$volume = $obj->volume();

//menampilkan volume
echo "Volume Balok: " . $volume;
?>
```

Anda juda dapat memasukan operasi lainnya, misalnya mencetak nilai volume ke dalam balok. Melalui cara seperti ini, kode di luar kelas hanya perlu memanggil metode untuk menampilkan nilai volume ke layar.

Tugas 1
Coba lakukan perubahan pada contoh `class Balok()` diatas, sehingga untuk mencetak nilai volume dilakukan dengan menggunakan metode.

5. Menggunakan Kata Kunci `private`

Dalam merancang suatu kelas, pada umumnya data atau variabel tidak boleh langsung diakses dari kode di luar kelas. Maka dari itu, kita perlu mendeklarasikannya dengan tingkat akses `private`. Oleh sebab itu untuk mengakses variabel-variabel di dalamnya kita perlu mendefinisikan metode yang berperan sebagai antarmuka atau jembatan antara data `private` di dalam kelas dengan kode di luar kelas.

Tugas 2
Coba lakukan perubahan pada contoh `class Balok()` diatas ke dalam akses `private`, sehingga untuk mengakses variabel-variabel dilakukan melalui suatu antarmuka.

6. Method Chaining

Pada contoh sebelumnya kita sudah membuat object dan berhasil mengakses *property* atau *method*, baik itu dari *global scope* (luar *class*) atau dari dalam *class*. Untuk mengakses *property* atau *method* dari dalam *class*, kita dapat menggunakan keyword $this.

Hal yang mengagumkan terjadi ketika suatu *method* mengembalikan keyword $this, kita bisa menyambung pemanggilan *method-method* tersebut. *Method* chaining adalah **pemanggilan method yang berantai**.

Perhatikan contoh berikut ini.

```
<?php
//mendefinisikan kelas
class Chain {

	private $nilaiAwal = 0;

	public function tambahkan($nilai)
	{
		//$nilaiAwal = $nilaiAwal + $nilai
		$this->nilaiAwal += $nilai;
		return $this;
	}

	public function kurangi ($nilai)
	{
		$this->nilaiAwal -= $nilai;
		return $this;
	}

	public function hasil()
	{
		return $this->nilaiAwal;
	}
}

//membuat instance objek dari kelas
//tanpa method chain
$a = new Chain();
$a->tambahkan(100);
$a->kurangi(10);
echo $a->hasil();
?>
```

Perhatikan contoh berikut ini.

```
//dengan method chain
$b= new Chain();
$b->tambahkan(100);
$b->kurangi(10);
echo $b->hasil();
```

Dalam kasus nyata, teknik ini digunakan untuk mempersingkat penulisan kode. Selain itu juga dapat meningkatkan "keterbacaan" kode program.

7. Mendefinisikan Konstruktor

Konstruktor adalah metode khusus yang akan dipanggil secara otomatis ketika suatu objek dari kelas tertentu dibuat. Konstruktor biasanya digunakan untuk melakukan inisialisasi atau pengisian nilai awal terhadap properti-properti yang terdapat di dalam suatu objek. Dalam PHP, konstruktor didefinisikan menggunakan fungsi `__construct()`.

8. Mendefinisikan Destruktor

PHP menyediakan fasilitas *Garbage Collector* (GC) yang akan membuang objek secara otomatis ketika kode selesai dieksekusi atau suatu objek dari kelas tertentu sudah tidak lagi diacu oleh referensi manapun.

Destruktor adalah kebalikan dari konstruktor, yang akan dipanggil secara otomatis sesaat sebelum objek dibuang dari memori oleh GC. Destruktor umumnya digunakan ketika mendefinisikan sebuah objek baru di dalam konstruktor. Bisa juga digunakan ketika kita mengakses file atau database di dalam konstruktor, dan kita ingin menutup file atau koneksi database tersebut melalui destruktor. Dalam PHP, destruktor didefinisikan menggunakan fungsi `__destruct()`.

Tugas 3
Buat sebuah program perhitungan matematika yang dibuat kedalam 10 buah class yang berbeda-beda. Buatlah fungsi sendiri (tidak menggunakan pustaka yang ada dalam PHP).