MODUL 9
PEMROGRAMAN BERBASIS OBJEK PHP

1. Pengertian Pemograman Berbasis Objek

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.

Berbeda dengan konsep fungsi atau 'function' di dalam pemrograman, sebuah objek bisa memiliki data dan function tersendiri. Setiap objek ditujukan untuk mengerjakan sebuah tugas, dan menghasilkan nilai akhir untuk selanjutnya dapat ditampilkan atau digunakan oleh objek lain.

2. Fungsi Pemrograman Berbasis Objek dalam PHP

PHP bukan bahasa pemrograman yang 'murni' berbasis objek seperti Java. Bahkan, konsep OOP dalam PHP baru hadir dalam PHP versi 4, dan disempurnakan oleh PHP versi 5. Dengan kata lain, OOP di PHP merupakan 'fitur tambahan'. Anda bisa membuat situs web dengan PHP tanpa menggunakan objek sama sekali.

Dalam studi pemrograman, pembuatan program dalam PHP tanpa menggunakan objek disebut juga dengan pemrograman prosedural atau pemrograman fungsional. Dikenal dengan pemrograman prosedural, karena kita memecah kode program menjadi bagian-bagian atau fungsi-fungsi kecil, kemudian menyatukannya untuk menghasilkan nilai akhir.

Dengan membuat program secara prosedural, aplikasi bisa dibuat dengan cepat dan mudah dipelajari jika dibandingkan dengan pemrograman berbasis objek. Keuntungan pemrograman berbasis objek baru terasa ketika program tersebut telah 'besar' atau kita bekerja dengan tim untuk membagi tugas. Konsep 'objek' untuk memisahkan program menjadi bagian-bagian yang berdiri sendiri akan memudahkan dalam membuat program.

3. Pengertian Class, Object, Property dan Method

Pengertian Class dalam Pemrograman Berbasis 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.

Sebagai analogi, class bisa diibaratkan dengan laptop atau notebook. Kita tahu bahwa laptop memiliki ciri-ciri seperti merk, memiliki keyboard, memiliki processor, dan beberapa ciri khas lain yang menyatakan sebuah benda tersebut adalah laptop. Selain memiliki ciri-ciri, sebuah laptop juga bisa dikenakan tindakan, seperti: menghidupkan laptop atau mematikan laptop.

Class dalam analogi ini adalah gambaran umum tentang sebuah benda. Di dalam pemrograman nantinya, contoh class seperti: koneksi_database dan profile_user.

Di dalam PHP, penulisan class diawali dengan keyword class, kemudian diikuti dengan nama dari class. Aturan penulisan nama class sama seperti aturan penulisan variabel dalam PHP, yakni diawali dengan huruf atau underscore untuk karakter pertama, kemudian boleh diikuti dengan huruf, underscore atau angka untuk karakter kedua dan selanjutnya. Isi dari class berada dalam tanda kurung kurawal.
Berikut adalah contoh penulisan class dalam PHP:

```
<?php
class laptop {
  // isi dari class laptop...
}
?>
```

Pengertian Property dalam Pemrograman Berbasis Objek

Property (atau disebut juga dengan atribut) adalah data yang terdapat dalam sebuah class. Melanjutkan analogi tentang laptop, property dari laptop bisa berupa merk, warna, jenis processor, ukuran layar, dan lain-lain.

Jika anda sudah terbiasa dengan program PHP, property ini sebenarnya hanyalah variabel yang terletak di dalam class. Seluruh aturan dan tipe data yang biasa diinput kedalam variabel, bisa juga diinput kedalam property. Aturan tata cara penamaan property sama dengan aturan penamaan variabel.
Berikut adalah contoh penulisan class dengan penambahan property:

```
<?php
class laptop {
  var $pemilik;
  var $merk;
  var $ukuran_layar;
  // lanjutan isi dari class laptop...
}
?>
```

Dari contoh diatas, $merk, $ukuran_layar dan $jenis_processor adalah property dari class laptop. Seperti yang kita lihat, penulisan property di dalam PHP sama dengan cara penulisan variabel, yakni menggunakan tanda dollar ($). Sebuah class tidak harus memiliki property.
Pengertian Method dalam Pemrograman Berbasis Objek

Method adalah tindakan yang bisa dilakukan didalam class. Jika menggunakan analogi class laptop kita, maka contoh method adalah: menghidupkan laptop, mematikan laptop, mengganti cover laptop, dan berbagai tindakan lain.

Method pada dasarnya adalah function yang berada di dalam class. Seluruh fungsi dan sifat function bisa diterapkan kedalam method, seperti argumen/parameter, mengembalikan nilai (dengan keyword return), dan lain-lain.

Berikut adalah contoh penulisan class dengan penambahan method:

```
<?php
class laptop {
  function hidupkan_laptop() {
  //... isi dari method hidupkan_laptop
  }

  function matikan_laptop() {
  //... isi dari method matikan_laptop
  }

  ... //isi dari class laptop
}
?>
```

Dari contoh diatas, function hidupkan_laptop() dan function matikan_laptop() adalah method dari class laptop. Seperti yang kita lihat, bahwa penulisan method di dalam PHP sama dengan cara penulisan function. Sebuah class tidak harus memiliki method.

Pengertian Object dalam Pemrograman Berbasis Objek

Object atau Objek adalah hasil cetak dari class, atau hasil 'konkrit' dari class. Jika menggunakan analogi class laptop, maka objek dari class laptop bisa berupa: laptop_andi, laptop_anto, laptop_duniailkom, dan lain-lain. Objek dari class laptop akan memiliki seluruh ciri-ciri laptop, yaitu property dan method-nya.

Proses 'mencetak' objek dari class ini disebut dengan 'instansiasi' (atau instantiation dalam bahasa inggris). Pada PHP, proses instansiasi dilakukan dengan menggunakan keyword 'new'. Hasil cetakan class akan disimpan dalam variabel untuk selanjutnya digunakan dalam proses program.

Sebagai contoh, berikut adalah cara membuat objek laptop_andi dan laptop_anto yang dibuat dari class laptop:

```
<?php
class laptop {
  //... isi dari class laptop
  }
$laptop_andi = new laptop();
$laptop_anto = new laptop();
?>
```

Dari contoh diatas, $laptop_andi dan $laptop_anto merupakan objek dari class laptop. Kedua objek ini akan memiliki seluruh property dan method yang telah dirancang dari class laptop.

4. Cara Membuat dan Mengakses Objek dalam PHP

Cara Membuat Objek dalam PHP

Istilah Objek dalam Objek Oriented Programming (OOP), sebenarnya terdiri dari class, property, method dan object. Berikut adalah contoh cara pembuatan objek di dalam PHP:

```
<?php
// buat class laptop
class laptop {
  // buat property untuk class laptop
  var $pemilik;
  var $merk;
  var $ukuran_layar;

  // buat method untuk class laptop
  function hidupkan_laptop() {
    return "Hidupkan Laptop";
  }
  function matikan_laptop() {
    return "Matikan Laptop";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();
?>
```

Dalam kode diatas, kita telah membuat sebuah class dengan nama laptop, lengkap dengan property dan menthod-nya. Kemudian kita membuat 1 buah objek dari class laptop dengan nama $laptop_anto pada baris terakhir.

Walaupun dalam kode diatas kita telah membuat objek, objek tersebut belum menampilkan apa-apa, karena class laptop belum berisi data apapun. Kita akan segera mempelajari cara mengakses ‘isi’ dari class dalam PHP.

Cara Mengakses Objek dalam PHP

Cara mengakses objek yang kita maksud sebenarnya adalah cara untuk mengakses ‘isi’ dari sebuah objek, yakni property dan method-nya. Agar lebih mudah dipahami, berikut adalah revisi contoh class laptop sebelumnya:

```
<?php
// buat class laptop
class laptop {
  // buat property untuk class laptop
  var $pemilik;
```

```
  var $merk;
  var $ukuran_layar;

  // buat method untuk class laptop
  function hidupkan_laptop() {
    return "Hidupkan Laptop";
   }
  function matikan_laptop() {
    return "Matikan Laptop";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();

// set property
$laptop_anto->pemilik="Anto";
$laptop_anto->merk="Asus";
$laptop_anto->ukuran_layar="15 inchi";

// tampilkan property
echo $laptop_anto->pemilik;
echo "<br />";
echo $laptop_anto->merk;
echo "<br />";
echo $laptop_anto->ukuran_layar;
echo "<br />";

// tampilkan method
echo $laptop_anto->hidupkan_laptop();
echo "<br />";
echo $laptop_anto->matikan_laptop();
?>
```

5. Pengertian Enkapsulasi Objek (Public, Protected dan Private)

Pengertian Enkapsulasi (Encapsulation)

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’.

Dengan enkapsulasi, kita bisa memilih property dan method apa saja yang boleh diakses,

dan mana yang tidak boleh diakses. Dengan menghalangi kode program lain untuk mengubah property tertentu, class menjadi lebih terintegrasi, dan menghindari kesalahan ketika seseorang 'mencoba' mengubahnya. Programmer yang merancang class bisa menyediakan property dan method khusus yang memang ditujukan untuk diakses dari luar.

Melanjutkan analogi tentang class laptop, perusahaan pembuat laptop telah menyediakan 'method' khusus untuk menghidupkan laptop, yakni dengan cara menekan tombol on. Di dalam laptop sendiri, banyak 'method-method' lain yang akan dijalankan ketika kita menyalakan laptop, contohnya: mengirim sinyal booting ke processor, mengirim data dari processor ke memory, dan mengirim sinyal listrik ke LED di monitor. Akan tetapi, proses ini adalah method internal laptop dimana kita tidak perlu memahaminya untuk menghidupkan laptop.

Enkapsulasi Objek: Public, Protected dan Private

Untuk membatasi hak akses kepada property dan method di dalam sebuah class, Objek Oriented Programming menyediakan 3 kata kunci, yakni Public, Protected dan Private. Kata kunci ini diletakkan sebelum nama property atau sebelum nama method. Berikut adalah pembahasannya:

Pengertian Hak Akses: Public

Ketika sebuah property atau method dinyatakan sebagai public, maka seluruh kode program di luar class bisa mengaksesnya, termasuk class turunan. Berikut adalah contoh penulisan public property dan public method dalam PHP:

```
<?php
// buat class laptop
class laptop {
  // buat public property
  public $pemilik;
  // buat public method
  public function hidupkan_laptop() {
    return "Hidupkan Laptop";
  }
}
// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();
// set property
$laptop_anto->pemilik="Anto";

// tampilkan property
echo $laptop_anto->pemilik; // Anto
// tampilkan method
echo $laptop_anto->hidupkan_laptop(); // "Hidupkan Laptop"
?>
```

Jika hak akses property dan method tidak ditulis, maka PHP menganggapnya sebagai public.

Dalam contoh sebelumnya, kita membuat property class laptop menggunakan kata kunci var, seperti berikut ini:

var $pemilik;

Kata kunci var sebenarnya tidak perlu ditulis, bahkan pada PHP versi 5.0 sampai dengan 5.1.3, menggunakan kata var di dalam class akan menghasilkan warning. Untuk PHP versi 5.1.3 keatas, menggunakan kata var tidak lagi menghasilkan warning (dianggap sebagai public).

Agar sejalan dengan konsep enkapsulasi, setiap property dan method dalam class harus menggunakan kata kunci seperti public, protected, atau private. Oleh karena itu, membuat property dengan keyword var tidak disarankan lagi .

Pengertian Hak Akses: Protected

Jika sebuah property atau method dinyatakan sebagai protected, berarti property atau method tersebut tidak bisa diakses dari luar class, namun bisa diakses oleh class itu sendiri atau turunan class tersebut.

Apabila kita mencoba mengakses protected property atau protected method dari luar class, akan menghasilkan error, seperti contoh berikut ini:

```
<?php
// buat class laptop
class laptop {
  // buat protected property
  protected $pemilik;
 // buat protected method
  protected function hidupkan_laptop() {
    return "Hidupkan Laptop";
  }
}
// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();

// set protected property akan menghasilkan error
$laptop_anto->pemilik="Anto";
// Fatal error: Cannot access protected property laptop::$pemilik

// tampilkan protected property akan menghasilkan error
echo $laptop_anto->pemilik;
// Fatal error: Cannot access protected property laptop::$pemilik
```

```
// jalankan protected method akan menghasilkan error
echo $laptop_anto->hidupkan_laptop();
// Fatal error: Call to protected method laptop::hidupkan_laptop()
// from context
?>
```

Dalam contoh diatas, pemanggilan property $pemilik dan method hidupkan_laptop() dari luar class akan menghasilkan error.

Walaupun akses level protected tidak bisa diakses dari luar class, namun bisa diakses dari dalam class itu sendiri, berikut adalah contohnya:

```
<?php
// buat class laptop
class laptop {

  // buat protected property
  protected $pemilik="Anto";

  public function akses_pemilik() {
    return $this->pemilik;
  }
  protected function hidupkan_laptop() {
    return "Hidupkan Laptop";
  }
  public function paksa_hidup() {
    return $this->hidupkan_laptop();
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();
// jalankan method akses_pemilik()
echo $laptop_anto->akses_pemilik(); // "Anto"

// jalankan method paksa_hidup()
echo $laptop_anto->paksa_hidup(); // "Hidupkan Laptop"
?>
```

Hampir sama dengan contoh kita sebelumnya, property $pemilik di deklarasikan sebagai protected, sehingga pengaksesan dari luar class akan menghasilkan error. Oleh karena itu, kita membuat sebuah public method yang akan menampilkan hasil property $pemilik, yakni method akses_pemilik().

Begitu juga dengan method hidupkan_laptop() yang tidak bisa diakses secara langsung. kita menambahkan method paksa_hidup() yang secara internal akan mengakses method

hidupkan_laptop().

Kata kunci $this merujuk kepada objek saat ini. Namun karena satu-satunya cara pengaksesan 'isi' class dari dalam class adalah dengan $this.

Selain dari dalam class itu sendiri, property dan method dengan hak akses protected juga bisa diakses dari class turunan.

```
<?php
// buat class komputer
class komputer{
  // property dengan hak akses protected
  protected $jenis_processor = "Intel Core i7-4790 3.6Ghz";
}

// buat class laptop
class laptop extends komputer{
  public function tampilkan_processor() {
    return $this->jenis_processor;
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_baru = new laptop();

// jalankan method
echo $laptop_baru->tampilkan_processor(); // "Intel Core i7-4790 3.6Ghz"
?>
```

Pada kode diatas, walaupun method $jenis_processor di set sebagai protected pada class komputer, tetapi masih bisa diakses dari class laptop yang merupakan turunan dari class komputer.

Pengertian Hak Akses: Private

Hak akses terakhir dalam konsep enkapsulasi adalah private. Jika sebuah property atau method di-set sebagai private, maka satu-satunya yang bisa mengakses adalah class itu sendiri. Class lain tidak bisa mengaksesnya, termasuk class turunan.

Sebagai contoh, berikut adalah hasil yang di dapat jika kita mengakses property dan method dengan level private:

```
<?php
// buat class komputer
class komputer {

  // property dengan hak akses protected
```

```
  private $jenis_processor = "Intel Core i7-4790 3.6Ghz";

  public function tampilkan_processor() {
    return $this->jenis_processor;
  }
}

// buat class laptop
class laptop extends komputer{

  public function tampilkan_processor() {
    return $this->jenis_processor;
  }
}

// buat objek dari class laptop (instansiasi)
$komputer_baru = new komputer();
$laptop_baru = new laptop();

// jalankan method dari class komputer
echo $komputer_baru->tampilkan_processor(); // "Intel Core i7-4790 3.6Ghz"

// jalankan method dari class laptop (error)
echo $laptop_baru->tampilkan_processor();
// Notice: Undefined property: laptop::$jenis_processor
?>
```

Dalam kode diatas, kita membuat 2 buah class, yakni class komputer, dan class laptop. Class laptop merupakan turunan dari class komputer. Di dalam class komputer terdapat property $jenis_processor dengan akses level private. Di dalam class komputer dan class laptop, kita membuat method tampilkan_processor() yang digunakan untuk mengakses property $jenis_processor.

Pengaksesan method tampilkan_processor() dari objek $komputer_baru sukses ditampilkan karena berada di dalam satu class dimana property $jenis_processor berada.

Akan tetapi, jika method tampilkan_processor() diakses dari objek $laptop_baru yang merupakan turunan dari class komputer, PHP akan mengeluarkan error karena property $jenis_processor tidak dikenal.

Akses level private sering digunakan untuk menyembunyikan property dan method agar tidak bisa diakses di luar class.

6. Pengertian dan Fungsi Variabel $this dalam Pemrograman Objek

Variabel $this adalah sebuah variabel khusus dalam OOP PHP yang digunakan sebagai penunjuk kepada objek, ketika kita mengaksesnya dari dalam class. Dalam manual PHP, $this disebut dengan istilah: pseudo-variable.

Dalam contoh berikut, kita membuat class laptop dengan method yang saling memanggil method lain menggunakan variabel $this, silahkan anda pahami alur kerja dari class dan objek dibawah ini:

```
<?php
// buat class laptop
class laptop {

  // buat property untuk class laptop
  public $pemilik;
  public $merk;

  // buat method untuk class laptop
  public function hidupkan_laptop() {
    return "Hidupkan Laptop $this->merk punya $this->pemilik";
  }

  public function matikan_laptop() {
    return "Matikan Laptop $this->merk punya $this->pemilik";
  }

  public function restart_laptop() {
    $matikan=$this->matikan_laptop();
    $hidupkan= $this->hidupkan_laptop();
    $restart=$matikan."<br />".$hidupkan;
    return $restart;
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_anto = new laptop();

// isi property objek
$laptop_anto->pemilik="Anto";
$laptop_anto->merk="Asus";
echo $laptop_anto->hidupkan_laptop();
// hasil: "Hidupkan Laptop Asus punya Anto";

echo "<br />";

echo $laptop_anto->matikan_laptop();
// hasil: "Matikan Laptop Asus punya Anto";
```

```
echo "<br />";

echo $laptop_anto->restart_laptop();
// hasil:
// "Matikan Laptop Asus punya Anto";
// "Hidupkan Laptop Asus punya Anto";
?>
```

7. Cara Membuat Method dalam Pemrograman Objek PHP

Cara Membuat Method dengan Argumen/Parameter

Karena method pada dasarnya hanyalah function yang berada di dalam sebuah class, maka kita bisa memberikan argumen/parameter ke dalam method tersebut.
Langsung saja kita lihat struktur dasar pembuatan parameter di dalam method PHP:

```
hak_akses nama_method ($argumen1, argumen2, dst...)
{
  //... isi dari method
}
```

Dengan menggunakan contoh method hidupkan_laptop(), kita bisa membuatnya menjadi:

```
public hidupkan_laptop($pemilik, $merk)
{
  //... isi dari method
}
```

Sehingga apabila metod itu dipanggil dari objek, kita tinggal mengisi argumen dengan nilai yang diinginkan, seperti contoh berikut:

```
$laptop_andi('Andi','Lenovo');
```

Cara Membuat Argumen dalam Method Class

Sebagai contoh, kita akan kembali memodifikasi class laptop dengan menambahkan fitur argumen dalam method:

```
<?php
// buat class laptop
class laptop {
  // buat method untuk class laptop
  public function hidupkan_laptop($pemilik,$merk) {
    return "Hidupkan Laptop $merk punya $pemilik";
  }
```

```
}

// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop();

echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
// hasil: "Hidupkan Laptop Lenovo punya Andi";
?>
```

Dalam contoh diatas, kita memanggil method hidupkan_laptop() dengan 2 argumen, yakni “Andi” dan “Lenovo”. Kedua nilai ini akan diproses oleh method hidupkan_laptop().

Perhatikan bahwa kita tidak menggunakan variabel $this, karena argumen tersebut ‘milik’ method, perhatikan bedanya jika kita mengubah class laptop menjadi berikut ini:

```
<?php
// buat class laptop
class laptop {
  // buat property untuk class laptop
  private $pemilik="Anto";
  private $merk="Acer";

  // buat method untuk class laptop
  public function hidupkan_laptop($pemilik,$merk) {
    return "Hidupkan Laptop $merk punya $pemilik";
  }

  public function hidupkan_laptop_anto() {
    return "Hidupkan Laptop $this->merk punya $this->pemilik";
  }
}
// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop();
echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
// hasil: "Hidupkan Laptop Lenovo punya Andi";

echo $laptop_andi->hidupkan_laptop_anto();
// hasil: "Hidupkan Laptop Acer punya Anto";
?>
```

Pada class laptop diatas, kita menambahkan 2 property: $pemilik dan $merk, kemudian memberikan nilai “Anto” dan “Acer”. Jika yang kita inginkan adalah nilai dari variabel ini, maka di dalam method, kita harus menggunakan $this.

Semua fitur function, juga bisa diterapkan di dalam method, termasuk default parameter

seperti contoh berikut:

```
<?php
// buat class laptop
class laptop {
  // buat method untuk class laptop
  public function hidupkan_laptop($pemilik="Joko",$merk="Samsung") {
    return "Hidupkan Laptop $merk punya $pemilik";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop();

echo $laptop_andi->hidupkan_laptop();
// hasil: "Hidupkan Laptop Samsung punya Joko";

echo "<br />";

echo $laptop_andi->hidupkan_laptop("Andi", "Lenovo");
// hasil: "Hidupkan Laptop Lenovo punya Andi";
?>
```

Pada contoh kode program PHP diatas, dengan membuat method sebagai berikut:

```
public function hidupkan_laptop($pemilik="Joko",$merk="Samsung")
```

Maka, ketika method tersebut dipanggil tanpa menambahkan argumen, nilai “Joko” dan “Samsung” akan digunakan sebagai nilai default, namun jika argumen ditulis, nilai argumen yang diinput akan menimpa nilai default ini.

8. Pengertian Constructor dan Destructor

Pengertian Constructor dalam OOP

Constructor (bahasa indonesia: konstruktor) adalah method khusus yang akan dijalankan secara otomatis pada saat sebuah objek dibuat (instansiasi), yakni ketika perintah “new” dijalankan.

Constructor biasa digunakan untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai awal kepada property, memanggil method internal dan beberapa proses lain yang digunakan untuk ‘mempersiapkan’ objek.

Dalam PHP, constructor dibuat menggunakan method : __construct().

Pengertian Destructor dalam OOP

Destructor (bahasa indonesia: destruktor) adalah method khusus yang dijalankan secara

otomatis pada saat sebuah objek dihapus. Di dalam PHP, seluruh objek secara otomatis dihapus ketika halaman PHP dimana objek itu berada selesai diproses. Tetapi kita juga dapat menghapus objek secara manual.

Destructor biasanya digunakan untuk 'membersihkan' beberapa variabel, atau menjalankan proses tertentu sebelum objek dihapus. Dalam PHP, destructor dibuat menggunakan method : __destruct().

PHP memiliki fitur 'penghapusan objek massal' yang dikenal dengan Garbage Collection. Fitur garbage collection akan menghapus seluruh objek secara otomatis ketika halaman PHP selesai di eksekusi, sehingga akan memanggil method __destructor dari seluruh objek.

Namun PHP menyediakan cara jika kita ingin menghapus objek sebelum mekanisme ini berjalan, yaitu menggunakan fungsi unset() atau mengisi variabel objek dengan nilai null. Kita akan melihat contoh penggunaannya dengan contoh program dibawah.

Cara Penggunaan Destructor dan Constructor dalam PHP

Berikut adalah contoh penggunaan constructor dan destructor dalam PHP:

```
<?php
// buat class laptop
class laptop {

  private $pemilik = "Andi";
  private $merk = "Lenovo";

  public function __construct(){
    echo "Ini berasal dari Constructor Laptop";
  }

  public function hidupkan_laptop(){
    return "Hidupkan Laptop $this->merk punya $this->pemilik";

  }
  public function __destruct(){
    echo "Ini berasal dari Destructor Laptop";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop();

echo "<br />";
echo $laptop_andi->hidupkan_laptop();
echo "<br />";
?>
```

Dalam contoh diatas, kita membuat class laptop dengan 3 method:

a) Method __construct() merupakan constructor dari class laptop. Method ini akan dipanggil secara otomatis ketika class laptop di instansiasi.
b) Method hidupkan_laptop() merupakan method 'biasa' yang akan menampilkan hasil string. Untuk menggunakan method ini, kita memanggilnya dari objek.
c) Method ketiga adalah __destruct() yang merupakan destructor dari class laptop. Method ini akan dipanggil saat objek dihapus.

Setelah pendefinisian class, kita membuat objek $laptop_andi, dan memanggil method hidupkan_laptop(). Berikut adalah hasil yang didapat:

Ini berasal dari Constructor Laptop
Hidupkan Laptop Lenovo punya Andi
Ini berasal dari destructor Laptop

Seperti yang terlihat, method __construct() dan __destruct() secara otomatis dipanggil saat objek dibuat dan saat objek dihapus. Untuk mencoba menghapus objek $laptop_andi secara manual, kita bisa menggunakan fungsi isset() sebagai berikut:

```
<?php
// buat class laptop
class laptop {

  private $pemilik = "Andi";
  private $merk = "Lenovo";

  public function __construct(){
    echo "Ini berasal dari Constructor Laptop";
  }

  public function hidupkan_laptop(){
    return "Hidupkan Laptop $this->merk punya $this->pemilik";

  }
  public function __destruct(){
    echo "Ini berasal dari Destructor Laptop";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop();

echo "<br />";
```

```
echo $laptop_andi->hidupkan_laptop();
echo "<br />";

// hapus objek $laptop_andi
unset($laptop_andi);

echo "<br />";
echo "Objek Telah Dihancurkan";
?>
```

Jika kita menjalankan kode diatas, berikut adalah hasil yang didapat:

```
Ini berasal dari Constructor Laptop
Hidupkan Laptop Lenovo punya Andi
Ini berasal dari Destructor Laptop
Objek Telah Dihancurkan
```

Setelah memanggil method $laptop_andi->hidupkan_laptop(), kita kemudian menghapus objek $laptop_andi secara manual menggunakan fungsi unset($laptop_andi).

Untuk membuktikan bahwa destructor $laptop_andi sudah dijalankan, kita menambahkan perintah echo “Objek Telah Dihancurkan” diakhir halaman. Sehingga kita bisa melihat bahwa destructor objek $laptop andi di jalankan sebelum dihapus otomatis oleh PHP. Silahkan anda coba hapus perintah unset($laptop_andi), maka string “Objek Telah Dihancurkan” akan tampil sebelum destructor objek $laptop_andi.

Anda juga bisa mengganti perintah unset($laptop_andi) dengan $laptop_andi=null. Kedua perintah ini akan menghapus objek $laptop_andi dan men-trigger pemanggilan destructor.

Contoh constructor yang sering digunakan untuk membuat objek dengan nilai awal. Konsep ini sering digunakan dalam pemrograman objek. Berikut adalah contoh penggunaannya:

```
<?php
// buat class laptop
class laptop {

  private $pemilik;
  private $merk;

  // constructor sebagai pembuat nilai awal
  public function __construct($pemilik, $merk) {
    $this->pemilik = $pemilik;
    $this->merk = $merk;
```

```
  }

  public function hidupkan_laptop() {
    return "Hidupkan Laptop $this->merk punya $this->pemilik";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_andi= new laptop("Andi", "Lenovo");

echo $laptop_andi->hidupkan_laptop();
echo "<br />";

$laptop_anto= new laptop("Anto", "Acer");

echo $laptop_anto->hidupkan_laptop();
?>
```

Pada kode diatas, kita menggunakan constructor sebagai pembuat nilai awal dari objek. Method constructor menerima 2 buah argumen yang kemudian disimpan kedalam property internal objek.
Berikut adalah hasil yang didapat dari kode program diatas:

```
Hidupkan Laptop Lenovo punya Andi
Hidupkan Laptop Acer punya Anto
```

Di dalam PHP, constructor dan destructor harus memiliki hak akses public. Jika anda mengubah hak akses constructor atau destructor menjadi protected atau private, PHP akan mengeluarkan error:

```
Fatal error: Call to private laptop::__construct()
from invalid context
```

9. Pengertian Inheritance (Pewarisan)

Inheritance atau Pewarisan/Penurunan adalah konsep pemrograman dimana sebuah class dapat ‘menurunkan’ property dan method yang dimilikinya kepada class lain. Konsep inheritance digunakan untuk memanfaatkan fitur ‘code reuse’ untuk menghindari duplikasi kode program.

Konsep inheritance membuat sebuah struktur atau ‘hierarchy’ class dalam kode program. Class yang akan ‘diturunkan’ bisa disebut sebagai class induk (parent class), super class, atau base class. Sedangkan class yang ‘menerima penurunan’ bisa disebut sebagai class anak (child class), sub class, derived class atauheir class.

Tidak semua property dan method dari class induk akan diturunkan. Property dan method

dengan hak akses private, tidak akan diturunkan kepada class anak. Hanya property dan method dengan hak akses protected dan public saja yang bisa diakses dari class anak.

Cara Penggunaan Inheritance dalam PHP

Di dalam PHP, inheritance / penurunan dari sebuah class kepada class lain menggunakan kata kunci: 'extends', dengan penulisan dasar sebagai berikut:

```
class induk {
  //...isi class induk
}

class anak extends induk
{
  //... class anak bisa mengakses
  //... property dan method class induk
}
```

Agar lebih mudah dipahami, kita akan langsung masuk kedalam contoh program penggunaan inheritance/penurunan di dalam PHP:

```
<?php
// buat class induk: komputer
class komputer {

  public $merk;
  public $processor;
  public $memory;

  public function beli_komputer() {
    return "Beli komputer baru";
  }
}

// turunkan class komputer ke laptop
class laptop extends komputer {

  public function lihat_spec() {
    return "merk: $this->merk, processor: $this->processor,
    memory: $this->memory";
  }
}

// buat objek dari class laptop (instansiasi)
$laptop_baru = new laptop();
```

```
// isi property objek
$laptop_baru->merk = "acer";
$laptop_baru->processor ="intel core i5";
$laptop_baru->memory = "2 GB";

//panggil method objek
echo $laptop_baru->beli_komputer();
echo "<br />";
echo $laptop_baru->lihat_spec();
?>
```

Dalam contoh kode diatas, kita membuat class komputer dengan beberapa property dan sebuah method. Property class komputer belum berisi nilai apa-apa.

Dibawah class komputer, kita membuat class laptop extends class komputer. Disini kita menurunkan class komputer kedalam class laptop. Di dalam class laptop, kita bisa mengakses seluruh property dan method apapun dari class komputer selama memiliki hak akses public atau protected.

Untuk membuktikan hal tersebut, kita membuat objek $laptop_baru dari class laptop. Perhatikan bahwa kita bisa mengakses property $merk, $processor, dan$memory yang semuanya adalah milik class komputer, bukan class laptop. Method beli_komputer() juga sukses diakses dari objek $laptop baru. Inilah yang dimaksud dengan inheritance/penurunan class dalam OOP.

PHP tidak membatasi berapa banyak ‘penurunan objek’ yang bisa dilakukan, dalam contoh berikut, kita membuat 3 buah class yang saling ‘menurunkan’:

```
<?php
// buat class komputer
class komputer {
  protected function beli_komputer() {
    return "Beli komputer baru";
  }
}

// turunkan class komputer ke laptop
class laptop extends komputer {
  protected function beli_laptop() {
    return "Beli laptop baru";
  }
}

// turunkan class laptop ke chromebook
class chromebook extends laptop {
```

```
  protected function beli_chromebook() {
    return "Beli chromebook baru";
  }

  public function beli_semua(){
    $a = $this->beli_komputer();
    $b = $this->beli_laptop();
    $c = $this->beli_chromebook();
    return "$a <br /> $b <br /> $c";
  }
}

// buat objek dari class laptop (instansiasi)
$gadget_baru = new chromebook();

//panggil method objek
echo $gadget_baru->beli_semua();

// $gadget_baru->beli_komputer();
// Fatal error: Call to protected method komputer::beli_komputer()
?>
```

Dalam contoh diatas, kita membuat class komputer yang diturunkan kepada class laptop, dan kemudian diturunkan lagi kepada class chromebook. Dari dalam class chromebook ini kemudian kita memanggil method dari class diatasnya.

Jika anda perhatikan, setiap method selain method beli_semua(), memiliki hak akses protected. Hak akses protected ini 'menghalangi' kode program lain untuk mengaksesnya, selain class turunan.

Pada baris terakhir, kita menyisipkan kode program untuk mencoba mengakses method beli_komputer() . Kode ini sengaja kita beri tanda komentar. Jika anda menghapus tanda komentar, PHP akan mengeluarkan error yang menyatakan kita tidak bisa mengakses method dengan hak akses protected:

```
<?
$gadget_baru->beli_komputer();
// Fatal error: Call to protected method komputer::beli_komputer()
?>
```

Inilah yang dimaksud dengan enkapsulasi dalam OOP. Membatasi method yang tidak boleh diakses akan membuat kode program menjadi lebih terstruktur.

10. Cara Mengakses Property dan Method Parent Class

Konsep pewarisan/inheritance dimana sebuah class bisa memiliki property dan method

dari class lain, bisa menjadi permasalahan ketika property atau method dari class anak memiliki nama yang sama dengan class induk, atau dikenal dengan istilah overridden property dan overridden method.

Untuk memahami pengertian overridden property dan overridden method, perhatikan contoh kode program berikut ini:

```
<?php
// buat class komputer
class komputer {

  public function lihat_spec() {
    return "Spec Komputer: Acer,
    Processor Intel core i7, Ram 4GB";
  }
}

// turunkan class komputer ke laptop
class laptop extends komputer {

  public function lihat_spec() {
    return "Spec Laptop: Asus,
    Processor Intel core i5, Ram 2GB";
  }
}
// buat objek dari class laptop (instansiasi)
$gadget_baru = new laptop();

//panggil method lihat_spec()
echo $gadget_baru->lihat_spec();
?>
```

Pada kode program diatas, kita membuat 2 buah class: komputer dan laptop. kita menurunkan class komputer kedalam class laptop, sehingga seluruh property dan method dari class komputer bisa diakses dari class laptop.

Namun perhatikan bahwa method pada class komputer memiliki nama yang sama dengan method dalam class laptop. Ketika kita memanggil method lihat_spec(), method manakah yang akan dijalankan?
Jika anda menjalankan kode diatas, maka hasilnya adalah:

```
Spec Laptop: Asus, Processor Intel core i5, Ram 2GB
```

Berdasarkan hasil yang di dapat, terlihat bahwa method yang dijalankan adalah method milik class laptop.

Di dalam PHP, ketika nama property atau nama method child class memiliki nama yang sama dengan parent class, maka yang dijalankan adalah property atau method milik child class.

Jadi, bagaimana cara mengakses property dan method milik class komputer? PHP mengatasi hal ini dengan menggunakan 'Scope Resolution Operator'.

Pengertian Scope Resolution Operator PHP

Scope Resolution Operator adalah operator khusus di dalam PHP yang memungkinkan kita untuk mengakses 'informasi khusus' dari dalam class.

Informasi khusus ini terdiri dari: overridden property atau overridden method, static property atau static method, serta constanta class. Untuk saat ini, kita akan fokus kepada overridden property atau overridden method.

Scope Resolution Operator ditulis dengan tanda dua kali titik dua (double colon), yakni "::". Untuk mengakses property dan method dari class induk, kita mengaksesnya dengan perintah:

```
parent::nama_property;
parent::nama_method();
```

Kembali kepada contoh program, kali ini kita ingin menampilkan method lihat_spec() dari class komputer:

```
<?php
// buat class komputer
class komputer {

  public function lihat_spec() {
    return "Spec Komputer: Acer,
    Processor Intel core i7, Ram 4GB";
  }
}

// turunkan class komputer ke laptop
class laptop extends komputer {

  public function lihat_spec() {
    return "Spec Laptop: Asus,
    Processor Intel core i5, Ram 2GB";
  }

  public function lihat_spec_komputer() {
    return parent::lihat_spec();
```

```
  }
}
// buat objek dari class laptop (instansiasi)
$gadget_baru = new laptop();

//panggil method lihat_spec()
echo $gadget_baru->lihat_spec();

echo "<br />";

//panggil method lihat_spec_komputer()
echo $gadget_baru->lihat_spec_komputer();
?>
```

Kode program diatas adalah revisi dari contoh kita sebelumnya. kita menambahkan sebuah method lihat_spec_komputer() kedalam class laptop. Method ini selanjutkan akan memanggil method class komputer, dengan perintah parent::lihat_spec().

Hasilnya adalah:

```
Spec Laptop: Asus, Processor Intel core i5, Ram 2GB
Spec Komputer: Acer, Processor Intel core i7, Ram 4GB
```

Contoh diatas adalah cara mengakses method parent class dari child class.

11. Form dengan OOP PHP

Nama File : Form.php
Deskripsi : Program class untuk membuat sebuah form inputan sederhana.

```
<?php
class Form
{ var $fields = array();
  var $action;
  var $submit = "Submit Form";
  var $jumField = 0;

  public function __construct($action, $submit)
  {   $this->action = $action;
      $this->submit = $submit;
  }

  public function displayForm()
  {   echo "<form action='".$this->action."' method='POST'>";
```

```
    echo "<table width='100%'>";
    for ($j=0; $j<count($this->fields); $j++) {
       echo "<tr><td align='right'>".$this->fields[$j]['label']."</td>";
       echo"<td><input type='text' name='".$this->fields[$j]['name']."'></td></tr>";
    }
    echo "<tr><td colspan='2'>";
    echo "<input type='submit' value='".$this->submit."'></td></tr>";
    echo "</table>";
  }

  public function addField($name, $label)
  { $this->fields [$this->jumField]['name'] = $name;
    $this->fields [$this->jumField]['label'] = $label;
    $this->jumField ++;
  }
}
```

Nama File : view-form.php
Deskripsi : Memanfaatkan class-form.php untuk membuat form inputan sederhana.

```
<?php
include "Form.php";
echo "<html><head><title>Mahasiswa</title></head><body>";
$form = new Form ("","Input Form");
$form->addField ("txtnim", "Nim");
$form->addField ("txtnama", "Nama");
$form->addField ("txtalamat", "Alamat");
echo "<h3>Silahkan isi form berikut ini :</h3>";
$form->displayForm();
echo "</body></html>";
?>
```