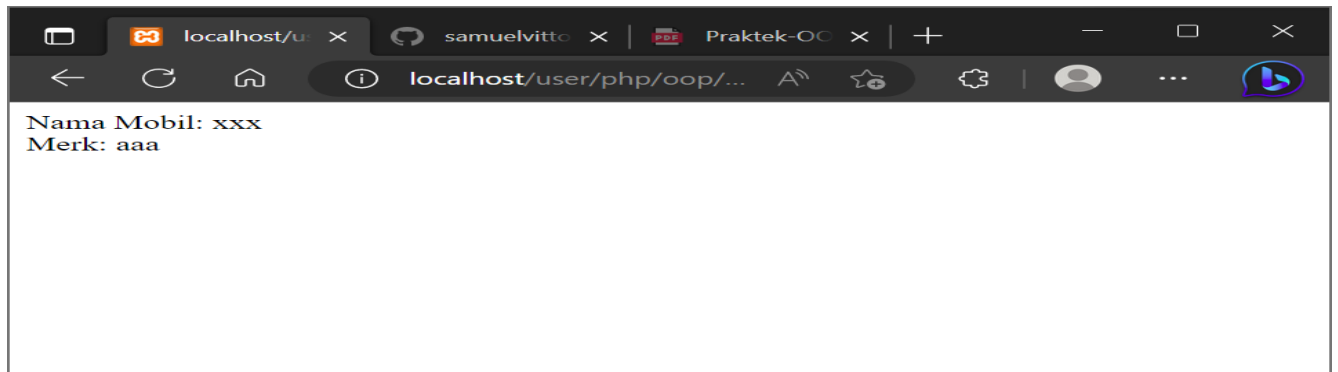


Nama : Samuel Vittorio Rivaldo

NIM : G.231.21.0187

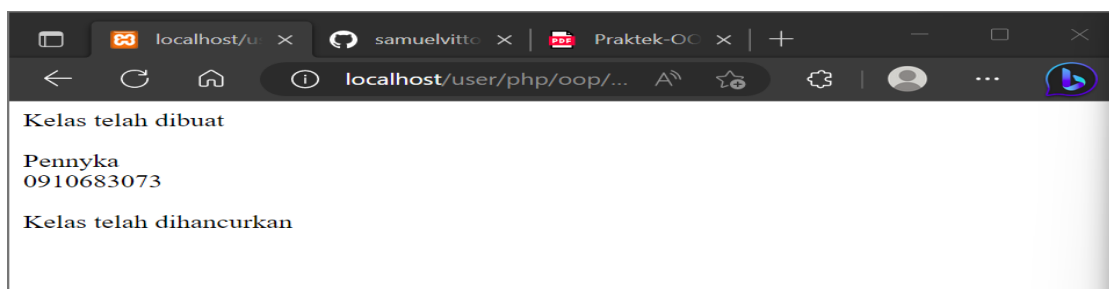


1a. Tampilan menampilkan nilai properti nama dan nama yang di masukkan secara langsung dari luar kelas mobil.

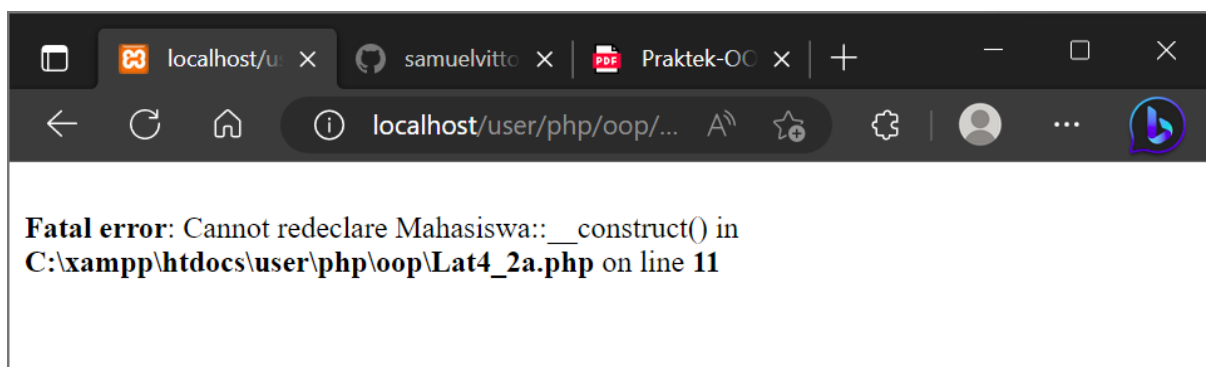
1b. Tidak ada perubahan yang terjadi karena parameter \$a tidak dipakai ke dalam method

1c. Kita dapat mengubah ubah property tanpa method dari luar kelas

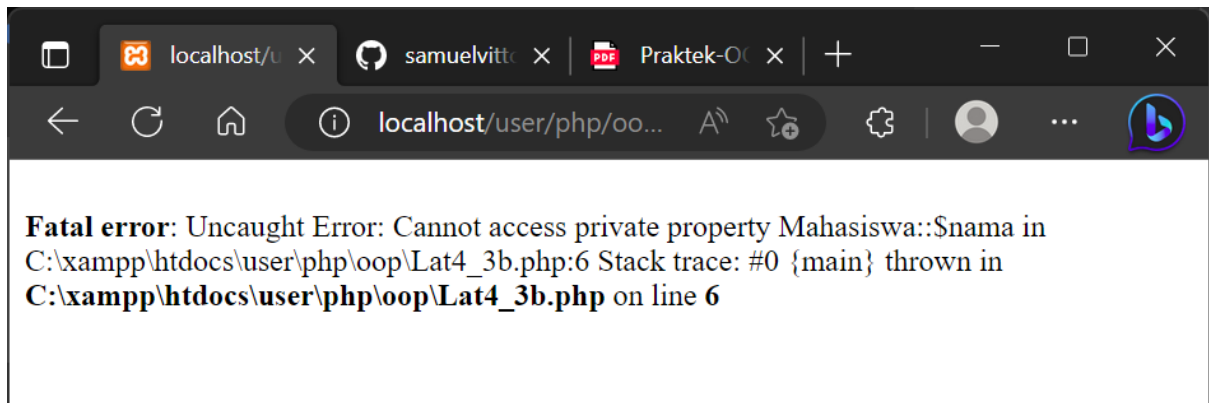
2. sebelum



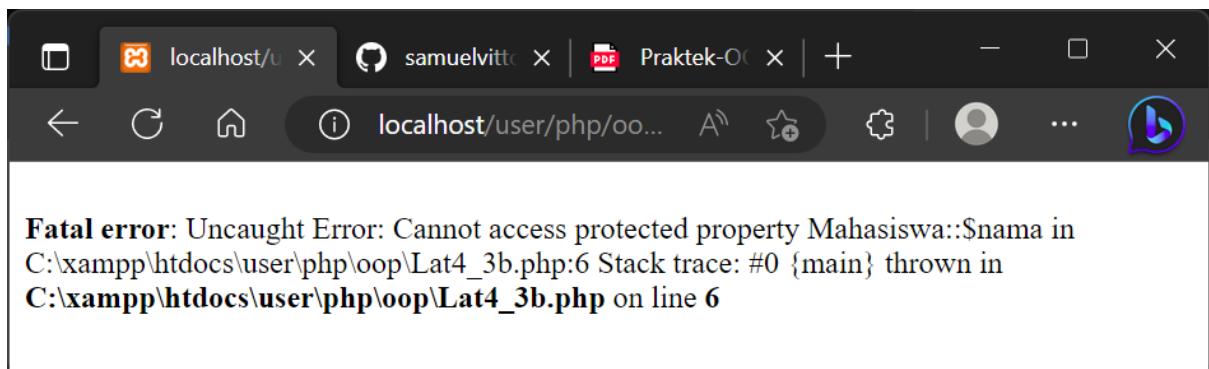
sesudah



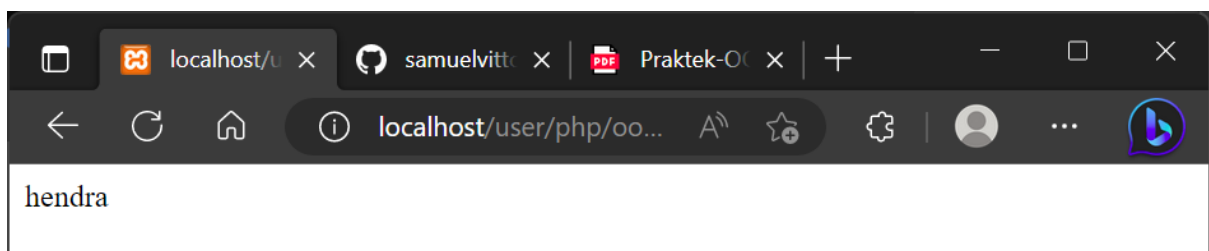
Kesimpulan :,fungsi konstruktor yang berjalan tiap instansiasi objek dapat digunakan untuk mengisi nilai – nilai tiap properti setiap kita menginstansiasi objek di dalam class, dan hanya boleh ada satu konstruktor di dalam satu kelas, lalu fungsi destruct memiliki fungsi kebalikan dari konstruktor yaitu untuk meruntuhkan properti dalam kelas.



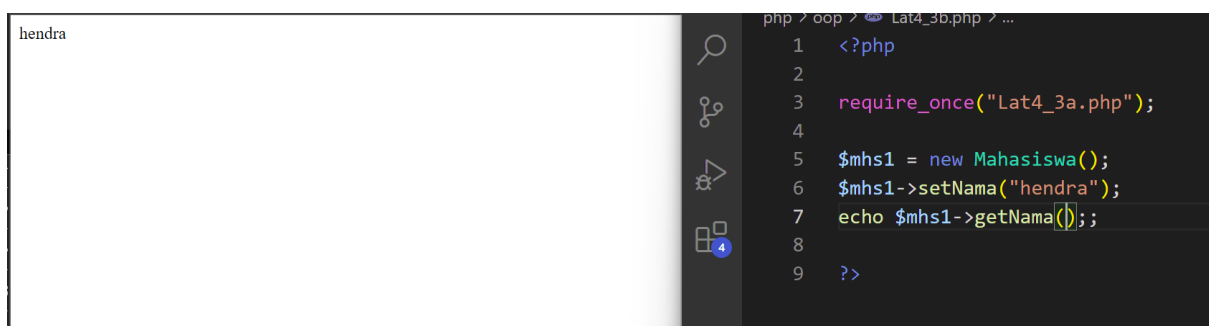
3a. Terjadi Error Karena property nama dari kelas Mahasiswa menggunakan Access Modifier Private, jadi property nama dan nim hanya bisa diakses di dalam kelas Mahasiswa saja. Jadi properti ini tidak bisa diakses dari luar kelas seperti yang dilakukan di latihan 4_3 ini.



3b1. Masih Terjadi error karena protected juga melindungi akses properti nama dan nim dari luar kelas. Protected Properti hanya bisa diakses oleh kelas yang bersangkutan beserta turunannya.

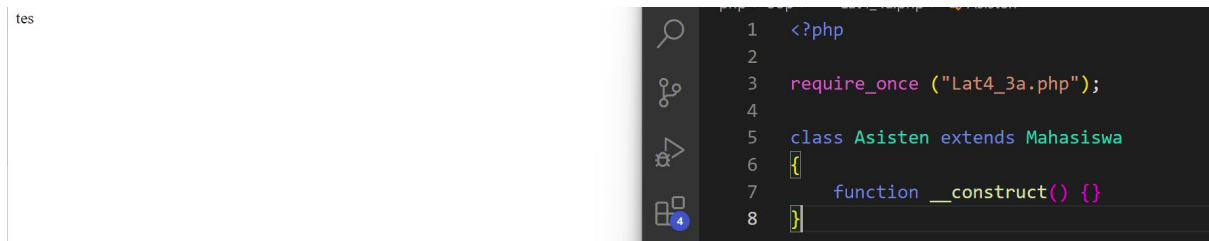


3b2. Berbeda dengan private dan protected, jika kita menggunakan public maka properti bisa diakses dari mana saja. Maka kali ini tidak terjadi error pada kode lat4.3



3c. Jika ingin mengakses properti private dari luar kelas, kita bisa menggunakan method setter dan getter yang sudah ada di dalam kelas Mahasiswa, selanjutnya mengganti pemanggilan properti dari luar kelas yang semula memanggil langsung properti, menjadi memanggil dengan menggunakan method setter dan getter seperti pada gambar sebelah kanan.


3d. Kesimpulan Lat4_3 : Dengan menggunakan Modifier protected dan private, kita bisa melindungi properti milik sebuah class dari akses dari luar kelas yang tidak diinginkan



```
1 <?php
2
3 require_once ("Lat4_3a.php");
4
5 class Asisten extends Mahasiswa
6 {
7     function __construct() {}
8 }
```

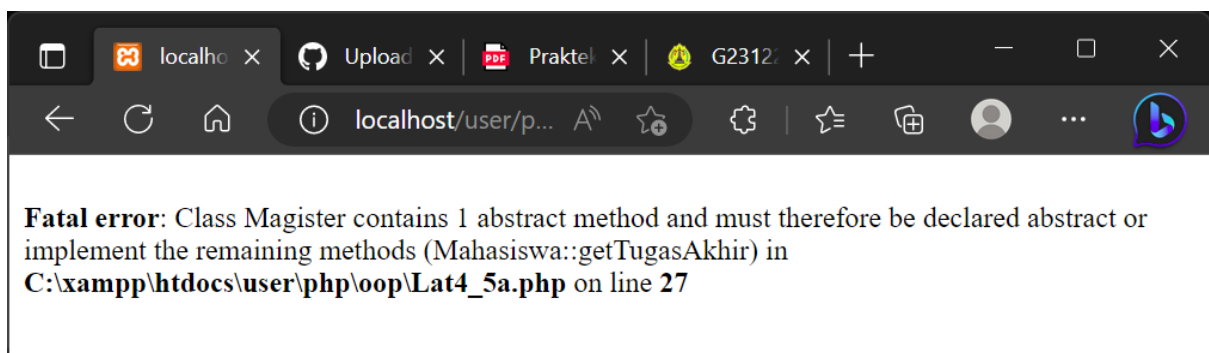
4. Kelas Asisten mewarisi semua properti dan method dari kelas Mahasiswa. Jadi meski tidak mengisi properti nama dan method setter dan getter nama pada kelas Asisten, PHP dapat menjalankan perintah setter dan getter dari kelas Asisten dan tidak terjadi error.

Mahasiswa S1 Skripsi
Mahasiswa S2 Tesis



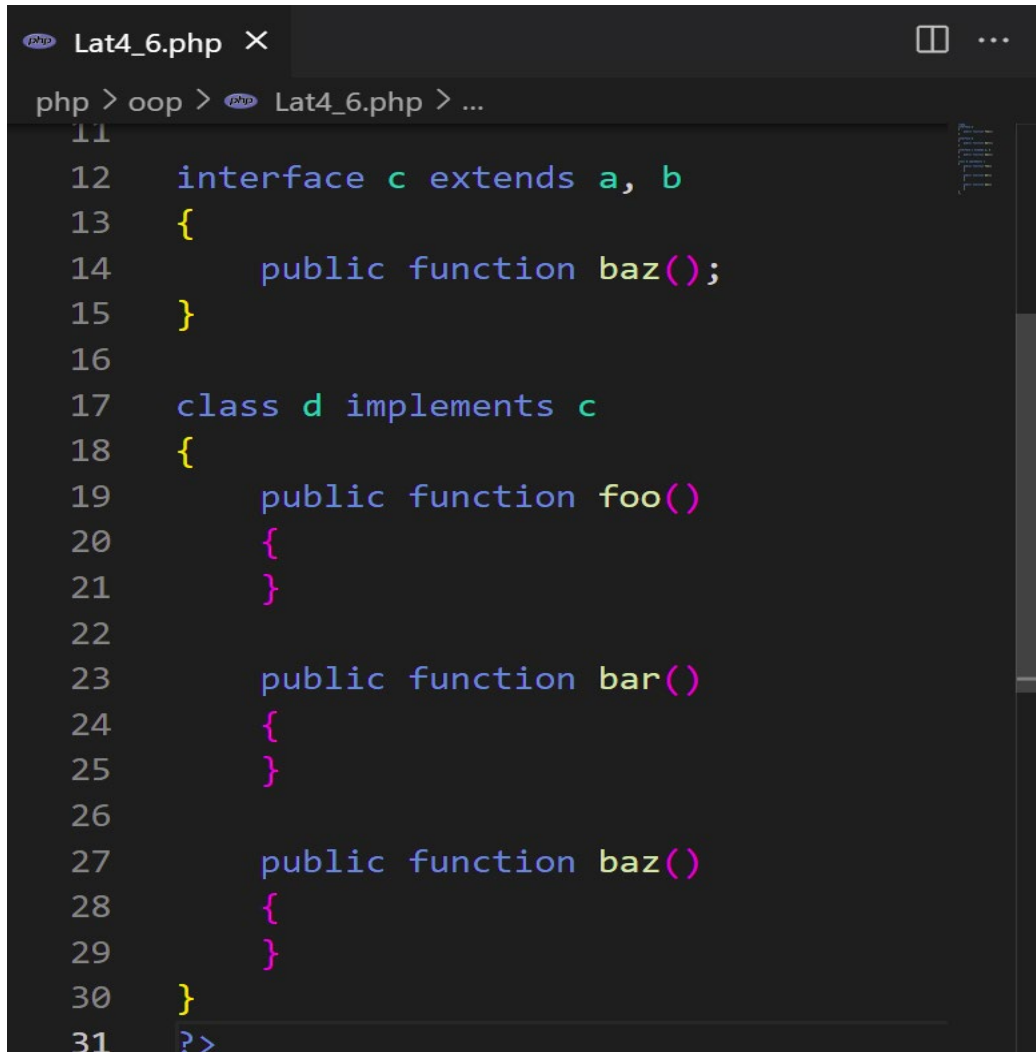
```
1 <?php
2
3 require_once("Lat4_5a.php");
4
5 $s = new Sarjana;
6 $s->getProgram('Mahasiswa') . "<br>";
7 $s->tugasAkhir();
8
9 $m = new Magister;
10 $m->getProgram('Mahasiswa') . "<br>";
11 $m->tugasAkhir();
```

5a. Tampilan Menampilkan Hasil dari instansiasi Kelas Sarjana dan Kelas Magister yang mewarisi kelas abstrak Mahasiswa beserta method abstrak getTugasAkhir milik kelas Mahasiswa yang wajib diimplementasikan pada kelas-kelas turunannya. Tidak ada error yang ditampilkan.



5b. Apabila baris 29-32 yang berisi method abstrak getTugasAkhir() dihapus, maka akan tampil error karena di dalam kelas Magister wajib mengimplementasikan 1 method abstrak dari kelas abstrak Mahasiswa yaitu getTugasAkhir() yang dihapus kali ini.

5c. Kesimpulan Lat4_5 : Apabila kita menggunakan kelas abstrak, maka kita wajib memperhatikan satu method abstrak yang harus diimplementasikan pada kelas-kelas turunannya.



```
11
12 interface a
13 {
14     public function foo();
15 }
16
17 interface b extends a
18 {
19     public function bar();
20 }
21
22 interface c extends a, b
23 {
24     public function baz();
25 }
26
27 class d implements c
28 {
29     public function foo()
30     {
31     }
32
33     public function bar()
34     {
35     }
36
37     public function baz()
38     {
39     }
40 }
41
```

6a. Kode di atas berisi tentang interface. Interface a memiliki method foo() yang diwariskan ke interface b dan c, interface b memiliki method bar() lalu interface c memiliki method baz(). Karena interface b adalah turunan interface a maka method milik interface a juga dimiliki oleh interface b. Interface c adalah turunan dari interface a dan b, maka method milik a dan b juga dimiliki oleh c. Sehingga interface c sekarang memiliki 3 method. Kemudian class d adalah turunan dari interface c, karena turunan dari interface harus memiliki semua method dari parent interfacenya, jadi class d juga harus memiliki 3 method milik interface c.

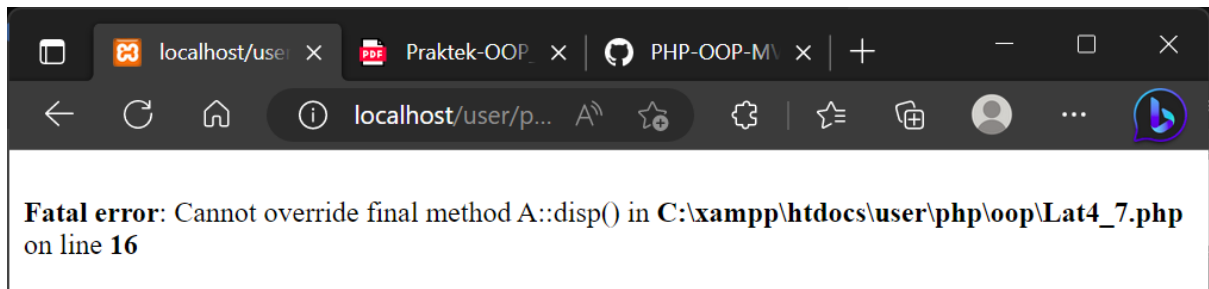


6b. Jika baris 27 – 29 yang berisi method baz() turunan dari interface c dihapus, maka kode tidak bisa berjalan (error) karena sebuah class implementasi dari interface harus memuat semua method milik parent interfacenya.

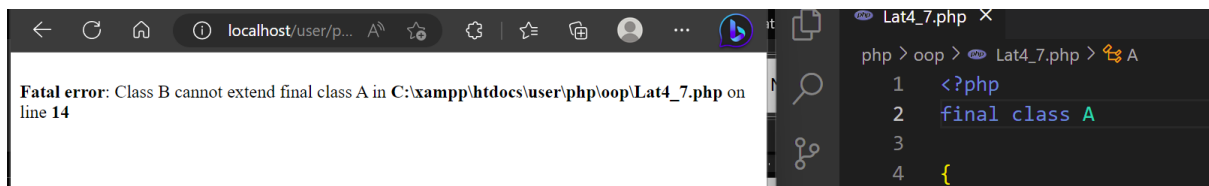
```
Lat4_6.php
php > oop > Lat4_6.php > e
22
23     public function bar()
24     {
25     }
26
27     public function baz()
28     {
29     }
30 }
31 class e implements b
32 {
33     public function foo()
34     {
35     }
36
37     public function bar()
38     {
39     }
40 }
41 ?>
```

6c. Jika ingin membuat kelas e dengan method foo() dan bar(), kita bisa membuat dengan mengimplementasikan dengan interface yang memiliki method yang ditunjuk. Dalam hal ini adalah interface b yang memiliki hanya 2 method tersebut.

6d. Kesimpulan : Interface dapat diwariskan kepada interface yang lain, sehingga interface turunannya juga memiliki method yang dimiliki oleh parentnya. Interface turunan jg bisa memiliki methodnya sendiri. Sehingga class yang menjadi implementasi interface turunan tadi harus memiliki semua method yang dimiliki interface yang menjadi parent class tersebut.

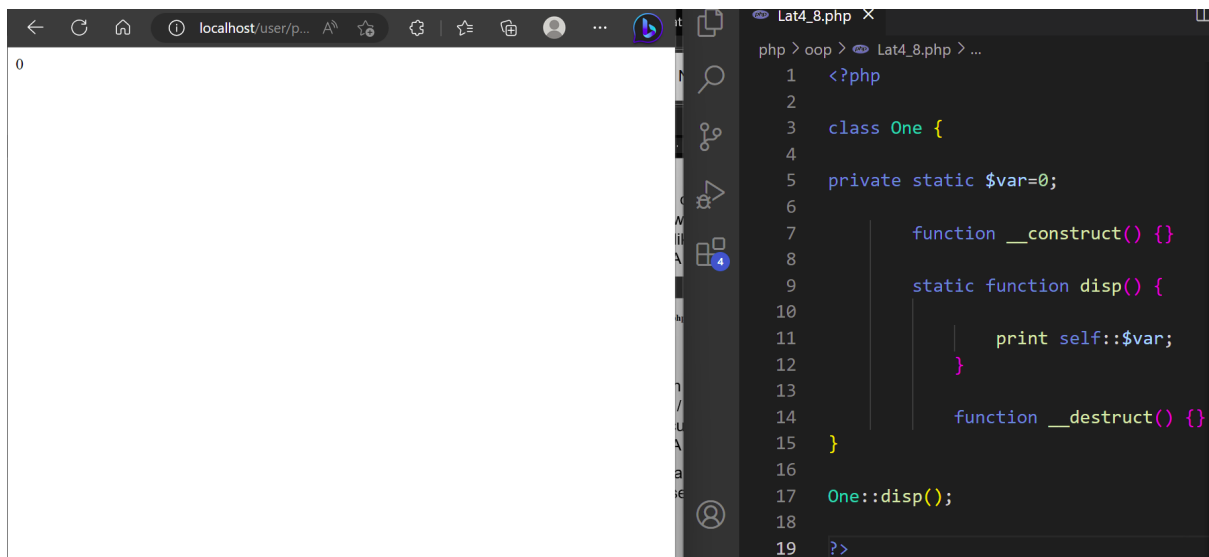


7a. Terjadi error pada tampilan, karena method disp() pada kelas A menggunakan keyword final yang artinya method ini tidak bisa diwariskan kepada kelas yang menjadi turunan dari kelas A. Jadi apabila di kelas B memiliki method dengan nama yang sama dengan tujuan untuk override method dari kelas A akan muncul error.

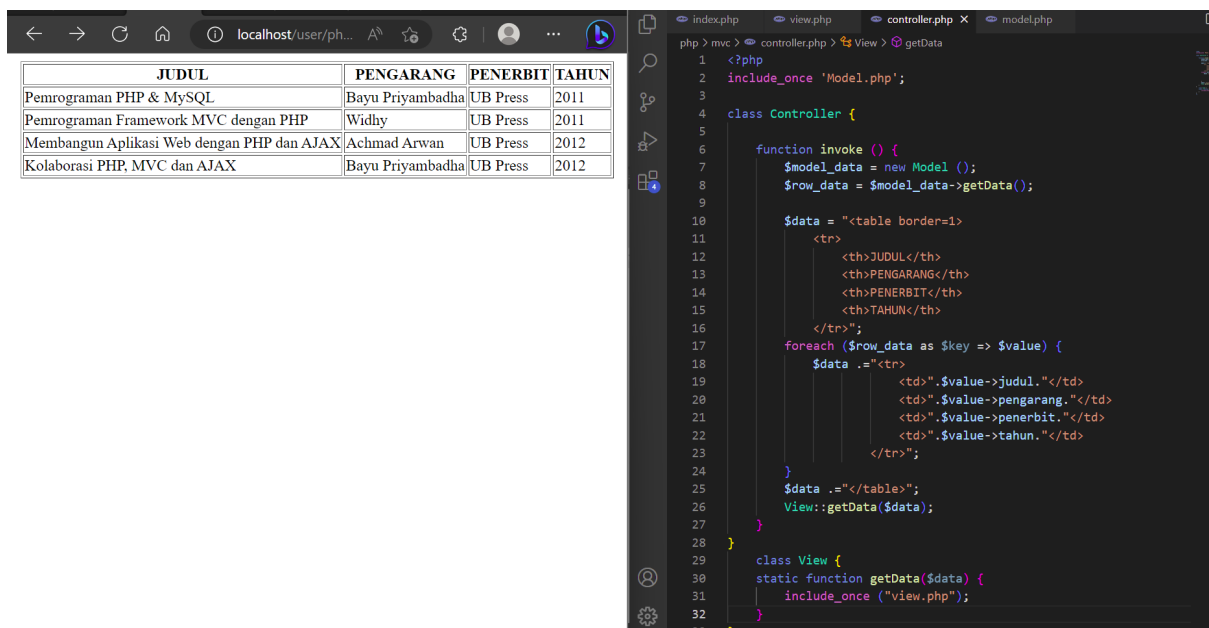
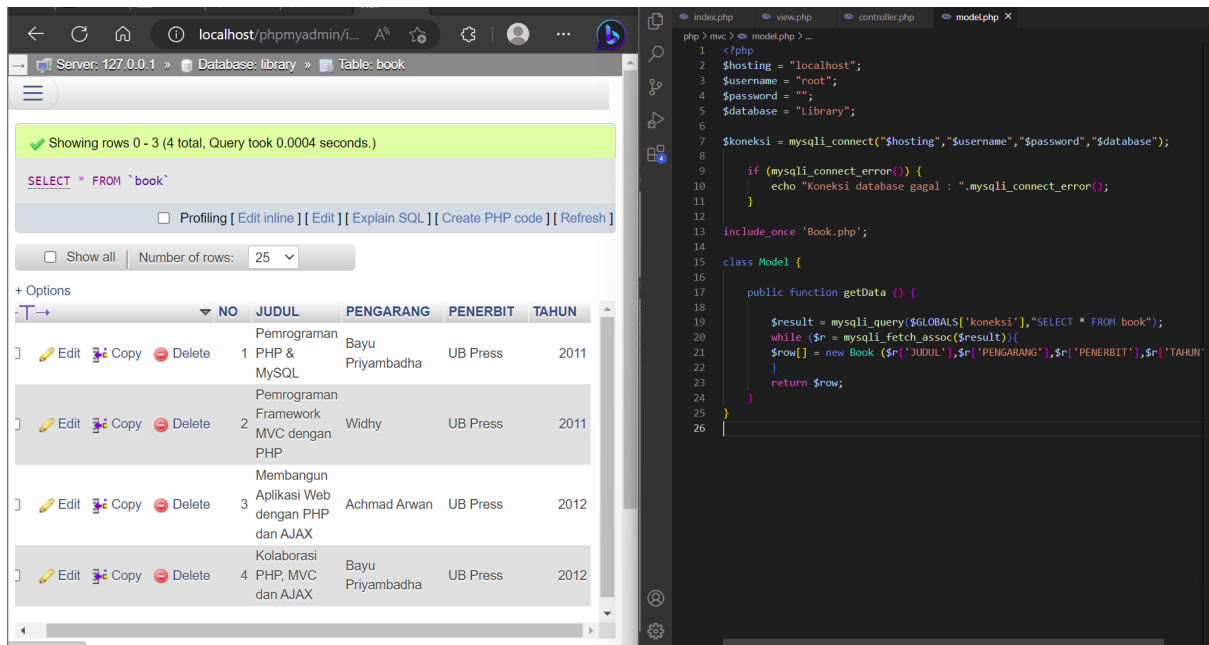


7b. Jika keyword final pada method disp() dipindah ke sebelum keyword kelas A, maka kelas A menjadi tidak bisa memiliki kelas turunan / kelas tidak bisa diwariskan kepada kelas lain. Sehingga pada tampilan di atas muncul error cannot extend final class A, karena kelas B berusaha menjadi turunan kelas A yang merupakan final kelas.

7c. Kesimpulan : Apabila keyword “final” ditambah pada sebuah class atau method pada suatu class, maka kelas atau method tersebut menjadi tidak bisa diwariskan kepada kelas yang lain.



8. Kesimpulan : Apabila kita menggunakan keyword static pada property atau method pada sebuah class, kita jadi bisa memanggil property atau method static tersebut tanpa menginstansiasi objek pada kelas tersebut dengan menggunakan keyword self sebagai pengganti \$this dan keyword {nama kelas} atau parent untuk pengganti {nama variabel objek}->.



9c.

