Pertemuan 4 PBO

Pemrograman Berbasis Object

Pengertian Pemrograman Berbasis Object

"Pemrograman berbasis objek adalah sebuah tata cara pembuatan program (programming paradigm) dengan menggunakan konsep “objek”, dimana objek ini bisa memiliki data (dikenal dengan istilah atribut) dan kode program dalam bentuk prosedur (dikenal dengan istilah method)."
Secara sederhana, **pemrograman berbasis object** adalah sebuah cara penulisan kode program dengan menggunakan **object** untuk memecahkan masalah. Dalam teori pemrograman, terdapat 3 prinsip dasar yang melandasi pemrograman berbasis object, yakni *encapsulation*, *inheritance* dan *polymorphism*.

Kenapa Harus Pemrograman Berbasis Object?

Untuk bisa menjawab pertanyaan ini, kita harus cari pembandingnya. Selain pemrograman berbasis object, terdapat cara atau "paradigma" lain untuk membuat kode program, yakni **pemrograman prosedural**.
Pemrograman prosedural (*procedural programming*), atau kadang disebut juga sebagai pemrograman fungsional (*functional programming*), adalah cara penulisan kode program yang dipecah ke dalam function-function. Secara umum, pemrograman prosedural lebih sederhana jika dibandingkan dengan pemrograman berbasis object. Dalam pemrograman prosedural, kode program ditulis secara berurutan dari baris paling atas sampai baris paling bawah. Jika kode atau masalah yang dihadapi cukup panjang, kita bisa pecah menjadi beberapa function yang kemudian di satukan kembali di dalam kode program utama.

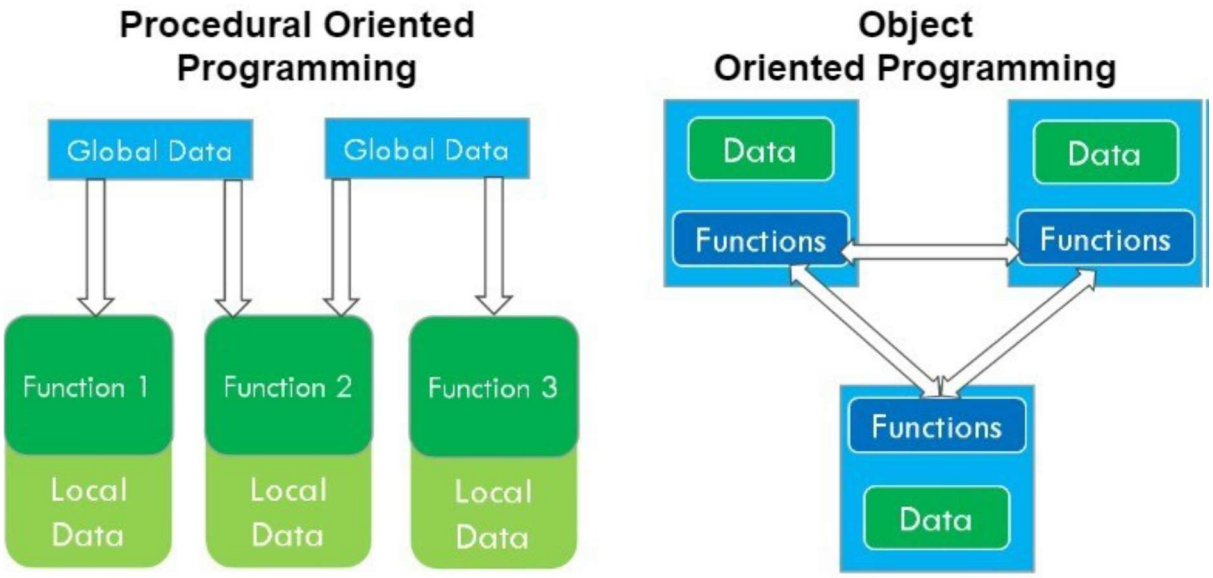
Tidak ada yang salah dari pemrograman prosedural. Selain praktis, cara pembuatan kode program seperti ini juga sederhana dan mudah dipelajari. Namun seiring kompleksitas aplikasi yang dibuat, pada titik tertentu pemrograman prosedural memiliki beberapa keterbatasan.

Pertama, tidak ada mekanisme pengelompokkan function. Semua function bisa diakses dari mana saja dan memiliki nama bebas. Jika kode program ini dibuat oleh tim, sangat mungkin seorang anggota tim membuat nama function yang sama persis dengan programmer lain.

Kedua, pemrograman prosedural berfokus ke alur program secara linear (berurutan). Ini membuatnya susah jika suatu saat terdapat perubahan kode program. Satu perubahan di awal akan berdampak ke bagian yang lain.

Ketiga, kode program "terlalu melekat" dengan masalah yang dipecahkan saat ini, sehingga tidak bisa dipakai untuk masalah lain. Sebagai contoh, jika kita memiliki form login, form register dan form edit, agak susah membuat sebuah mekanisme agar ketiganya bisa diproses dengan kode program yang sama.

Masalah inilah yang bisa diatasi dengan lebih mudah jika menggunakan pemrograman berbasis object atau OOP.
Di dalam OOP, kita bisa memakai perumpamaan objek di dunia nyata, seperti object *user*, object *produk*, atau object *gambar*, dsb. Setiap object ini punya "dunianya" masing-masing sehingga saling terpisah satu sama lain. Dengan demikian, perubahan di satu bagian tidak akan berdampak langsung ke bagian lain.



Kekurangan OOP

Pertama, kita perlu "perencanaan" sebelum membuat kode program. Perencanaan yang dimaksud adalah object apa saja yang nantinya harus dibuat, lalu bagaimana hubungan satu object dengan object lain. Dalam teori programming, perancangan konsep OOP ini biasanya menggunakan diagram **UML** (Unified Modeling Language) atau **Class Diagram**. Perencanaan inilah yang membuat OOP menjadi *modular,* mudah dikembangkan dan dikelola. Di awal kita sudah harus memikirkan seperti apa alur kode program secara keseluruhan, kemudian bagaimana mengantisipasi jika terdapat perubahan atau butuh penambahan fitur baru.

Kelemahan **kedua** adalah perubahan pola pikir (mindset) agar bisa membuat kode program yang benar-benar menerapkan prinsip pemrograman object, dan ini tidak mudah. Dalam banyak kasus, kita lebih sering menggabungkan konsep pemrograman prosedural dengan pemrograman object. Pemrograman object biasa dipakai untuk mengakses library atau object bawaan PHP (yang berisi fungsi-fungsi tertentu), namun program utama tetap dibuat dalam bentuk prosedural.

Kelemahan **ketiga** dari OOP adalah, untuk website sederhana, kode program yang diperlukan akan lebih panjang daripada pemrograman prosedural. Namun ini sebenarnya sebagai kompensasi dari keunggulan yang ditawarkan oleh OOP.

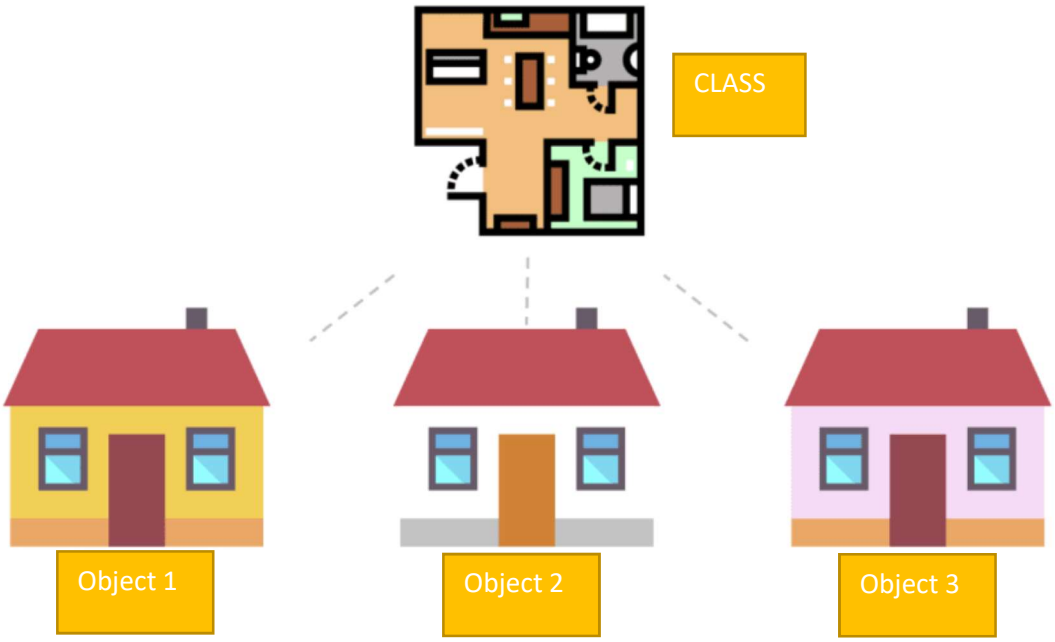
Basic OOP PHP

Class dan Object

Class dan **object** merupakan fondasi paling dasar dari *object oriented programming*, keduanya serupa tapi tak sama. Class adalah *blueprint* atau "cetakan" untuk object. Bisa disebut juga bahwa object adalah implementasi konkret dari sebuah class.

Sebagai analogi, ibaratnya kita ingin membuat object rumah. **Class** adalah gambar desain

rumah yang dirancang oleh arsitek. Melalui gambar design ini, nantinya bisa dibuat tidak hanya 1 rumah, tapi bisa banyak rumah (seperti yang ada di kompleks perumahan). Di sini, ***object rumah adalah implementasi konkret dari class gambar rumah***.



Pemrograman berbasis object mencoba menjadikan object dunia nyata ke dalam konsep pemrograman. Misalnya jika nanti kita membuat kode untuk pemrosesan user (login, register, logout, dst), maka akan ada class **User**. Jika kita ingin membuat website jual beli, nanti akan ada class **Produk**, **Invoice**, dst.

Untuk membuat class di dalam PHP, caranya adalah dengan menulis *keyword* **class** kemudian diikuti dengan nama class. Isi class nantinya ada di dalam tanda kurung kurawal. Berikut contoh pembuatan class **Produk** di dalam PHP, simpan dengan nama **class.php** pada folder baru **pertemuan_4**:

```php
<?php
class Produk {

}
```

Dengan program diatas kita sudah berhasil membuat sebuah class basic dengan nama **Produk,**

Catatan:
Kebiasaan banyak programmer PHP (dan juga di sebagian besar bahasa pemrograman lain), adalah menggunakan huruf besar pada karakter pertama nama class. Daripada membuat class `produk`, lebih baik menggunakan `class Produk`. Jika nama class tersebut lebih dari 2 kata, gunakan penulisan seperti `class ProdukTelevisi` atau `class UserAdmin`.

Dari penjelasan sebelumnya disebutkan bahwa class hanya kerangka kerja atau *prototype*. Kita tidak akan mengakses class secara langsung, tapi harus melalui object. Object adalah **bentuk konkret dari class**

membuat sebuah object.

Untuk membuat object dari suatu class, gunakan keyword new seperti contoh berikut, simpan dengan nama **object.php**:

```php
<?php
class Produk {

}

$televisi = new Produk();       //object 1
$buku = new Produk();           //object 2
$smartphone = new Produk();     //object 3
$speaker = new Produk();        //object 4 dan seterusnya
```

Program diatas artinya kita telah membuat 4 buah object dari class Produk. Lihat strukturnya, object berada diluar class atau diluar tanda kurung kurawal. Jumlah object yang dibuat bisa tak terbatas sesuai kebutuhan.

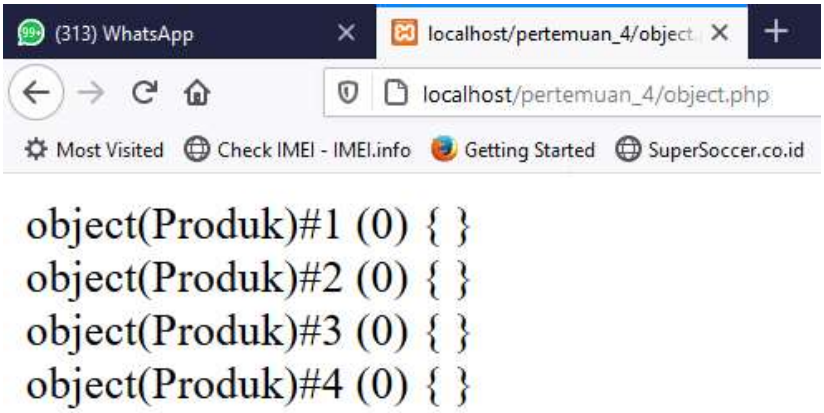
Catatan:
Proses pembuatan object ini dikenal dengan istilah *instansiasi object* atau *object instantiation*. !!!

Bila program diatas dipanggil melalui browser maka tidak akan menghasilkan sesuatu apapun meskipun tidak error, karena memang belum ada method yang menjalankan fungsi apapun pada class maupun object.

Ok agar ada sedikit gambaran tentang object yang telah kita buat, kita akan menambahkan fungsi **var_dump()** untuk melihat detail dari variable yang kita buat. Lengkapi program sebelumnya atau boleh buat baru.

```php
<?php
class Produk {

}

$televisi = new Produk();
$buku = new Produk();
$smartphone = new Produk();
$speaker = new Produk();

var_dump($televisi);    //  object(Produk)#1 (0) { }
echo "<br>";
var_dump($buku);    //  object(Produk)#2 (0) { }
echo "<br>";
var_dump($smartphone);    //  object(Produk)#3 (0) { }
echo "<br>";
var_dump($speaker);    //  object(Produk)#4 (0) { }
```

apabila program diatas kembali dipanggil melalui browser maka aka nada output yang ditampilkan.



Catatan:

Hasil perintah `var_dump($televisi)` bisa dibaca bahwa `$televisi` adalah **object** dari **class** `Produk`. Angka `#1` menandakan ini adalah object `Produk` pertama yang ada di dalam file PHP. Angka (0) menandakan jumlah *property* yang ada di dalam object (akan kita bahas sesaat lagi).

PAHAM???

Property dan Method

Property dan **method** tidak lain adalah sebutan untuk *variabel* dan *function* yang berada di dalam class. Cara penulisannya pun sama seperti variabel dan function, tapi dengan tambahan *access modifier* di awal penulisan seperti contoh berikut, simpan dengan nama **property_method.php**:

```php
<?php
class Produk { //class dengan 3 property dan 1 fungsi
  public $id = "001"; //property class harus mempunyai access modifier
  public $merek = "Samsung";
  public $harga = 4000000;

  public function pesanProduk(){
    return "Produk dipesan...";
  }
}
```

Program diatas hanya melengkapi class yang telah kita buat sebelumnya, dengan menambahkan property pada class, ada 3 property yang ditambahkan yaitu $id, $merek,

$harga, dengan access modifier atau visibility **public (akan dijelaskan selanjutnya).** Property yang dibuat juga telah diisi langsung dengan sebuah contoh nilai.

Selain menambahkan property kita juga menambahkan sebuah method/function di dalam class dengan nama **pesanProduk(), dimana fungsi sederhana ini akan mengembalikan nilai (return) berupa teks Produk dipesan...**

Catatan:

Cara penulisan sebuah fungsi diawalai dengan nama fungsi diikuti tanda kurung buka dan kurung tutup contoh: pesanProduk().

Setelah itu untuk menuliskan statement dari fungsi diawali dan diakhiri dengan tanda kurung kurawal, contoh

```
public function pesanProduk(){
    return "Produk dipesan...";
  }
```

Ini adalah bentuk paling sederhana dari sebuah fungsi, untuk bentuk lain akan kita pelajari selanjutnya.

Paham???

Cara Mengakses Property dan Method

Bagaimana cara mengakses property dan method? **Ingat bahwa dalam pemrograman berbasis object, yang akan kita akses adalah object, bukan class**. Artinya apa??? Agar kita bisa mengakses property dan method dari sebuah class maka kita harus membuat object dari class tersebut (sudah kita lakukan sebelumnya cara membuat sebuah object). Kita akan membuat sebuah contoh object untuk melengkapi latihan kita.

```
<?php
class Produk {
  public $id = "001";
  public $merek = "Samsung";
  public $harga = 4000000;

  public function pesanProduk(){
    return "Produk dipesan...";
  }
}

$televisi = new Produk(); //buat object dari class produk

echo $televisi->id;// 001 (object televisi mengakses atribut id dari class)
echo "<br>";
echo $televisi->merek; // Samsung
echo "<br>";
echo $televisi->harga; // 4000000
echo "<br>";
```

```
echo $televisi->pesanProduk();
// Produk dipesan... (object televisi mengakses fungsi dari class)
```

program diatas menjelaskan, setelah kita memiliki sebuah class lengkap dengan attribute dan method maka untuk dapat mengakses attribute dan method tersebut kita harus membuat sebuah object pada contoh di atas adalah televisi.

Karena televisi merupakan object dari class produk maka secara otomatis televisi mewarisi seluruh property dan method yang dimiliki oleh class Produk.

Sehingga pada saat kita ingin menampilkan isi dari property dari televisi caranya adalah

```
echo $televisi->id;
```

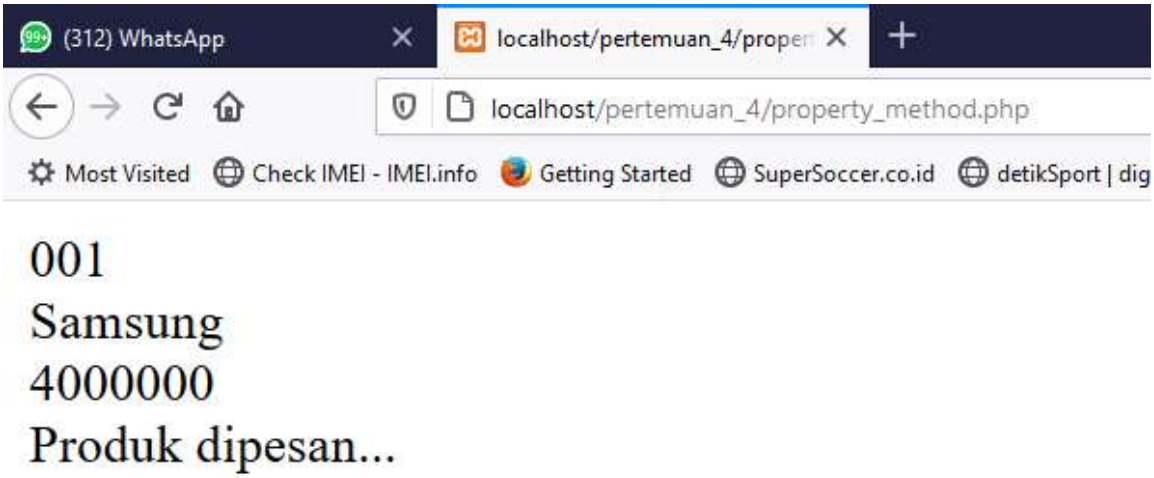
artinya object televisi mengakses property id. Cara yang sama untuk mengkases property yang lain. Fungsi echo agar isi dari attribute dapat tampil pada browser.

Sedangkan untuk mengakses fungsi dilakukan cara yang sama yaitu

```
echo $televisi->pesanProduk();
```

artinya object televise mengakses fungsi pesanProduk(), lihat perbedaan saat mengakses property. Saat mengakses fungsi ciri cirinya adalah ada tanda kurung buka dan kurung tutup.

Apabila kita akses melalui browser akan memberikan hasil sebagai berikut:



SILAKAN DIPAHAMI DULU…!!!

MEMBUAT 2 OBJECT DARI CLASS

Kali ini kita akan mencoba membuat 2 buah object berbeda dari class Produk, kita akan buat object televisi dan mesinCuci.

Berikut kode programnya, simpan dengan nama **object_lanjut.php**

```php
<?php
class Produk {
  public $id = "000"; // nilai awal bebas
  public $merek = ""; //boleh berupa string kosong
  public $harga = 0; //boleh berupa angka 0 untuk bilangan

  public function pesanProduk(){
    return "Produk dipesan...";
  }
}

$televisi = new Produk(); //object pertama
$televisi->id = "001";
$televisi->merek = "LG";
$televisi->harga = 1500000;

$mesinCuci = new Produk(); //object kedua
$mesinCuci->id = "002";
$mesinCuci->merek = "Samsung";
$mesinCuci->harga = 1500000;

print_r ($televisi); // print_r() fungsi untuk mencetak detail object
echo "<br>";
print_r ($mesinCuci);
echo "<br>";
```

perbedaan dengan program sebelumnya adalah pada program ini kita memberikan nilai dari masing-masing property setelah kita membuat object nya, sedikit berbeda dengan program sebelumnya, silakan dibandingkan. **Nilai awal property pada saat definisi class akan ditimpa dengan nilai yang baru.**

APAKAH KALIAN PAHAM DENGAN OUTPUT PROGRAM DIATAS???

Yup baris pertama menampilkan detail dari object Televisi dan baris kedua menampilkan detail dari object mesinCuci.

Fungsi print_r() dapat menampilkan detail property beserta nilainya dari sebuah object
Catatan:
Pahami secara detail dan pelan-pelan apa yang sudah kalian lakukan diatas. Bagaimana cara membuat kelas, bagaimana cara mendefinisikan property class dan method class, bagaimana mengisikan sebuah nilai pada property.

Ingat pahami jangan asal program jalan lalu ditinggalkan!!!

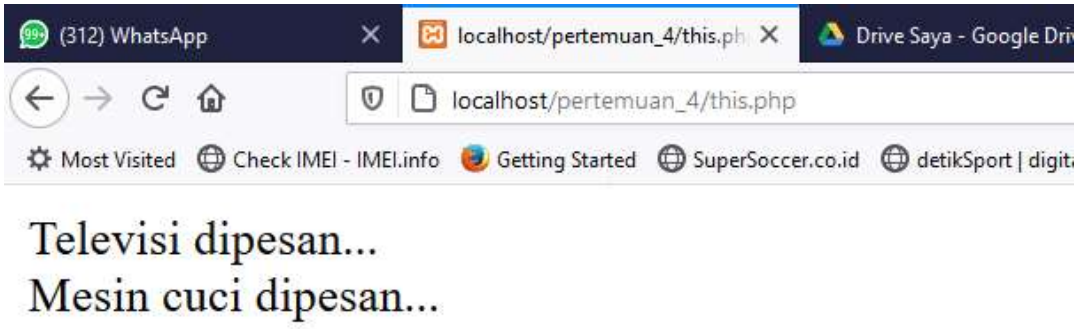
Pseudo-variable $this

Kali ini kita akan membahas sebuah konsep yang sering membuat bingung tapi sangat sering digunakan dalam konsep OOP, yakni tentang **pseudo-variable `$this`**. Sebelum ke sana, silahkan pelajari sebentar kode program berikut ini, simpan dengan nama **this.php**:

```php
<?php
class Produk {
  public $jenis = "";
  public $merek = "";

  public function pesanProdukTelevisi(){
    return "Televisi dipesan...";
  }

  public function pesanProdukMesinCuci(){
    return "Mesin cuci dipesan...";
  }
}

$produk01 = new Produk();
$produk01->jenis = "Televisi";
$produk01->merek = "Samsung";

$produk02 = new Produk();
$produk02->jenis = "Mesin cuci";
$produk02->merek = "LG";

echo $produk01->pesanProdukTelevisi();    // Televisi dipesan...
echo "<br>";
echo $produk02->pesanProdukMesinCuci();   // Mesin cuci dipesan...
```

langsung pada perbedaan dengan program kita sebelumnya, class produk ini memiliki 2 fungsi masing-masing untuk **pesanProdukTelevisi()** dan **pesanProdukMesinCuci()** sesuai dengan produk yang rencananya akan dibuat yaitu **produk01** dan **produk02**
sampai sini masih aman ya.

Ok sekarang coba akses program diatas melalui browser!!



Apakah ada yang salah dengan program diatas??? Tentu tidak, program dapat berjalan dengan baik, menampilkan output sesuai yang diinginkan..

LALU MASALAHNYA DIMANA???

Masalahnya adalah dengan struktur class seperti diatas maka class Produk **menjadi tidak fleksible,** maksudnya????

Bayangkan kalian akan membuat 100 produk dan masing2 produk memili fungsi pesanProduk() sesuai dengan 2 fungsi sebelumnya, artinya dalam class produk kita perlu membuat 100 fungsi pesanProduk juga sesuai dengan produk yang dibuat. Dan ini sangat merepotkan serta tidak fleksible, bukan seperti itu cara kerja OOP.

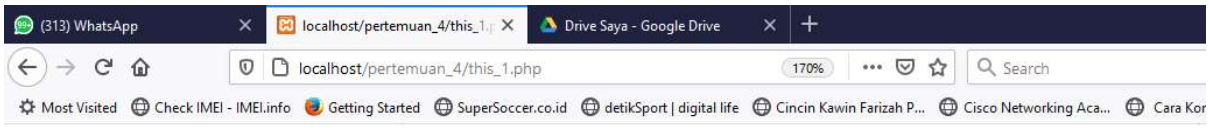
Solusinya???
Ada cara yang lebih baik, dari pada membuat 1 method untuk setiap tipe produk, kita bisa rancang method "generik" bernama `pesanProduk()` dengan konsep sebagai berikut simpan dengan nama **this_1.php**:

```php
<?php
class Produk {
  public $jenis = "";
  public $merek = "";

  public function pesanProduk(){
    return $jenis." dipesan...";
  }
}

$produk01 = new Produk();
$produk01->jenis = "Televisi";
$produk01->merek = "Samsung";

echo $produk01->pesanProduk();
```

program diatas mirip dengan sebelumnya hanya saja kita memodifikasi bagian fungsi dengan membuat sebuah fungsi generic **pesanProduk()** yang nantinya dapat digunakan oleh seluruh produk yang dibuat.

Pada contoh diatas kita membuat 1 buah produk baru produk01.

Ok sekarang kalian coba jalankan programnya dan lihat hasilnya



Notice: Undefined variable: jenis in **C:\xampp\htdocs\pertemuan_4\this_1.php** on line **7**
dipesan...

Yup ternyata ada error yang tidak diharapkan, harapan kita adalah program diatas bisa mempilkan teks "**Televisi dipesan…**" tetapi ternyata variable jenis yang harusnya berisi televisi tidak dikenali…

Pesan error di atas mengatakan bahwa PHP tidak menemukan variabel `$jenis` di baris 7. Loh, bukannya di baris 3 kita sudah mendefinisikan property `$jenis?` yang kemudian diisi kembali di baris 12? kenapa PHP tidak bisa membacanya?

Untuk bisa memahami apa yang terjadi, kita harus kembali kepada definisi dari object, yakni implementasi konkret sebuah class. Selama pemrosesan kode PHP, yang seharusnya kita akses adalah object, bukan class. Class adalah tempat untuk pembuatan definisi saja (sebagai *blueprint*).

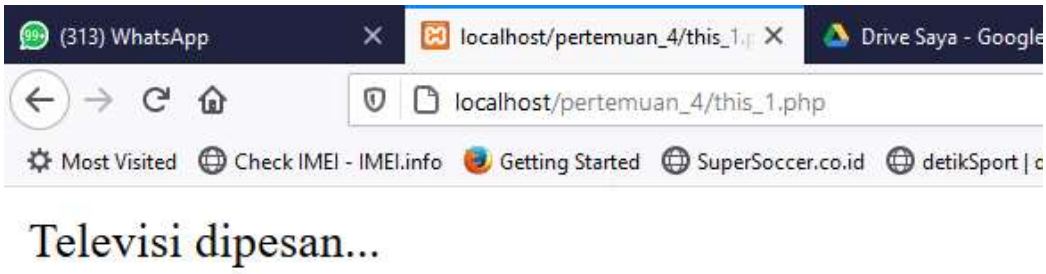
PENTING!!!

Ketika saya menulis perintah `return $jenis." dipesan..."` di dalam method `pesanProduk()`, artinya saya sedang mencoba mengakses **property milik class**, padahal seharusnya yang diakses itu adalah **property milik object**. Di sinilah kita butuh sebuah variabel bantu sebagai penanda bahwa yang ingin diakses adalah property milik object, **bukan milik class**. Variabel bantu tersebut adalah **$this**.

Agar method `pesanProduk()` bisa diproses sebagaimana mestinya, saya harus tulis ulang menjadi: `return $this->jenis." dipesan..."`. Berikut perubahan dari kode program sebelumnya:

```php
<?php
class Produk {
  public $jenis = "";
  public $merek = "";

  public function pesanProduk(){
    return $this->jenis." dipesan..."; //perubahan di sini!!!
  }
}

$produk01 = new Produk();
$produk01->jenis = "Televisi";
$produk01->merek = "Samsung";

echo $produk01->pesanProduk();
```

coba jalankan ulang program kalian dan pahami…



Pesan akan muncul sesuai dengan yang diharapkan.. sampai sini jelass???
Pahami lagi ya, baca pelan-pelan

Buat latihan kedua dengan memodifikasi sedikit baris program sebelumnya, simpan dengan nama **this_2.php**

```php
<?php
class Produk {
  public $jenis;
  public $merek;

  public function pesanProduk(){
    return $this->jenis." ".$this->merek." dipesan...";
  }
}

$produk01 = new Produk();
$produk01->jenis = "Televisi";
$produk01->merek = "Samsung";

$produk02 = new Produk();
$produk02->jenis = "Mesin cuci";
$produk02->merek = "LG";

echo $produk01->pesanProduk();     // Televisi Samsung dipesan...
echo "<br>";
echo $produk02->pesanProduk();     // Mesin cuci LG dipesan...
```

pada contoh ini terlihat jelas perbedaan dengan latihan **this.php sebelumnya,** silakan dibandingkan. Fungsi pada class hanya ada 1 fungsi generic yaitu pesanProduk() yang dapat digunakan berulang kali pada object yang dibentuk..