

RESPONSI
PROYEK PENGANTAR BASIS DATA



Disusun Oleh :

Nama : Epriantravolta Saragih
NPM : G1F022016

Asisten Dosen :

1. Fadia Nur Shafitri (G1F021010)
2. Alvin Indrawan (G1F021020)

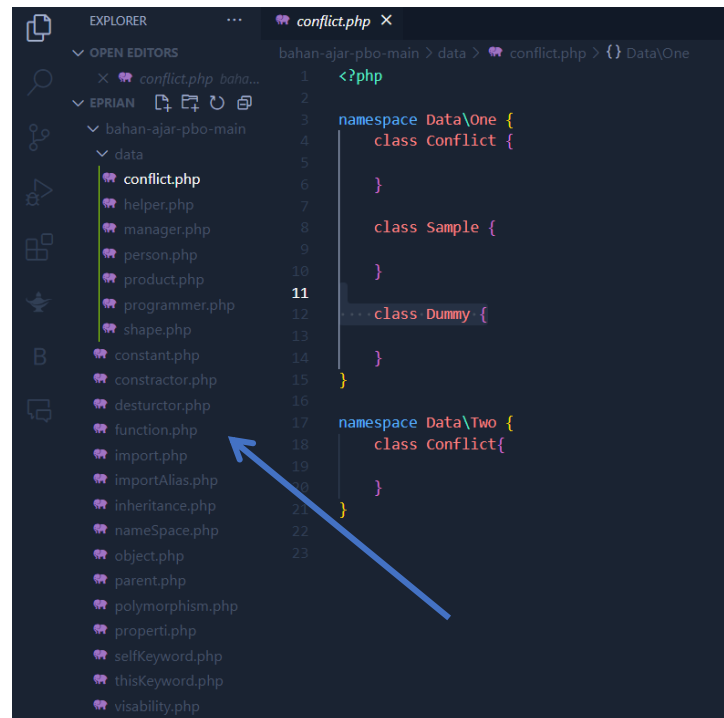
Dosen Pengampu :

1. Ferzha Putra Utama, S.T., M.Eng
2. Arie Vatesia, S.T., M.TI, Ph.D.

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS TEKNIK
UNIVERSITAS BENGKULU
T.A 2023/2024

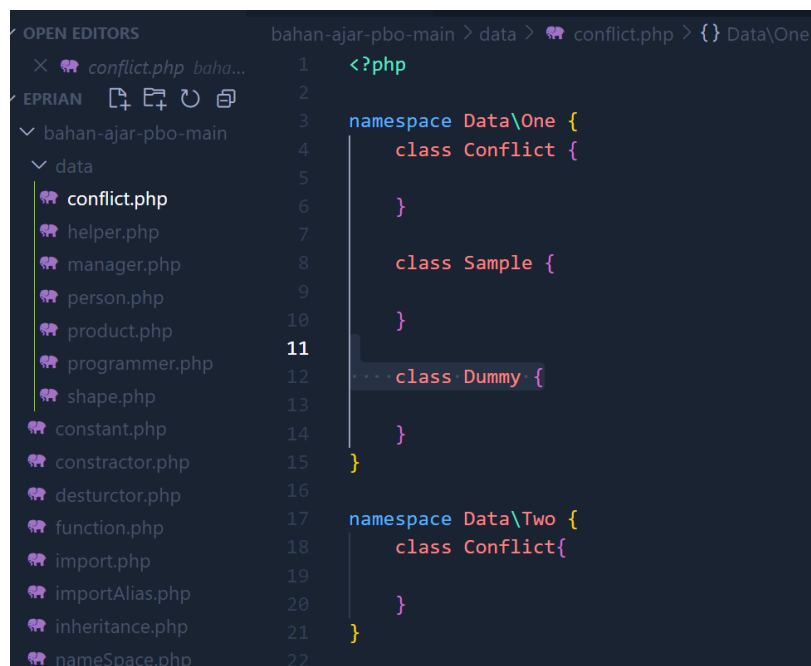
PEMBAHASAN

1. Tampilan Semua File



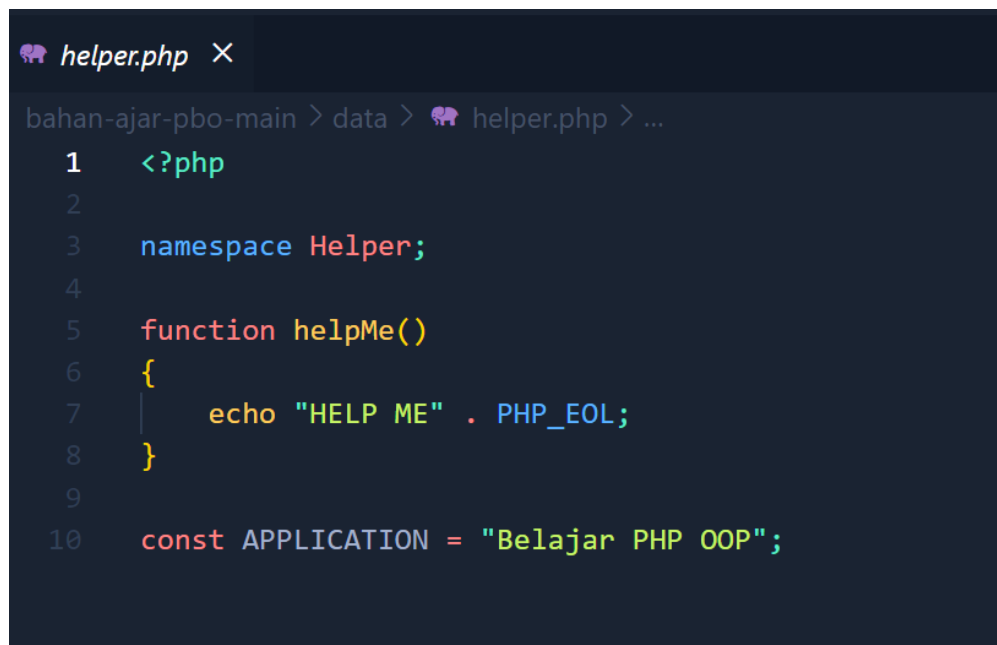
Gambar 1.1 File Kodingan

Dari daftar file PHP yang disebutkan, terdapat beberapa file yang berkaitan dengan konsep pemrograman berorientasi objek (PBO), seperti constructor.php, destructor.php, inheritance.php, namespace.php, object.php, parent.php, polymorphism.php, property.php, selfKeyword.php, thiskeyword.php dan visability.php. File-file tersebut kemungkinan berisi contoh implementasi dari konsep-konsep tersebut dalam pemrograman berorientasi objek.



Gambar 1.2 Source Code Conflict

Kode yang disediakan mendefinisikan dua namespace dan kelas di setiap namespace. Namun, cuplikan kode tidak menunjukkan penggunaan namespace dan kelas ini. Itu hanya mendefinisikan namespace dan kelas tanpa tindakan lebih lanjut. `Data\OneData\TwoConflict` Jika Anda ingin menggunakan namespace dan kelas ini, biasanya Anda perlu membuat instance kelas, memanggil metodenya, atau melakukan operasi lain dalam namespace yang ditentukan. Selain itu, Anda mungkin perlu menggunakan kelas dari namespace ini di bagian lain kode Anda. Kode yang disediakan adalah pengaturan dasar untuk mendefinisikan ruang nama dan kelas, namun tidak memiliki penggunaan atau demonstrasi ruang nama dan kelas yang ditentukan.



```
helper.php X
bahan-ajar-pbo-main > 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";
```

Gambar 1.3 Source Code Helper.php

Kode yang disediakan mendefinisikan namespace dengan fungsi dan konstanta . Fungsi ini hanya menggemakan string "HELP ME" diikuti dengan baris baru. Kata kunci digunakan untuk mendefinisikan namespace, yang merupakan cara mengelompokkan kode terkait dan menghindari konflik penamaan dengan kode lain. Kata kunci digunakan untuk mendefinisikan suatu fungsi, yaitu blok kode yang dapat dipanggil dan dieksekusi. Kata kunci digunakan untuk mendefinisikan konstanta, yaitu nilai yang tidak dapat diubah selama eksekusi program. *namespacefunctionconst*

Fungsi ini hanya menggemakan string "HELP ME" diikuti dengan baris baru menggunakan pernyataan tersebut. Konstanta didefinisikan dengan nilai *"Belajar PHP OOP"*. *helpMeechoAPPLICATION*

```

manager.php X
bahan-ajar-pbo-main > data > manager.php > ...
1  <?php
2
3  class Manager
4  {
5      var string $name;
6
7      var string $title;
8
9      public function __construct(string $name = "", string $title = "Manager"){
10         $this->name = $name;
11         $this->title = $title;
12     }
13
14     function sayHello(string $name): void
15     {
16         echo "Hi $name, my name is Manager $this->name" . PHP_EOL;
17     }
18 }
19
20 class VicePresident extends Manager
21 {
22
23     public function __construct(string $name = "")
24     {
25         // tidak wajib, tapi direkomendasikan
26         parent::__construct($name, "VP");
27     }
28
29     function sayHello(string $name): void
30     {
31         echo "Hi $name, my name is VP $this->name" . PHP_EOL;
32     }
33
34 }

```

Gambar 1.4 Source Code manager.php

Kode PHP yang disediakan mendefinisikan dua kelas, dan . Kelas tersebut memiliki konstruktor yang menginisialisasi properti and , dan metode yang menggemakan pesan ucapan. Kelas memperluas kelas dan mengganti metodenya. Kelas memiliki konstruktor yang mengatur properti and , dan metode yang menyambut seseorang yang menggunakan properti tersebut. Kelas memperluas kelas dan mengganti metode untuk memberikan pesan ucapan yang berbeda. Kode ini mendemonstrasikan penggunaan kelas, pewarisan, konstruktor, dan penggantian metode di PHP.

```

person.php X
bahan-ajar-pbo-main > data > person.php > ...
1  <?php
2
3  class Person
4  {
5      const AUTHOR = "Eprian";
6
7      var string $name;
8      var ?string $address = null;
9      var string $country = "Indonesia";
10
11     function __construct(string $name, ?string $address)
12     {
13         $this->name = $name;
14         $this->address = $address;
15     }
16
17     function sayHello(?string $name)
18     {
19         if (is_null($name)) {
20             echo "Hi, my name is $this->name" . PHP_EOL;
21         } else {
22             echo "Hi $name, my name is $this->name" . PHP_EOL;
23         }
24     }
25
26     function info()
27     {
28         echo "Author : " . self::AUTHOR . PHP_EOL;
29     }
30
31     function __destruct()
32     {
33         echo "Object person $this->name is destroyed" . PHP_EOL;
34     }
35 }

```

Gambar 1.5 Source Code person.php

Kode PHP yang disediakan mendefinisikan kelas dengan properti, metode, dan destruktur. Berikut rincian kodenya **Person**

1.Kelas :Person

- Mendefinisikan konstanta dengan nilai "Eprian".AUTHOR
- Berisi properti: \$namebertipe string , \$addressbertipe string atau null. , \$countrybertipe string dengan nilai default "Indonesia".
- Memiliki konstruktor yang menginisialisasi properti dan .__constructnameaddress
- Berisi cara menyapa seseorang berdasarkan nama yang diberikan.sayHello
- Termasuk metode yang menggemakan nilai konstanta .infoAUTHOR

-Mendefinisikan destruktur yang menggemakan pesan ketika objek dihancurkan.__destruct

Kode ini mendemonstrasikan penggunaan konstanta, properti, metode, dan destruktur dalam kelas PHP. Kata kunci digunakan untuk mendefinisikan konstanta kelas, dan kata kunci digunakan untuk mendefinisikan metode dalam kelas. Metode adalah konstruktor, dan metode adalah destruktur yang dipanggil secara otomatis.

```
product.php X
bahan-ajar-pbo-main > 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 }
```

Gambar 1.6 Source Code product.php

Kode PHP yang disediakan mendefinisikan dua kelas, dan . Kelas memiliki dua properti, dan , dan konstruktor yang menginisialisasi properti ini. Ia juga memiliki dua metode, dan , yang mengembalikan nilai properti masing-masing. Kelas memiliki dua properti, dan , yang dilindungi, artinya properti tersebut hanya dapat diakses di dalam kelas dan subkelasnya. Metode ini adalah konstruktor yang menginisialisasi properti dan . Metode and bersifat publik dan mengembalikan nilai properti and masing-masing.

Product \$name \$price __construct \$name \$price getName getPrice \$name \$price

Kelas memperluas kelas dan menambahkan metode yang menggemakan nilai properti dan

.ProductDummy Product info \$name \$price

```

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

```

Gambar 1.7 Source Code programmer.php

Kode PHP yang disediakan mendefinisikan kelas dan subkelasnya dan . Selain itu, ini mencakup kelas dan fungsi yang menyambut programmer berdasarkan tipenya yaitu *ProgrammerBackendProgrammerFrontendProgrammerCompany* dan *sayHelloProgrammer*

Kelas memiliki properti dan konstruktor yang menginisialisasi nama. Kelas dan memperluas kelas. Kelas memiliki properti bertipe . Fungsi ini menyapa programmer berdasarkan tipenya. Kode ini mendemonstrasikan penggunaan pewarisan, pengecekan tipe, dan polimorfisme di PHP.

```

bahan-ajar-pbo-main > data > 🐼 shape.php > ...
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 1.8 Source Code shape.php

Kode PHP yang disediakan mendefinisikan kelas dengan metode yang mengembalikan -1, dan kelas yang memperluas kelas tersebut. Kelas mengesampingkan metode tersebut dan menambahkan metode yang memanggil metode kelas induk. }.Kelas memiliki metode yang mengembalikan -1. Kelas memperluas kelas dan mengganti metode untuk mengembalikan 4. Kelas ini juga menyertakan metode yang memanggil metode kelas induk menggunakan kata kunci *Shape get CornerRectangle Shape get Corner get Paren tCorner get Corner parent*.

```

bahan-ajar-pbo-main > 🐼 constant.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Person.php";
5
6  // buat define
7  define("APPLICATION", "Belajar PHP OOP");
8
9  // buat const app version
10 const APP_VERSION = "1.0.0";
11
12 // tampilkan hasil
13 echo APPLICATION . PHP_EOL;
14 echo APP_VERSION . PHP_EOL;
15 echo Person::AUTHOR . PHP_EOL;
16

```

Gambar 1.9 Source Code constant.php

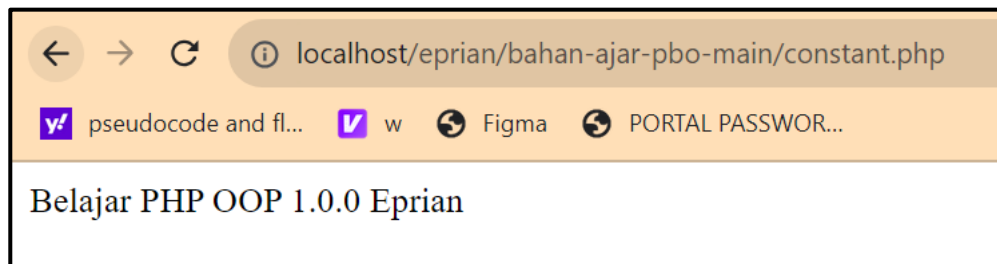
Kode PHP yang disediakan menunjukkan penggunaan konstanta dan konstanta kelas. Itu juga mengimpor kelas Person dan membuat objek baru dari kelas Person. Berikut rincian kodenya:

Pernyataan `require_once` digunakan untuk mengimpor kelas Person dari file `data/Person.php`.

Fungsi `define` digunakan untuk mendefinisikan sebuah konstanta bernama `APPLICATION` dengan nilai "Belajar PHP OOP".

Kata kunci `const` digunakan untuk mendefinisikan konstanta kelas `APP_VERSION` dengan nilai "1.0.0".

Pernyataan `echo` digunakan untuk menampilkan nilai konstanta yang ditentukan dan konstanta kelas.



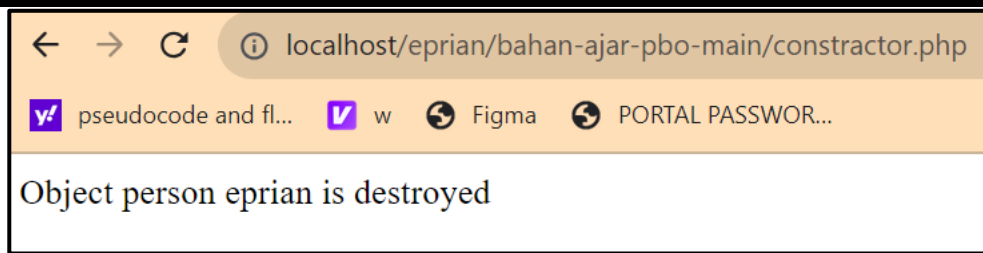
Gambar 1.10 Output constant.php

Gambar diatas merupakan output dari constant.php yang dimana isinya “Belajar PHP OOP 1.0.0 Eprian”

```
bahan-ajar-pbo-main > constructor.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Person.php";
5
6  // buat object new person dengan 2 parameter
7  $eprian = new Person("eprian", "siantar");
8
9  // vardump object
10
11
```

Gambar 1.11 Source Code constructor.php

Kode PHP yang disediakan mengimpor kelas Person dan membuat objek baru dari kelas Person dengan dua parameter. Namun, kode tersebut tidak menyertakan pernyataan `vardump` untuk menampilkan objek. Berikut adalah versi kode terbaru yang menyertakan `vardump`. Kode ini mengimpor kelas Person dan membuat objek baru dari kelas Person dengan dua parameter. Pernyataan `vardump` digunakan untuk menampilkan objek dan propertinya.



Gambar 1.12 Output constructor.php

Gambar diatas merupakan output dari constructor.php yang dimana isinya “Object person eprian is destroyed”

```
bahan-ajar-pbo-main > destrutor.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Person.php";
5
6  // buat 2 object new peson dengan parameter yang berbeda
7  $eprian = new Person("eprian", "siantar");
8  $volta = new Person("volta", "siantar");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
12
```

Gambar 1.13 Source Code destructor.php

Kode PHP yang diberikan mengimpor kelas Person dari file data/Person.php dan membuat dua instance baru dari kelas Person dengan parameter berbeda. Setelah itu, akan muncul bunyi "Program Selesai" untuk menandakan berakhirnya program.

Penjelasan setiap code sebagai berikut :

Mengimpor kelas menggunakan require_once.

Membuat instance baru dari suatu kelas dengan parameter berbeda.

Menggunakan pernyataan echo untuk mengeluarkan pesan.

Kode ini menampilkan penggunaan dasar mengimpor kelas dan membuat objek di PHP.



Gambar 1.14 Output destructor.php

Gambar diatas merupakan output dari destructor.php yang dimana isinya “Program Selesai Object person volta is destroyed Object person eprian is destroyed”

```

bahan-ajar-pbo-main > function.php > ...
1  <?php
2
3  require_once "data/Person.php";
4
5  $person1 = new Person("Eprian", "Siantar");
6  $person1->name = "Eprian";
7  $person1->sayHello("Eprian");
8
9  $person1->info();|

```

Gambar 1.15 Source Code Function.php

Kode PHP yang disediakan mengimpor kelas Person dan membuat dua instance baru dari kelas Person dengan parameter berbeda. Setelah itu, akan muncul bunyi "Program Selesai" untuk menandakan berakhirnya program.

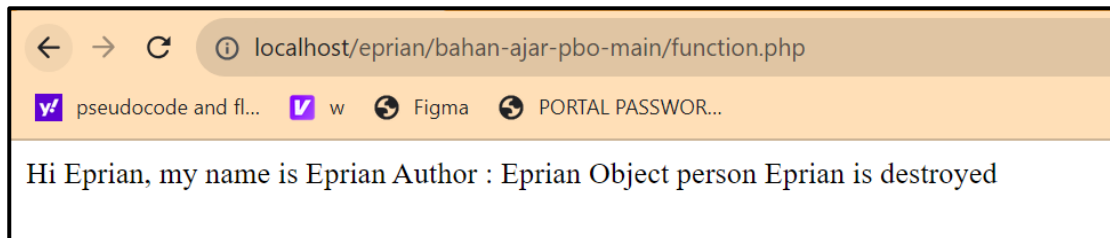
Cuplikan kode menunjukkan hal berikut:

Mengimpor kelas menggunakan require_once.

Membuat instance baru dari suatu kelas dengan parameter berbeda.

Menggunakan pernyataan echo untuk mengeluarkan pesan.

Kode ini menampilkan penggunaan dasar mengimpor kelas dan membuat objek di PHP.



Gambar 1.16 Output function.php

Gambar diatas merupakan output dari function.php yang dimana isinya "Hi Eprian, my name is Eprian Author : Eprian Object person Eprian is destroyed"

```

bahan-ajar-pbo-main > import.php > ...
1  <?php
2
3  require_once "data/Conflict.php";
4  require_once "data/Helper.php";
5
6  use Data\One\Conflict;
7  use function Helper\helpMe;
8  use const Helper\APPLICATION;
9
10 $conflict1 = new Conflict();
11 $conflict2 = new Data\Two\Conflict();
12
13 helpMe();
14
15 echo APPLICATION . PHP_EOL;

```

Gambar 1.17 Source Code import.php

Kode PHP yang disediakan menunjukkan penggunaan mengimpor kelas dan konstanta dari namespace yang berbeda. Itu juga membuat dua instance kelas dan memanggil metodenya. Berikut rincian kodenya:

Pernyataan `require_once` digunakan untuk mengimpor kelas `Conflict` dan fungsi `Helper` dari masing-masing file `data/Conflict.php` dan `data/Helper.php`.

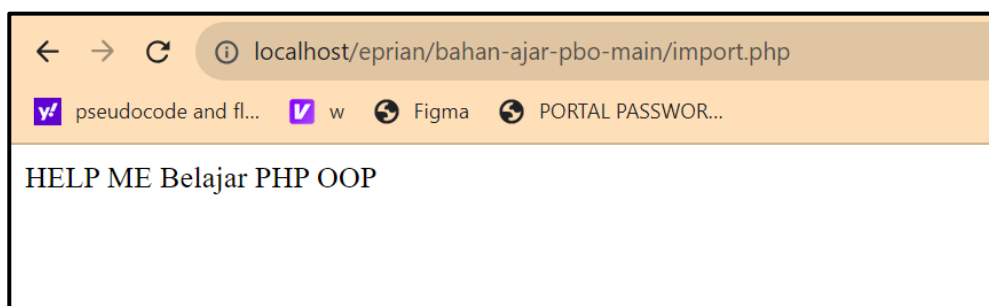
Kata kunci `use` digunakan untuk mengimpor kelas `Conflict` dan fungsi `helpMe` dari namespace yang ditentukan.

Dua contoh kelas Konflik dibuat dengan kata kunci baru, satu dari namespace `Data\One` dan satu lagi dari namespace `Data\Two`.

Fungsi `helpMe` dipanggil menggunakan operator `()`.

Pernyataan `echo` digunakan untuk menampilkan nilai konstanta `APPLICATION`.

Kode ini mendemonstrasikan penggunaan mengimpor kelas dan fungsi dari namespace berbeda dan membuat instance kelas untuk memanggil metodenya.



Gambar 1.18 Output import.php

Gambar diatas merupakan output dari `importAlias.php` yang dimana isinya “HELP ME Belajar PHP OOP”

```

bahan-ajar-pbo-main > 🐞 importAlias.php > ...
1  <?php
2
3  require_once "data/Conflict.php";
4  require_once "data/Helper.php";
5
6  use Data\One\Conflict as Conflict1;
7  use Data\Two\Conflict as Conflict2;
8  use function Helper\helpMe as help;
9  use const Helper\APPLICATION as APP;
10
11 $conflict1 = new Conflict1();
12 $conflict2 = new Conflict2();
13
14 help();
15
16 echo APP . PHP_EOL;

```

Gambar 1.19 Source Code importAlias.php

Kode PHP yang disediakan mengimpor kelas Conflict dari dua namespace berbeda menggunakan alias, mengimpor fungsi helpMe dari namespace Helper menggunakan alias, dan mengimpor konstanta APPLICATION dari namespace Helper menggunakan alias. Kemudian membuat dua instance kelas Conflict dan memanggil fungsi helpMe dan menampilkan nilai konstanta APPLICATION. Pada source code ini hampir sama helpme.php.



Gambar 1.20 Output import.php

Gambar diatas merupakan output dari importAlias.php yang dimana isinya “HELP ME Belajar PHP OOP”

```

bahan-ajar-pbo-main > 🐞 inheritance.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Manager.php";
5
6  // buat object new manager dan tambahkan value nama kemudian panggil function
7  $manager = new Manager();
8  $manager->name = "eprian";
9  $manager->sayHello("volta");
10
11 // buat object new vicepresident dan tambahkan value nama kemudian panggil function
12 $vp = new VicePresident();
13 $vp->name = "epri";
14 $vp->sayHello("volta");
15

```

Gambar 1.21 Source code inheritance.php

Kode PHP yang disediakan mengimpor kelas Manajer dan Wakil Presiden dan membuat dua instance dari masing-masing kelas, mengatur properti nama dan memanggil metode sayHello. Berikut rincian kodenya:

Pernyataan `require_once` digunakan untuk mengimpor kelas Manager dan VicePresident dari masing-masing file `data/Manager.php` dan `data/VicePresident.php`.

Dua contoh kelas Manager dibuat dengan kata kunci baru, satu dengan properti nama disetel ke "eprian" dan yang lainnya dengan properti nama disetel ke "volta". Metode `sayHello` dipanggil pada setiap instance, meneruskan "volta" sebagai parameternya.

Dua contoh kelas VicePresident dibuat dengan kata kunci baru, satu dengan properti nama disetel ke "epri" dan yang lainnya dengan properti nama disetel ke "volta". Metode `sayHello` dipanggil pada setiap instance, meneruskan "volta" sebagai parameternya.

Kode ini menunjukkan penggunaan mengimpor kelas dan membuat instance kelas Manajer dan Wakil Presiden, mengatur properti, dan metode pemanggilan.



Gambar 1.22 Output inheritance.php

Gambar diatas merupakan output inheritance.php yang dimana isinya "Hi volta, my name is Manager eprian Hi volta, my name is VP epri"

```
bahan-ajar-pbo-main > namespace.php > ...
1  <?php
2
3  namespace {
4
5      require_once "data/Conflict.php";
6      require_once "data/Helper.php";
7
8      $conflict1 = new Data\One\Conflict();
9      $conflict2 = new Data\Two\Conflict();
10
11      echo Helper\APPLICATION . PHP_EOL;
12
13      Helper\helpMe();
14
15  }
16
```

Gambar 1.23 Source code namespace.php

Kode PHP yang disediakan menunjukkan impor kelas dan konstanta dari namespace yang berbeda. Itu juga membuat instance kelas Conflict dari dua namespace berbeda dan memanggil konstanta APPLICATION. Berikut rincian kodenya:

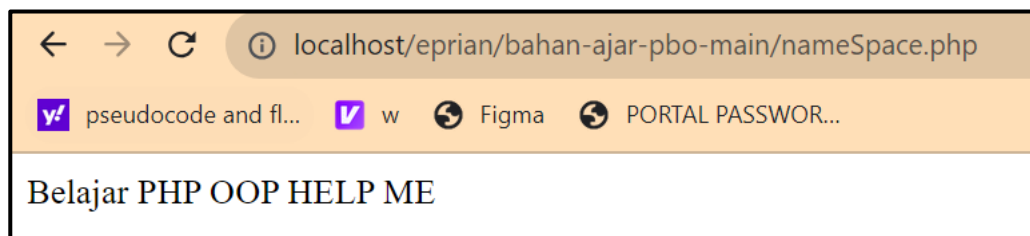
Pernyataan `require_once` digunakan untuk mengimpor kelas Conflict dari dua namespace berbeda, `Data\One` dan `Data\Two`.

Dua contoh kelas Konflik dibuat dengan kata kunci `new`, satu dari namespace `Data\One` dan satu lagi dari namespace `Data\Two`.

Konstanta `APPLICATION` ditampilkan menggunakan pernyataan `echo`.

Fungsi `helpMe` dipanggil menggunakan operator `()`.

Kode ini mendemonstrasikan penggunaan mengimpor kelas dari namespace berbeda dan membuat instance kelas Konflik, menampilkan nilai konstanta `APPLICATION`, dan memanggil fungsi `helpMe`.



Gambar 1.24 Output inheritance.php

Gambar diatas merupakan output inheritance.php yang dimana isinya “Belajar PHP OOP HELP ME”

```
bahan-ajar-pbo-main > object.php > ...
1  <?php
2
3  require_once "data/person.php";
4
5  $person = new Person("eprian","siantar");
6  $person->name = "eprian";
7  $person->address = "siantar";
8  $person->country = "portugal";
9
10 $person2 = new person("volta", "siantar");
11 $person2->name = "volta";
12 $person2->address = "siantar";
13 $person2->country = "portugal";
14
15 // menampilkan hasil
16 echo "nama = {$person->name}" . PHP_EOL;
17 echo "alamat = {$person->address}" . PHP_EOL;
18 echo "negara = {$person->country}" . PHP_EOL;
19
20 echo "nama = {$person2->name}" . PHP_EOL;
21 echo "alamat = {$person2->address}" . PHP_EOL;
22 echo "negara = {$person2->country}" . PHP_EOL;
```

Gambar 1.25 Source code object.php

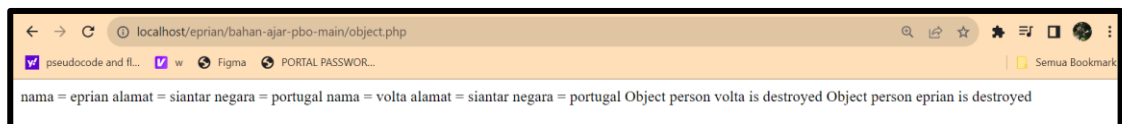
Pernyataan `require_once` digunakan untuk mengimpor kelas `Person` dari file `data/person.php`.

Dua instance kelas `Person` dibuat dengan kata kunci baru, satu dengan parameter "eprian" dan "siantar" dan yang lainnya dengan parameter "volta" dan "siantar".

Nama, alamat, dan properti negara setiap instance diatur menggunakan operator `->`.

Pernyataan `echo` digunakan untuk menampilkan nilai properti nama, alamat, dan negara dari setiap instance.

Kode ini mendemonstrasikan penggunaan pembuatan instance kelas `Person`, mengatur properti, dan menampilkan nilainya.



Gambar 1.26 Output object.php

Gambar diatas merupakan output object.php yang dimana isinya “nama = eprian alamat = siantar negara = portugal nama = volta alamat = siantar negara = portugal Object person volta is destroyed Object person eprian is destroyed”

```
ahan-ajar-pbo-main > parent.php > ...
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 1.27 Source code parent.php

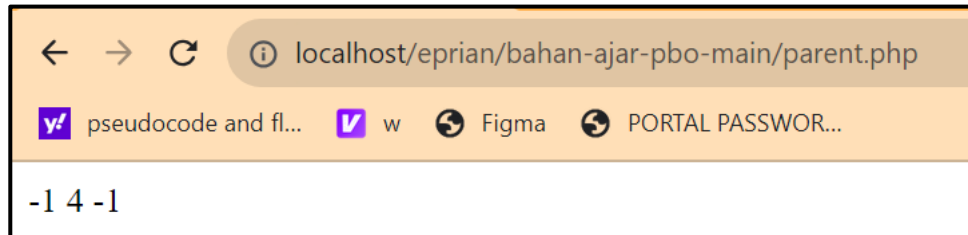
Pernyataan `require_once` digunakan untuk mengimpor kelas `Shape` dan `Rectangle` dari file `data/Shape.php`.

Kata kunci `use` digunakan untuk mengimpor kelas `Bentuk` dan `Persegi Panjang` dari namespace `Data` menggunakan sintaks kurung kurawal.

Sebuah instance dari kelas `Shape` dibuat dengan kata kunci `new`, dan metode `getCornernya` dipanggil menggunakan pernyataan `echo`.

Sebuah instance dari kelas `Rectangle` dibuat dengan kata kunci `new`, dan metode `getCorner` dan `getParentCornernya` dipanggil menggunakan pernyataan `echo`.

Kode ini mendemonstrasikan penggunaan mengimpor kelas dari namespace berbeda dan membuat instance kelas tersebut, memanggil metodenya, dan menampilkan nilainya.



Gambar 1.28 Output parent.php

Gambar diatas merupakan output parent.php yang dimana isinya “-1 4 -1 “

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

Gambar 1.29 Source code polymorphism.php

Pernyataan `require_once` digunakan untuk mengimpor kelas `Programmer`, `BackendProgrammer`, dan `FrontendProgrammer` dari file `data/Programmer.php`.

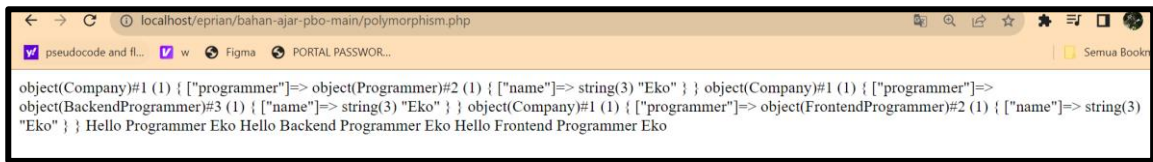
Sebuah instance dari kelas Perusahaan dibuat dengan kata kunci `new`, dan properti pemrogramnya diatur menggunakan operator `->`.

Contoh kelas `BackendProgrammer` dan `FrontendProgrammer` dibuat dengan kata kunci `new`, dan properti pemrogramnya disetel menggunakan operator `->`.

Fungsi `var_dump` digunakan untuk menampilkan instance kelas Perusahaan.

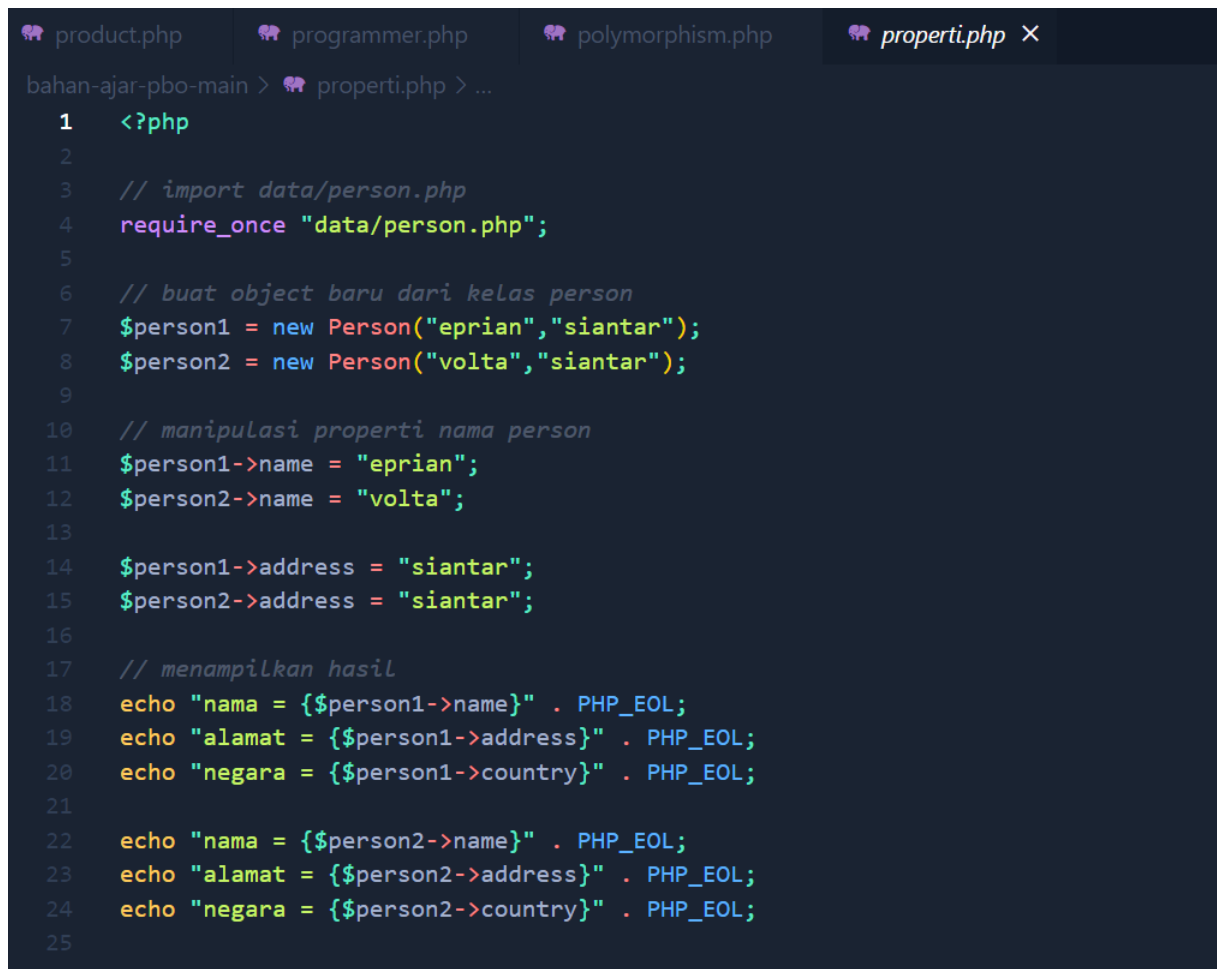
Fungsi `sayHelloProgrammer` dipanggil dengan instance kelas `Programmer`, `BackendProgrammer`, dan `FrontendProgrammer` sebagai parameternya.

Kode ini mendemonstrasikan penggunaan mengimpor kelas dari namespace berbeda, membuat instance kelas tersebut, mengatur properti, dan memanggil fungsi yang ditentukan di kelas lain.



Gambar 1.30 Output polymorphism.php

Gambar diatas merupakan output polymorphism.php yang dimana isinya “object(Company)#1 (1) { [“programmer”]=> object(Programmer)#2 (1) { [“name”]=> string(3) “Eko” } } object(Company)#1 (1) { [“programmer”]=> string(3) “Eko” } } object(Company)#1 (1) { [“programmer”]=> object(BackendProgrammer)#3 (1) { [“name”]=> string(3) “Eko” } } object(Company)#1 (1) { [“programmer”]=> object(FrontendProgrammer)#2 (1) { [“name”]=> string(3) “Eko” } } Hello Programmer Eko Hello Backend Programmer Eko Hello Frontend Programmer Eko “



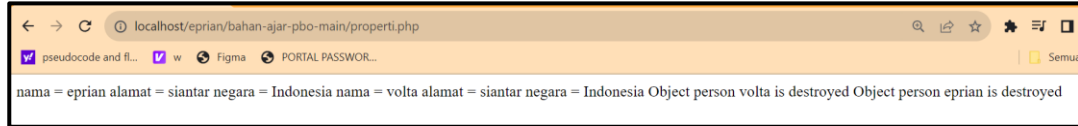
Gambar 1.31 Source code properti.php

Pernyataan `require_once` digunakan untuk mengimpor kelas `Person` dari file `data/person.php`.

Dua instance kelas Person dibuat dengan kata kunci baru, satu dengan parameter "eprian" dan "siantar" dan yang lainnya dengan parameter "volta" dan "siantar".

Nama, alamat, dan properti negara setiap instance diatur menggunakan operator ->.

Pernyataan echo digunakan untuk menampilkan nilai properti nama, alamat, dan negara dari setiap instance.



Gambar 1.32 Output properti.php

Gambar diatas merupakan output properti.php yang dimana isinya “nama = eprian alamat = siantar negara = Indonesia nama = volta alamat = siantar negara = Indonesia Object person volta is destroyed Object person eprian is destroyed “

```
bahan-ajar-pbo-main > selfKeyword.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new Person("eprian","siantar");
8
9  // panggil function
10 $person1->sayHello("eprian");
11
12 // panggil self keyword
13 $person1->info();
14
```

Gambar 1.33 Source code self.Keyword.php

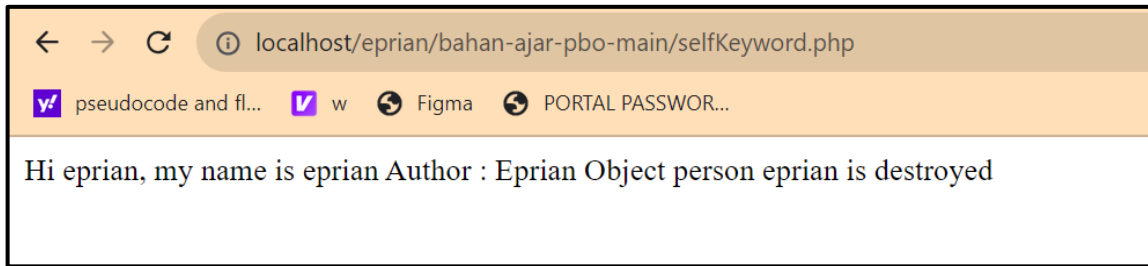
Pernyataan require_once digunakan untuk mengimpor kelas Person dari file data/person.php.

Sebuah instance baru dari kelas Person dibuat dengan kata kunci baru dan parameter "eprian" dan "siantar".

Metode sayHello pada kelas Person dipanggil dengan parameter "eprian" menggunakan operator ->.

Metode info kelas Person dipanggil menggunakan operator ->.

Kode ini mendemonstrasikan penggunaan pembuatan instance kelas Person, memanggil metodenya, dan menampilkan nilainya.



Gambar 1.34 Source code self.Keyword.php

Gambar diatas merupakan output properti.php yang dimana isinya “ Hi eprian, my name is eprian Author : Eprian Object person eprian is destroyed”

```
bahan-ajar-pbo-main > thisKeyword.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object dari kelas person
7  $eprian = new Person("eprian", "siantar");
8
9  // tambahkan value nama di object
10 $eprian->name = "eprian";
11
12 // panggil function sayHelloNull dengan parameter
13 $eprian->sayHello("volta");
14
15 // buat object dari kelas person
16 $volta = new Person("volta", "siantar");
17
18 // tambahkan value nama di object
19 $volta->name = "volta";
20
21 // panggil function sayHelloNull dengan parameter null
22 $volta->sayHello(null);
23
```

Gambar 1.35 Source code .thisKeyword.php

The require_once statement is used to import the Person class from the data/person.php file.

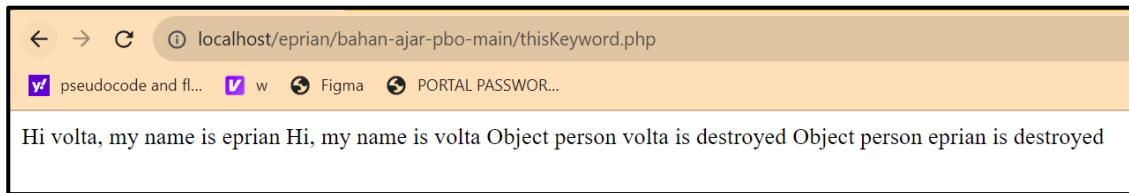
Two instances of the Person class are created with the new keyword, one with the parameters "eprian" and "siantar" and the other with the parameters "volta" and "siantar".

The name property of each instance is set using the -> operator.

The sayHello method of the Person class is called with the parameter "volta" for the eprian instance and with the parameter null for the volta instance using the -> operator.

The name property of each instance is displayed using the echo statement.

This code demonstrates the usage of creating instances of the Person class, setting properties, calling methods with different parameters, and displaying their values.



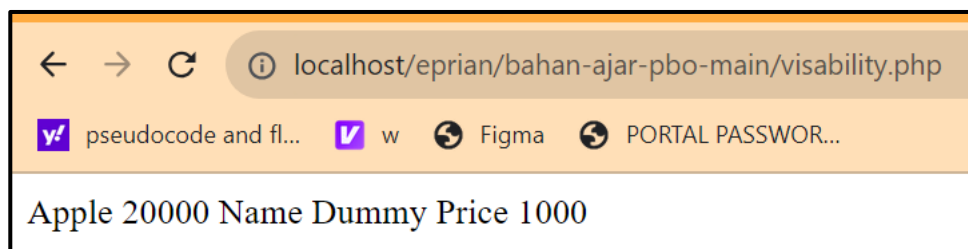
Gambar 1.36 Source code thisKeyword.php

Gambar diatas merupakan output properti.php yang dimana isinya “ Hi eprian, my name is eprian Author : Eprian Object person eprian is destroyed”

```
bahan-ajar-pbo-main > visibility.php > ...
1  <?php
2
3  require_once "data/Product.php";
4
5  $product = new Product("Apple", 20000);
6
7  echo $product->getName() . PHP_EOL;
8  echo $product->getPrice() . PHP_EOL;
9
10 $dummy = new ProductDummy("Dummy", 1000);
11 $dummy->info();
```

Gambar 1.37 Source code .visibility.php

Kode yang diberikan membuat instance kelas Produk, meneruskan nama "Apple" dan harga 20000, lalu memanggil metode getName() dan getPrice() untuk mencetak masing-masing nama dan harga produk. Ia kemudian membuat sebuah instance dari kelas ProductDummy, meneruskan nama "Dummy" dan harga 1000, dan memanggil metode info() di dalamnya. Namun, kode tersebut tidak menunjukkan implementasi kelas Product dan ProductDummy, sehingga tidak mungkin untuk menentukan apa yang dilakukan metode info() atau apakah metode tersebut ada.



Gambar 1.38 Source code visibility.php

Gambar diatas merupakan output visibility.php yang dimana isinya “Apple 20000 Name Dummy Price 1000”