

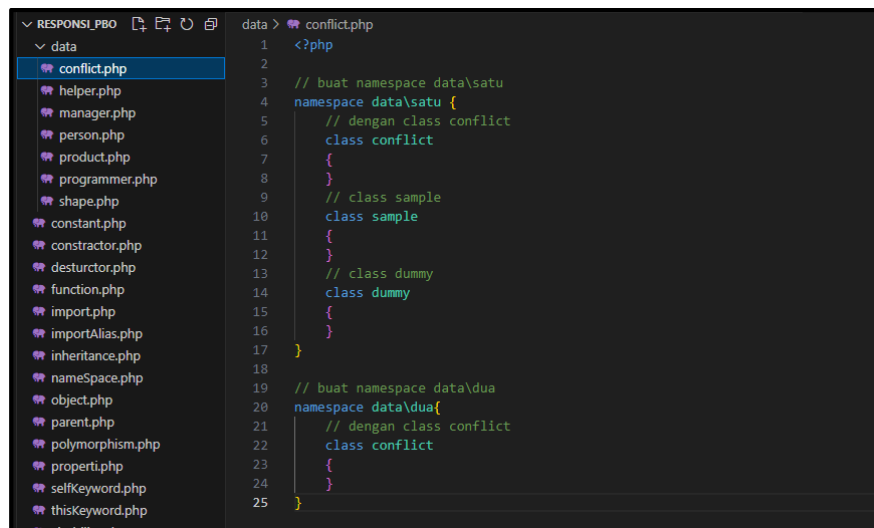
Nama : Zahra Nabila Melfiani

NPM : G1F022042

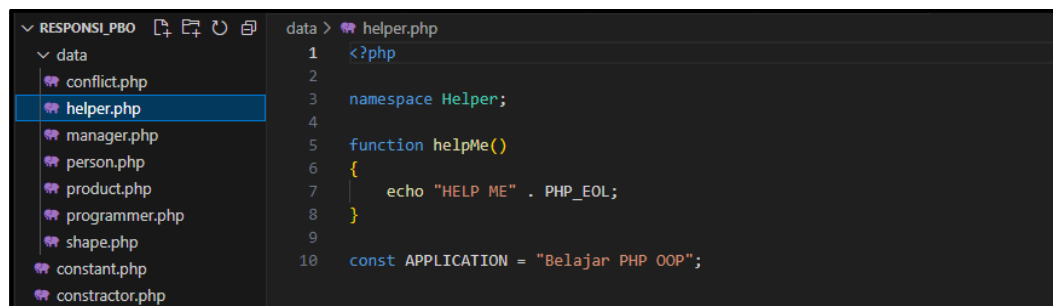
Matkul : Proyek Pemrograman Berorientasi Objek

RESPONSI PROYEK PEMROGRAMAN BERORIENTASI OBJEK

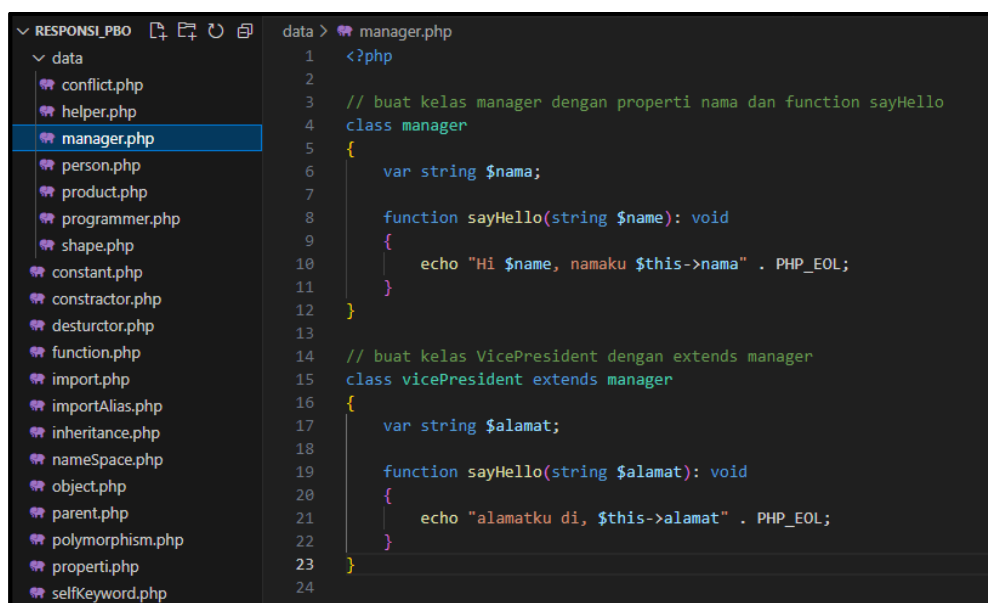
Setelah melakukan git clone pada repository yang telah ada, dan melengkapi code PHP yang belum lengkap. Maka didapatkanlah suatu code PHP dengan folder berjudul responsi_pbo.



```
data > conflict.php
1  <?php
2
3  // buat namespace data\satu
4  namespace data\satu {
5      // dengan class conflict
6      class conflict
7      {
8      }
9      // class sample
10     class sample
11     {
12     }
13     // class dummy
14     class dummy
15     {
16     }
17 }
18
19 // buat namespace data\dua
20 namespace data\dua{
21     // dengan class conflict
22     class conflict
23     {
24     }
25 }
```



```
data > helper.php
1  <?php
2
3  namespace Helper;
4
5  function helpMe()
6  {
7      echo "HELP ME" . PHP_EOL;
8  }
9
10 const APPLICATION = "Belajar PHP OOP";
```



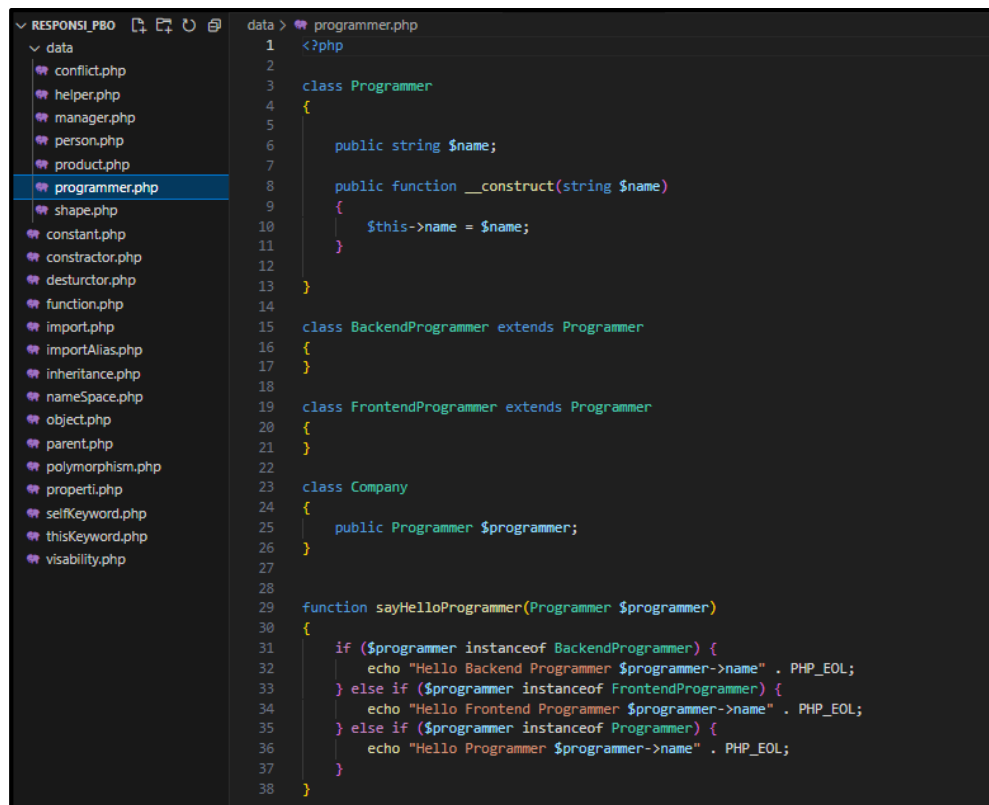
```
data > manager.php
1  <?php
2
3  // buat kelas manager dengan properti nama dan function sayHello
4  class manager
5  {
6      var string $nama;
7
8      function sayHello(string $name): void
9      {
10         echo "Hi $name, namaku $this->nama" . PHP_EOL;
11     }
12 }
13
14 // buat kelas VicePresident dengan extends manager
15 class vicePresident extends manager
16 {
17     var string $alamat;
18
19     function sayHello(string $alamat): void
20     {
21         echo "alamatku di, $this->alamat" . PHP_EOL;
22     }
23 }
24
```

```
RESPONSI_PBO
data
  conflict.php
  helper.php
  manager.php
  person.php
  product.php
  programmer.php
  shape.php
  constant.php
  constructor.php
  destructor.php
  function.php
  import.php
  importAlias.php
  inheritance.php
  namespace.php
  object.php
  parent.php
  polymorphism.php
  properti.php
  selfKeyword.php
  thisKeyword.php
  visibility.php

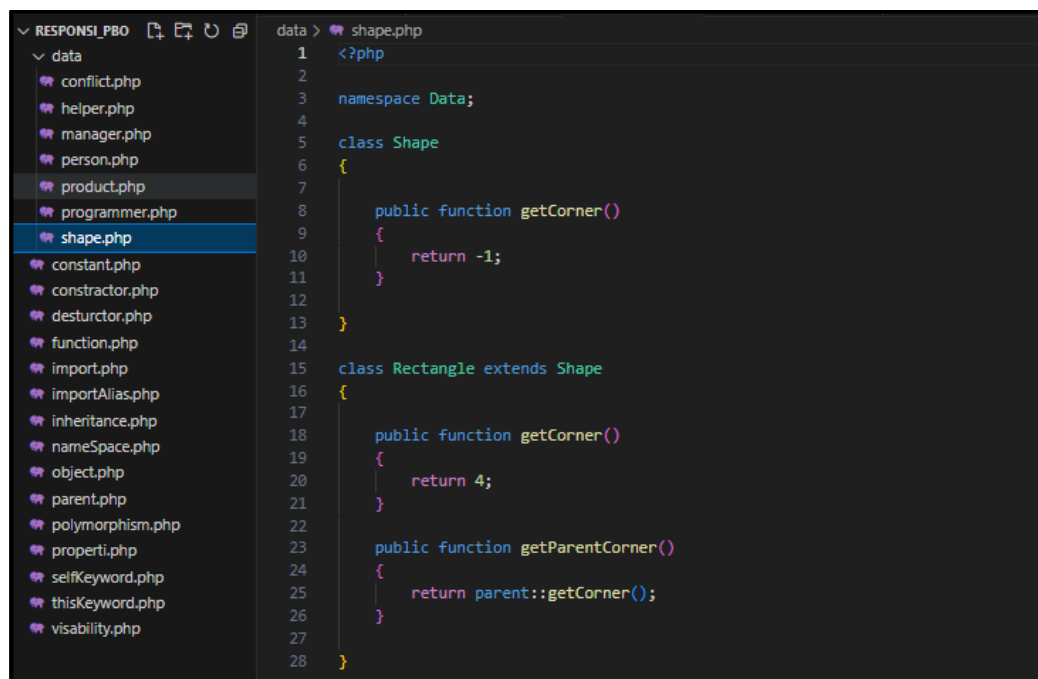
data > person.php
1 <?php
2 // membuat kelas person
3 class person{
4     // membuat properti
5     var string $nama;
6     // gunakan nullable properti
7     var ?string $alamat = null;
8     // gunakan default value untuk properti
9     var string $negara = "Indonesia";
10    // buat function sayHello
11    function sayHello(string $name)
12    {
13        echo "Hello $name" . PHP_EOL;
14    }
15    // buat function sayHello nullable dengan percabangan
16    function sayHelloNull(?string $name)
17    {
18        if (is_null($name)) {
19            echo "Hi,{\$this->nama}" . PHP_EOL;
20        } else {
21            echo "Hi $name, namaku {\$this->nama}" . PHP_EOL;
22        }
23    }
24    // buat const author
25    const AUTHOR = "Zahra Nabila Melfiani";
26    // buat function info untuk self keyword
27    function info()
28    {
29        echo "AUTHOR : " . self::AUTHOR . PHP_EOL;
30    }
31    // buat function constructor
32    function __construct(string $nama, ?string $alamat)
33    {
34        \$this->nama = $nama;
35        \$this->alamat = $alamat;
36    }
37    // buat function destructor
38    function __destruct()
39    {
40        echo "Object person \$this->nama telah di destroyed" . PHP_EOL;
41    }
42 }
```

```
RESPONSI_PBO
data
  conflict.php
  helper.php
  manager.php
  person.php
  product.php
  programmer.php
  shape.php
  constant.php
  constructor.php
  destructor.php
  function.php
  import.php
  importAlias.php
  inheritance.php
  namespace.php
  object.php
  parent.php
  polymorphism.php
  properti.php
  selfKeyword.php
  thisKeyword.php
  visibility.php

data > product.php
1 <?php
2
3 class Product
4 {
5     protected string $name;
6     protected int $price;
7
8     public function __construct(string $name, int $price)
9     {
10         \$this->name = $name;
11         \$this->price = $price;
12     }
13
14     public function getName(): string
15     {
16         return \$this->name;
17     }
18
19     public function getPrice(): int
20     {
21         return \$this->price;
22     }
23 }
24
25 class ProductDummy extends Product
26 {
27
28     public function info()
29     {
30         echo "Name \$this->name" . PHP_EOL;
31         echo "Price \$this->price" . PHP_EOL;
32     }
33
34 }
```



```
1 <?php
2
3 class Programmer
4 {
5     public string $name;
6
7     public function __construct(string $name)
8     {
9         $this->name = $name;
10    }
11
12 }
13
14 class BackendProgrammer extends Programmer
15 {
16 }
17
18 class FrontendProgrammer extends Programmer
19 {
20 }
21
22 class Company
23 {
24     public Programmer $programmer;
25 }
26
27
28
29 function sayHelloProgrammer(Programmer $programmer)
30 {
31     if ($programmer instanceof BackendProgrammer) {
32         echo "Hello Backend Programmer $programmer->name" . PHP_EOL;
33     } else if ($programmer instanceof FrontendProgrammer) {
34         echo "Hello Frontend Programmer $programmer->name" . PHP_EOL;
35     } else if ($programmer instanceof Programmer) {
36         echo "Hello Programmer $programmer->name" . PHP_EOL;
37     }
38 }
```



```
1 <?php
2
3 namespace Data;
4
5 class Shape
6 {
7
8     public function getCorner()
9     {
10         return -1;
11     }
12
13 }
14
15 class Rectangle extends Shape
16 {
17
18     public function getCorner()
19     {
20         return 4;
21     }
22
23     public function getParentCorner()
24     {
25         return parent::getCorner();
26     }
27
28 }
```

Gambar 1 Code data.

Penjelasan :

Setiap file memiliki keterkaitan, file di dalam folder data merupakan konsep-konsep dasar dalam pemrograman berorientasi objek. Beberapa file menggunakan konsep inheritance, namespace, polymorphism, dan visibility untuk menunjukkan penggunaan dasar-dasar OOP dalam PHP. File-file tersebut saling terhubung dan membentuk sebuah aplikasi sederhana yang menggambarkan konsep dasar dari Pemrograman Berorientasi Objek (OOP).

1. conflict.php

Berada dalam namespace Data\satu dan Data\dua, lalu membuat class Conflict di kedua namespace tersebut yang bertujuan adalah untuk memberikan contoh penggunaan namespace dan konflik nama class. namespace adalah cara untuk mengatur kode dalam kelompok logis atau ruang nama tertentu. namespace membantu menghindari konflik nama antar kelas, fungsi, dan variable. namespace tidak hanya digunakan untuk kelas, tetapi juga dapat digunakan untuk fungsi dan konstanta. class digunakan untuk membuat blueprint atau cetakan untuk objek. Objek adalah instansiasi dari sebuah class, dan class menyediakan struktur dan perilaku yang akan dimiliki objek tersebut.

2. helper.php

Berada dalam namespace Helper. Lalu membuat fungsi helpMe dan konstanta APPLICATION di dalam namespace tersebut yang digunakan untuk memberikan contoh penggunaan namespace pada file nameSpace.php. function digunakan untuk mendefinisikan dan memanggil suatu blok kode yang dapat dijalankan kapan saja dalam program. Function membantu dalam mengorganisir dan memecah kode menjadi bagian-bagian yang lebih kecil dan dapat digunakan ulang. Function juga dapat menerima parameter dan mengembalikan nilai. const digunakan untuk mendefinisikan konstanta. Konstanta adalah nilai yang tidak dapat diubah selama eksekusi program. Penting untuk diingat bahwa konstanta tidak menggunakan tanda dolar “\$” seperti variabel, dan biasanya dinamakan dengan huruf kapital untuk membedakan dari variabel. Konstanta dapat didefinisikan di dalam class atau di luar class, dan mereka dapat diakses menggunakan “::” jika didefinisikan di dalam class.

3. manager.php

Membuat class manager dan vicePresident. Class vicePresident merupakan turunan dari class manager. Code ini memberikan contoh penggunaan inheritance dalam konsep OOP. Terdapat code berupa var dalam file manager.php, var dapat digunakan untuk mendeklarasikan properti (atribut) dalam sebuah class.

4. person.php

Pada file person.php digunakan untuk membuat class person. Class ini memiliki properti, metode, konstanta, constructor, destructor, dan nullable property. Code ini memberikan contoh penggunaan konsep dasar OOP seperti properti, metode, konstanta, constructor, destructor, dan nullable property. Terdapat code PHP_EOL yang dimana PHP_EOL adalah konstanta yang menyimpan karakter baris baru sesuai dengan lingkungan sistem operasi tempat PHP dijalankan. Ini memungkinkan pengguna untuk membuat baris baru dalam output teks tanpa tergantung pada sistem operasi tertentu.

5. product.php

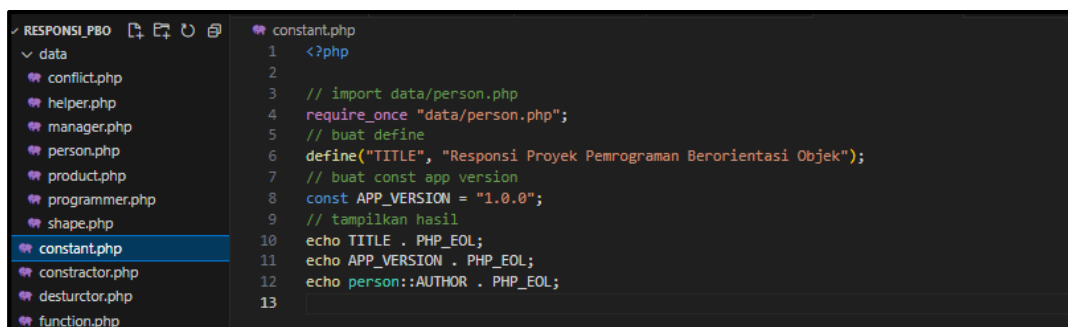
pada file product.php membuat class Product dan ProductDummy. Class ProductDummy merupakan turunan dari class Product. Code tersebut memberikan contoh penggunaan inheritance dan protected property dalam OOP. Kata kunci protected digunakan untuk menentukan tingkat akses properti atau metode dalam suatu class. Properti atau metode yang dinyatakan sebagai protected dapat diakses hanya dari dalam class itu sendiri dan class turunannya (subclasses). public digunakan untuk menentukan tingkat akses properti atau metode dalam PHP. Ketika suatu properti atau metode dinyatakan sebagai public, itu berarti dapat diakses dari mana saja, baik dari dalam class itu sendiri, dari class turunannya, atau dari luar class.

6. programmer.php

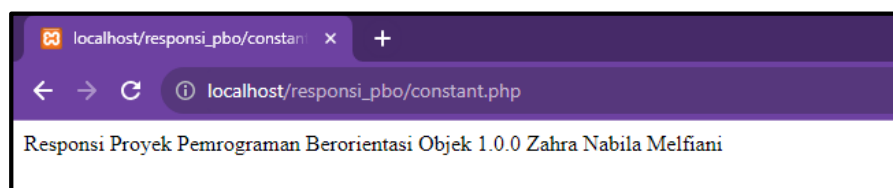
Code pada file programmer.php digunakan untuk membuat class Programmer, BackendProgrammer, FrontendProgrammer, dan Company. Code tersebut memberikan contoh penggunaan polymorphism dengan membuat objek Programmer yang bisa diisi dengan objek dari class turunannya (BackendProgrammer dan FrontendProgrammer). Class Company digunakan untuk menyimpan objek Programmer.

7. shape.php

Berada dalam namespace Data. Dimana digunakan untuk membuat class Shape dan Rectangle. Class Rectangle merupakan turunan dari class Shape. Code tersebut memberikan contoh penggunaan inheritance, overriding method, dan menggunakan parent class.



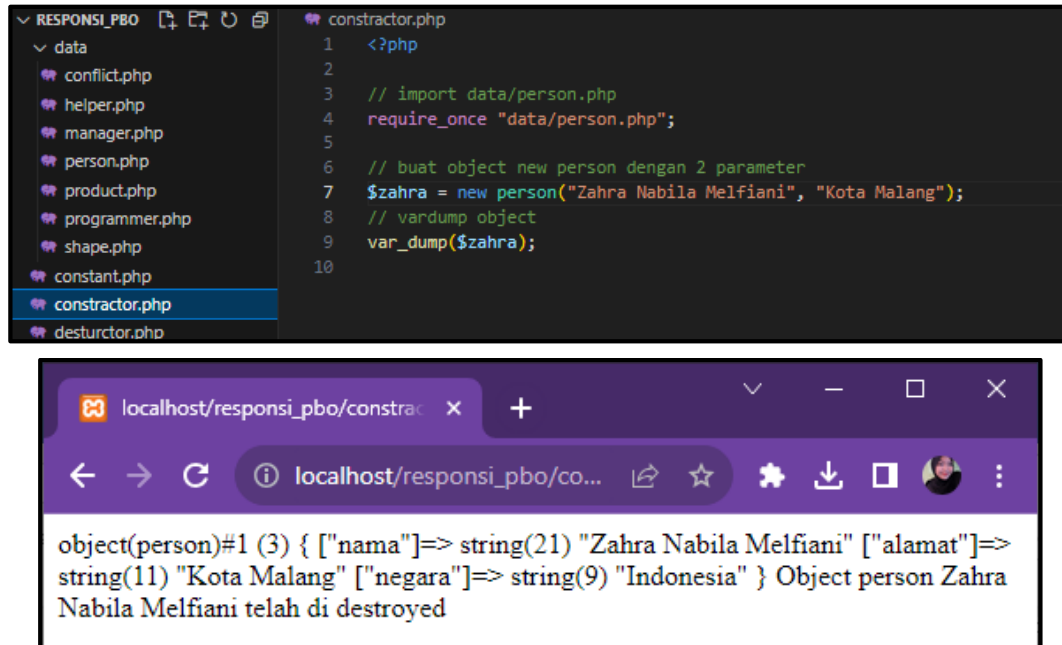
```
constant.php
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5  // buat define
6  define("TITLE", "Responsi Proyek Pemrograman Berorientasi Objek");
7  // buat const app version
8  const APP_VERSION = "1.0.0";
9  // tampilkan hasil
10 echo TITLE . PHP_EOL;
11 echo APP_VERSION . PHP_EOL;
12 echo person::AUTHOR . PHP_EOL;
13
```



Gambar 2 constant.php

Penjelasan :

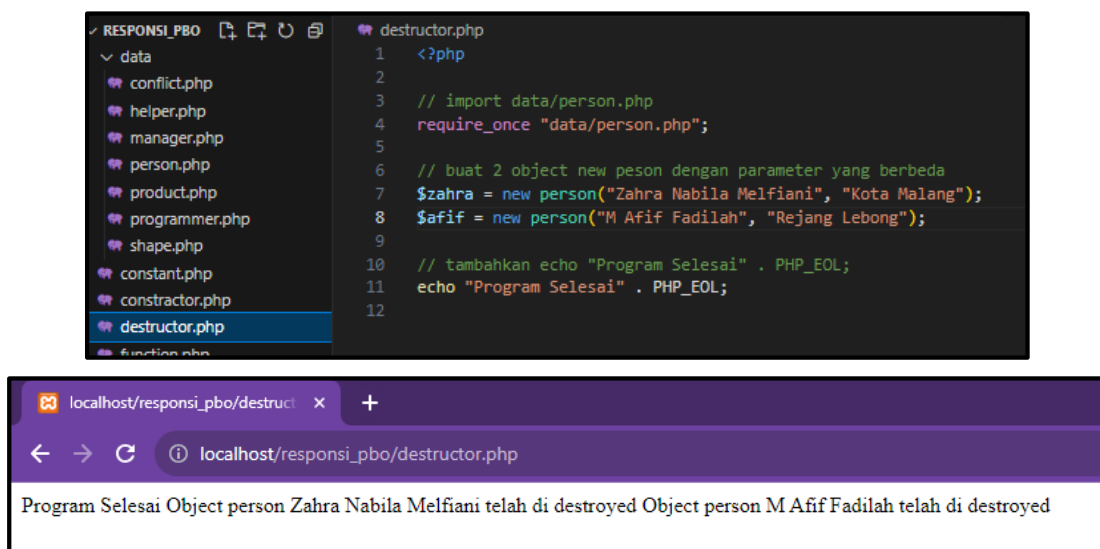
Dalam constant.php , menggunakan konsep konstanta (“define” dan “const”) untuk mendefinisikan nilai-nilai tetap. Menggunakan class “person” dari file “person.php” yang mengandung properti dan metode statis (“AUTHOR”). Dan menggunakan “require_once” untuk mengimpor file.



Gambar 3 constructor.php

Penjelasan :

Dalam constant.php , menggunakan var untuk mendeklarasikan atribut berupa “\$zahra”, dan menggunakan “require_once” untuk mengimpor file.

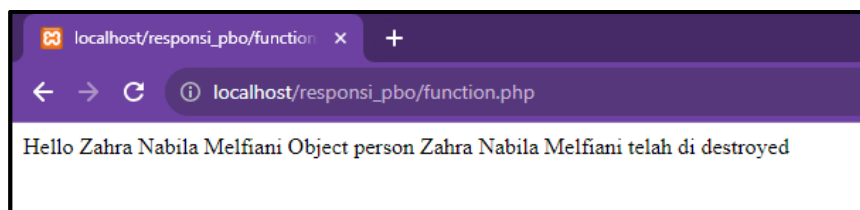


Gambar 4 destructor.php

Penjelasan :

Membuat dua objek “person” dengan parameter yang berbeda. Penggunaan konsep destructor untuk menampilkan pesan ketika objek dihancurkan “__destruct()”. Lalu menampilkan pesan "Program Selesai" setelah objek dibuat.

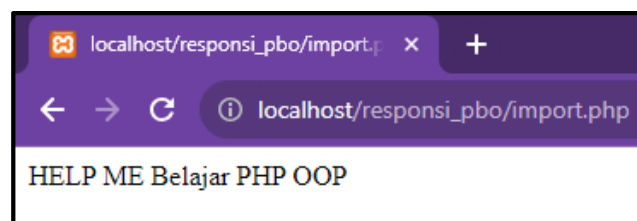
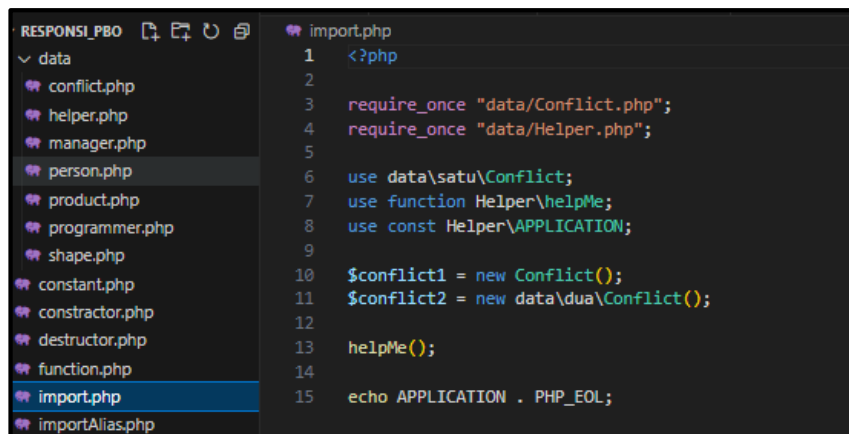
```
function.php
1  <?php
2
3  // import data/person.php
4  require "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new person("Zahra Nabila Melfiani","Kota Malang");
8
9  // panggil function
10 $person1->sayHello("Zahra Nabila Melfiani");
11
```



Gambar 5 function.php

Penjelasan :

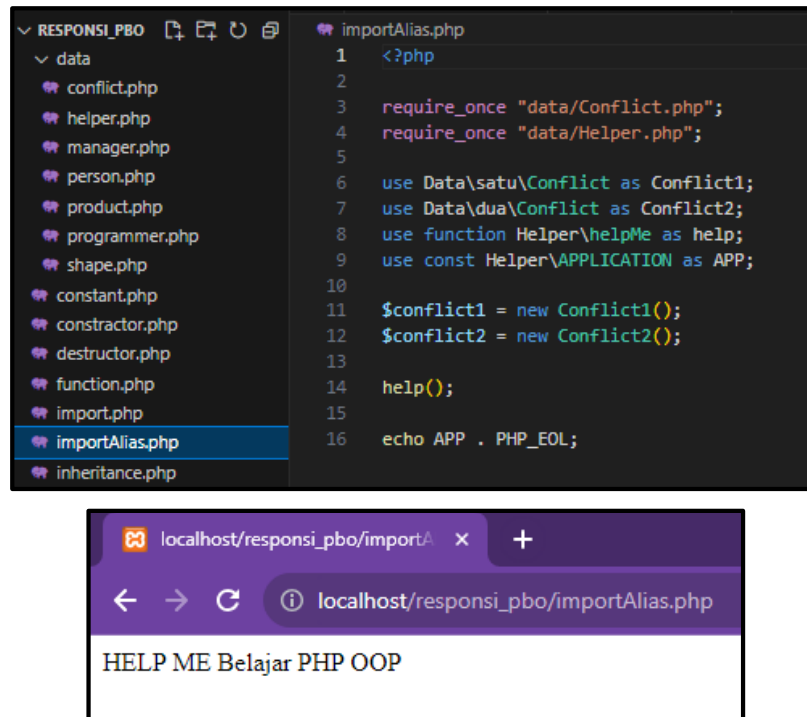
Menggunakan class “person” dari file “person.php”. Membuat objek “Person” dan memanggil metode “sayHello” dengan parameter, serta menunjukkan konsep fungsi dalam OOP.



Gambar 6 import.php

Penjelasan :

Menggunakan konsep import untuk mengakses class, fungsi, dan konstanta dari file lain. Penggunaan “require_once” untuk mengimpor file. Memanggil fungsi helpMe dan menampilkan nilai dari konstanta “APPLICATION”.

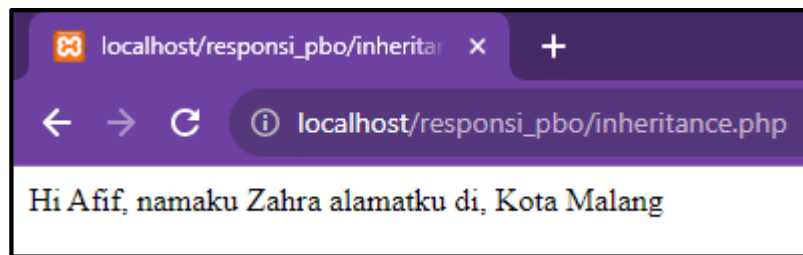


Gambar 7 importAlias.php

Penjelasan :

Penggunaan “require_once” untuk mengimpor file, mengimport class Conflict dari namespace “Data\satu” dan “Data\dua”, dan fungsi serta konstanta dari namespace Helper. Menggunakan konsep namespace untuk mengorganisir kode “Data\satu” dan “Data\dua”. Penggunaan konsep import untuk mengakses class, fungsi, dan konstanta dari file lain. Penggunaan alias digunakan untuk memberikan nama alternatif saat mengimpor class, fungsi, dan konstanta. Membuat objek Conflict1 dari namespace “Data\satu” dan Conflict2 dari namespace “Data\dua”. Fungsi helpMe dipanggil dan menampilkan nilai dari konstanta “APPLICATION”.

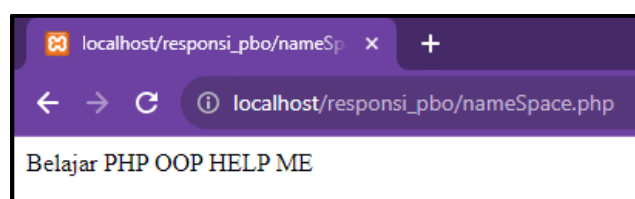
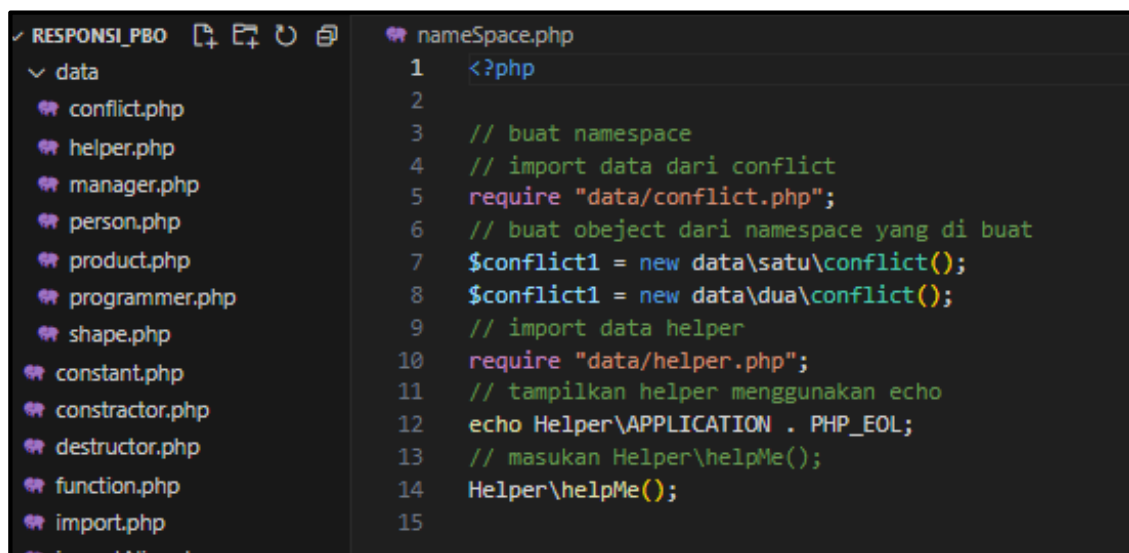
```
inheritance.php
1  <?php
2
3  // import data/person.php
4  require "data/manager.php";
5
6  // buat object new manager dan tambahkan value nama kemudian panggil function
7  $manager1 = new manager();
8  $manager1->nama = "Zahra";
9  $manager1->sayHello("Afif");
10
11 // buat object new vicepresident dan tambahkan value nama kemudian panggil function
12 $vicePresident1 = new vicePresident();
13 $vicePresident1->alamat = "Kota Malang";
14 $vicePresident1->sayHello("Zahra");
15
```

Gambar 8 inheritance.php

Penjelasan :

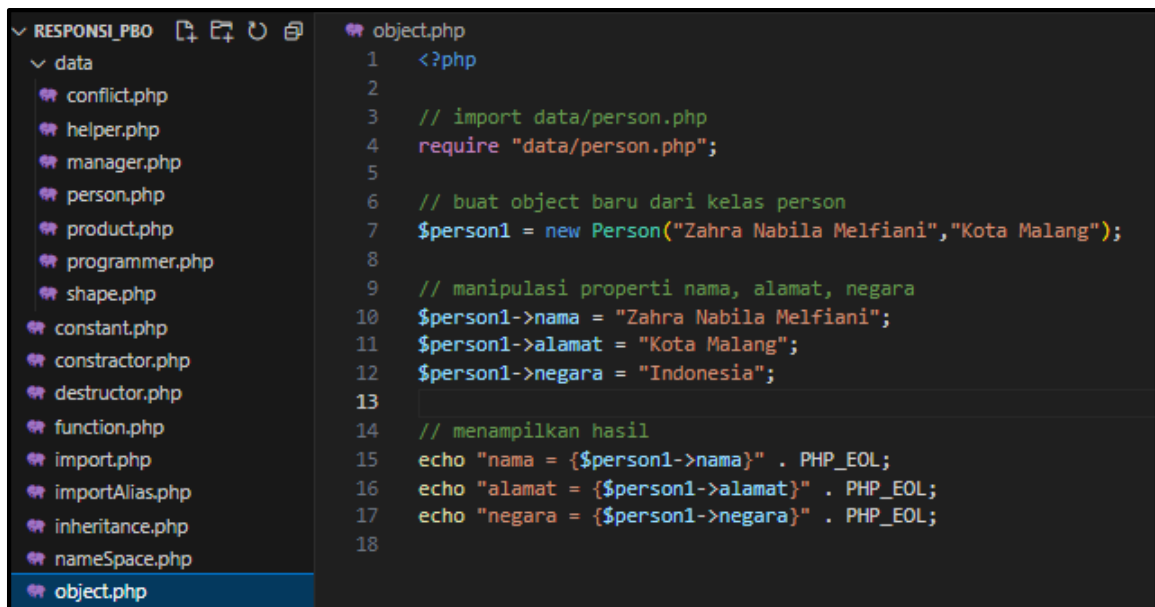
Menggunakan “require” untuk mengimport file, dimana mengimport class “manager” dan “vicePresident” dari file manager.php. Konsep inheritance digunakan dengan membuat class “manager” yang merupakan turunan dari class “person”. Membuat objek dari class “manager” dan “vicePresident” yang mewarisi properti dan metode dari class “person”, membuat objek manager dan vicePresident, kemudian memanggil method sayHello. Hal ini menunjukkan sebuah konsep pewarisan (inheritance).



Gambar 9 nameSpace.php

Penjelasan :

Menunjukkan penggunaan “require” untuk mengimpor file. Mengimport class Sample, Dummy, dan Conflict2 dari namespace Data\satu dan Data\dua. Seperti pada gambar penggunaan alias untuk namespace “Helper”.



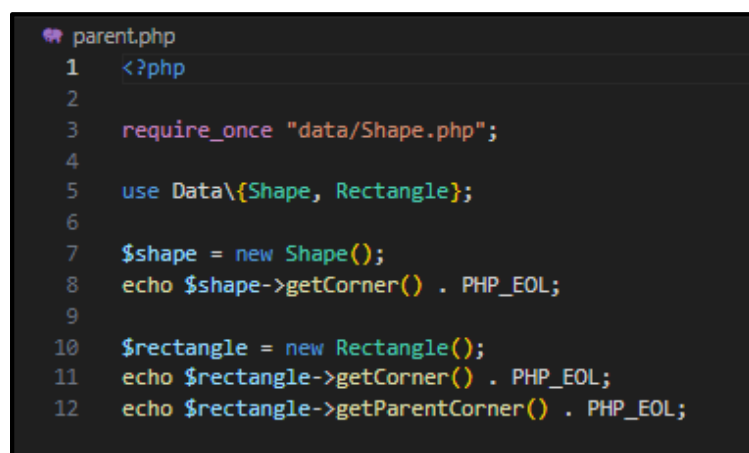
```
1 <?php
2
3 // import data/person.php
4 require "data/person.php";
5
6 // buat object baru dari kelas person
7 $person1 = new Person("Zahra Nabila Melfiani","Kota Malang");
8
9 // manipulasi properti nama, alamat, negara
10 $person1->nama = "Zahra Nabila Melfiani";
11 $person1->alamat = "Kota Malang";
12 $person1->negara = "Indonesia";
13
14 // menampilkan hasil
15 echo "nama = {$person1->nama}" . PHP_EOL;
16 echo "alamat = {$person1->alamat}" . PHP_EOL;
17 echo "negara = {$person1->negara}" . PHP_EOL;
18
```



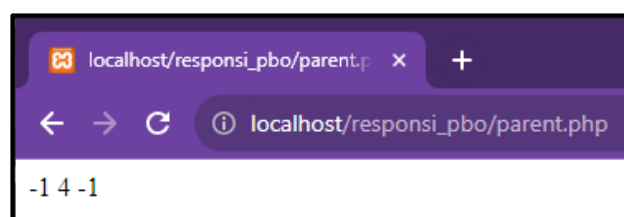
Gambar 10 object.php

Penjelasan :

Seperti pada gambar, code di atas bertujuan untuk mengimport class person dari file person.php, membuat objek person dan mengubah properti nama, alamat, dan negara, serta menampilkan nilai properti atau atribut.



```
1 <?php
2
3 require_once "data/Shape.php";
4
5 use Data\{Shape, Rectangle};
6
7 $shape = new Shape();
8 echo $shape->getCorner() . PHP_EOL;
9
10 $rectangle = new Rectangle();
11 echo $rectangle->getCorner() . PHP_EOL;
12 echo $rectangle->getParentCorner() . PHP_EOL;
```

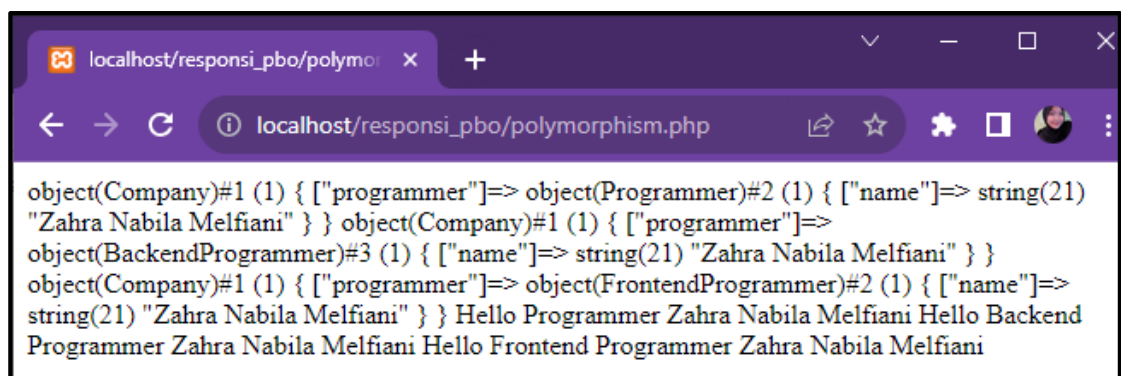


Gambar 11 parent.php

Penjelasan :

Seperti pada gambar di atas, code digunakan untuk mengimport class Shape dan Rectangle dari file Shape.php, membuat objek Shape dan Rectangle, kemudian menampilkan hasil pemanggilan metode, serta menunjukkan konsep pewarisan dan pemanggilan metode dari parent class.

```
polymorphism.php
1  <?php
2
3  require_once "data/Programmer.php";
4
5  $company = new Company();
6  $company->programmer = new Programmer("Zahra Nabila Melfiani");
7  var_dump($company);
8
9  $company->programmer = new BackendProgrammer("Zahra Nabila Melfiani");
10 var_dump($company);
11
12 $company->programmer = new FrontendProgrammer("Zahra Nabila Melfiani");
13 var_dump($company);
14
15 sayHelloProgrammer(new Programmer("Zahra Nabila Melfiani"));
16 sayHelloProgrammer(new BackendProgrammer("Zahra Nabila Melfiani"));
17 sayHelloProgrammer(new FrontendProgrammer("Zahra Nabila Melfiani"));
```



localhost/responsi_pbo/polymor x +

localhost/responsi_pbo/polymorphism.php

```
object(Company)#1 (1) { ["programmer"]=> object(Programmer)#2 (1) { ["name"]=> string(21)
"Zahra Nabila Melfiani" } } object(Company)#1 (1) { ["programmer"]=>
object(BackendProgrammer)#3 (1) { ["name"]=> string(21) "Zahra Nabila Melfiani" } }
object(Company)#1 (1) { ["programmer"]=> object(FrontendProgrammer)#2 (1) { ["name"]=>
string(21) "Zahra Nabila Melfiani" } } Hello Programmer Zahra Nabila Melfiani Hello Backend
Programmer Zahra Nabila Melfiani Hello Frontend Programmer Zahra Nabila Melfiani
```

Gambar 12 polymorphism.php

Penjelasan :

Code pada file polymorphism.php digunakan untuk mengimport class Programmer, BackendProgrammer, FrontendProgrammer, dan Company dari file programmer.php. kemudian digunakan untuk membuat objek Company dan mengganti tipe objek programmer, lalu memanggil fungsi sayHelloProgrammer dengan berbagai objek. Dimana menunjukkan konsep polimorfisme.

```

1  <?php
2
3  // import data/person.php
4  require "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new person("Zahra Nabila Melfiani","Kota Malang");
8
9  // manipulasi properti nama person
10 $person1->nama = "Zahra";
11
12 // menampilkan hasil
13 echo "nama = {$person1->nama}" . PHP_EOL;
14 echo "alamat = {$person1->alamat}" . PHP_EOL;
15 echo "negara = {$person1->negara}" . PHP_EOL;
16

```



Gambar 13 properti.php

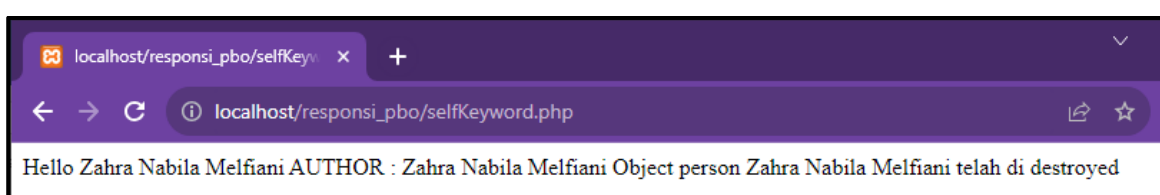
Penjelasan :

Code pada file properti.php, ditujukan untuk menampilkan manipulasi properti dari objek person. Kurang lebih sama dengan apa yang ditampilkan pada file object.php.

```

1  <?php
2
3  // import data/person.php
4  require "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new person("Zahra Nabila Melfiani","Kota Malang");
8
9  // panggil function
10 $person1->sayHello("Zahra Nabila Melfiani");
11
12 // panggil self keyword
13 $person1->info() . PHP_EOL;
14

```

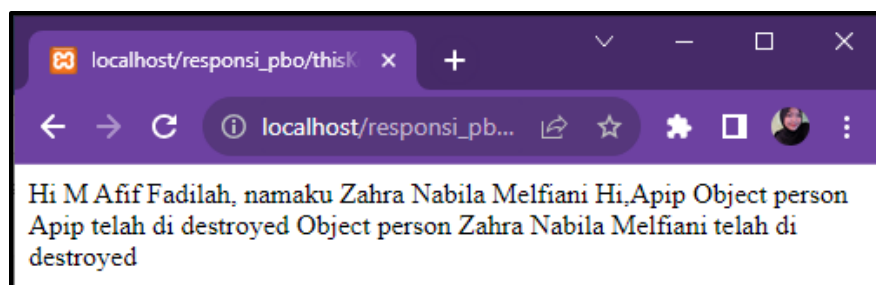


Gambar 14 selfKeyword.php

Penjelasan :

Code pada file selfKeyword.php, digunakan untuk mengimport class person dari file person.php. Lalu membuat objek person dan memanggil method info. Serta menunjukkan penggunaan dari self keyword.

```
thisKeyword.php
1  <?php
2
3  // import data/person.php
4  require "data/person.php";
5
6  // buat object dari kelas person
7  $person1 = new person("Zahra Nabila Melfiani", "Kota Malang");
8
9  // tambahkan value nama di object
10 $person1->nama = "Zahra Nabila Melfiani";
11
12 // panggil function sayHelloNull dengan parameter
13 $person1->sayHelloNull("M Afif Fadilah");
14
15 // buat object dari kelas person
16 $person2 = new person("M Afif Fadilah", "Rejang Lebong");
17
18 // tambahkan value nama di object
19 $person2->nama = "Apip";
20
21 // panggil function sayHelloNull dengan parameter null
22 $person2->sayHelloNull(null);
23
```

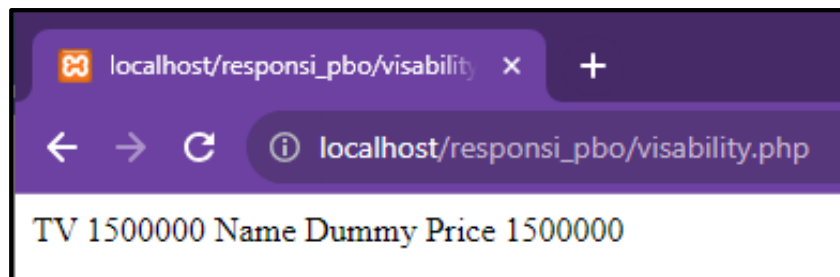


Gambar 15 thisKeyword.php

Penjelasan :

Code pada file thiskeyword.php, digunakan untuk mengimport class person dari file person.php. Kemudian membuat dua objek person dan memanggil method sayHelloNull. Serta menunjukkan penggunaan dari this keyword.

```
visibility.php
1  <?php
2
3  require_once "data/product.php";
4
5  $product = new Product("TV", 1500000);
6
7  // tampilkan product get name
8  echo $product-> getName() . PHP_EOL;
9  // tampilkan product get price
10 echo $product-> getPrice() . PHP_EOL;
11 $dummy = new ProductDummy("Dummy", 1500000);
12 $dummy->info();
```



Gambar 16 visibility.php

Penjelasan :

Code pada file visibility.php digunakan untuk mengimport class Product dan ProductDummy dari file Product.php. Lalu membuat objek Product dan ProductDummy, kemudian menampilkan informasi. Serta menunjukkan penggunaan dari visibility modifier (protected).