

# Responsi Proyek PBO

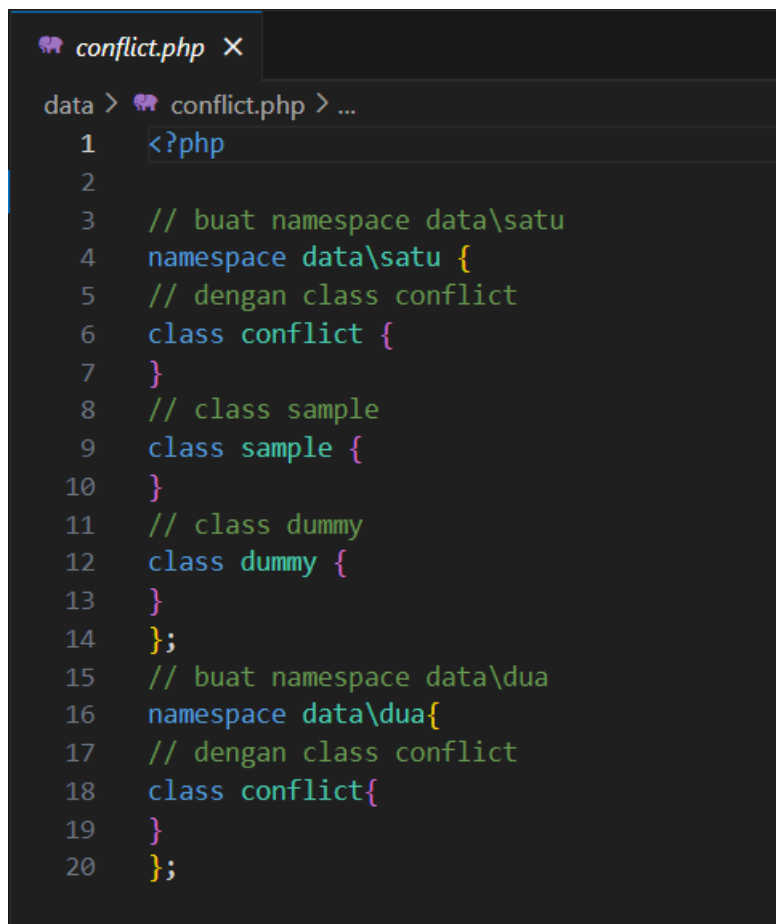
Nama : Arief Haris Prasetyo Rizaldi

NPM : G1F022073

Kelas : Sistem Informasi -A

Matkul : Proyek Pemrograman Berorientasi Objek

## Conflict.php



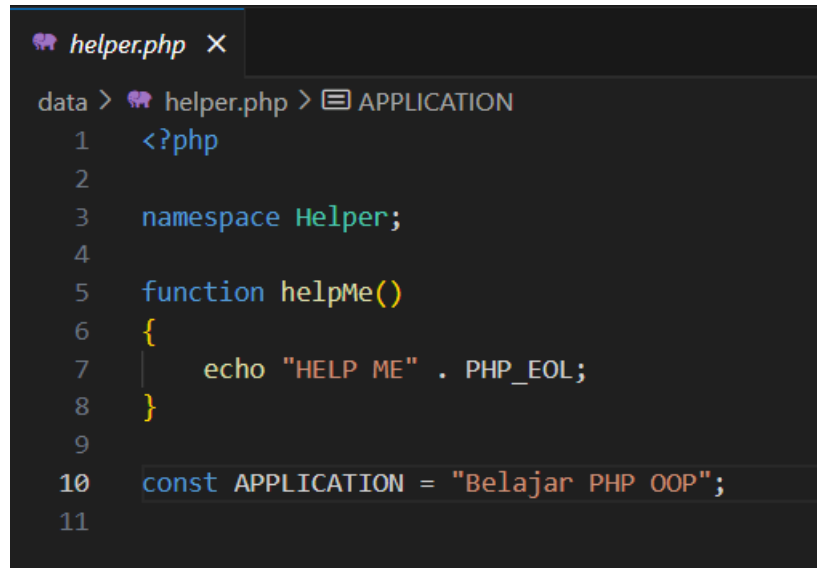
```
1 <?php
2
3 // buat namespace data\satu
4 namespace data\satu {
5 // dengan class conflict
6 class conflict {
7 }
8 // class sample
9 class sample {
10 }
11 // class dummy
12 class dummy {
13 }
14 };
15 // buat namespace data\dua
16 namespace data\dua{
17 // dengan class conflict
18 class conflict{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

### Penjelasan:

Kode PHP di atas membuat dua namespace, yaitu data\satu dan data\dua. Di dalam namespace data\satu, terdapat tiga kelas, yaitu conflict, sample, dan dummy. Namun, perlu diperhatikan bahwa terdapat kelas dengan nama conflict di dalam namespace data\satu. Selanjutnya, di dalam namespace data\dua, juga didefinisikan kelas conflict. Ini menciptakan konflik nama kelas antar-namespace karena ada dua kelas dengan nama yang sama (conflict) di dua namespace yang berbeda. Konsep namespace digunakan dalam PHP untuk

mengelompokkan kode, mencegah konflik nama, dan memungkinkan penggunaan nama kelas yang sama tanpa bertabrakan di dalam namespace yang berbeda.

## Helper.php



```
data > helper.php > APPLICATION
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";
11
```

### Penjelasan:

Kode PHP di atas menggunakan namespace Helper. Di dalam namespace tersebut, terdapat dua elemen:

- Fungsi `helpMe()`: Ini adalah sebuah fungsi bernama `helpMe()` yang, ketika dipanggil, akan mencetak teks "HELP ME" ke layar dengan tambahan baris baru (`PHP_EOL`).
- Konstanta `APPLICATION`: Sebuah konstanta dengan nama `APPLICATION` yang memiliki nilai string "Belajar PHP OOP". Konstanta ini dapat diakses di dalam namespace Helper dan memiliki nilai yang tetap dan tidak dapat diubah selama jalannya program.

Penggunaan namespace membantu mengorganisir kode dan menghindari konflik nama dengan kode lain di proyek yang lebih besar. Dengan demikian, elemen-elemen di dalam namespace dapat diakses menggunakan sintaks namespace, misalnya `Helper\helpMe()` untuk memanggil fungsi `helpMe()` atau `Helper\APPLICATION` untuk mengakses nilai konstanta `APPLICATION`.

# Manager.php

```
manager.php X
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 $nama)
9      {
10         echo "Hi $nama, Namaku {$this->nama}" . PHP_EOL;
11     }
12 }
13
14 // buat kelas VicePresident dengan extends manager
15 class VicePresident extends Manager
16 {
17 }
18
```

## Penjelasan:

Kode PHP di atas mendefinisikan dua kelas, yaitu Manager dan VicePresident.

- Kelas Manager: Kelas ini memiliki sebuah properti nama dengan tipe data string, dan sebuah fungsi sayHello yang menerima parameter string nama. Fungsi sayHello mencetak pesan sapaan yang mencakup nilai parameter nama dan nilai properti nama dari objek yang memanggil fungsi tersebut. Properti nama dideklarasikan menggunakan tipe data string.
- Kelas VicePresident: Kelas ini diwarisi (extends) dari kelas Manager, sehingga memiliki semua properti dan fungsi yang dimiliki oleh kelas Manager. Dalam hal ini, kelas VicePresident tidak mendefinisikan properti atau fungsi tambahan. Dengan adanya pewarisan, objek dari kelas VicePresident dapat mengakses properti dan fungsi dari kelas Manager.

Pewarisan (inheritance) memungkinkan kelas untuk memanfaatkan atau memperluas fungsionalitas dari kelas lain. Dalam hal ini, VicePresident dapat menggunakan properti nama dan fungsi sayHello yang telah didefinisikan di kelas Manager.

# Person.php

```
person.php X
data > person.php > Person > AUTHOR
1  <?php
2
3  // membuat kelas person
4  class Person
5  {
6      // membuat properti
7      var string $nama;
8      // gunakan nullable properti
9      var ?string $alamat = null;
10     // gunakan default value untuk properti
11     var string $negara = "Indonesia";
12     // buat function sayHello
13     function sayHello(string $nama)
14     {
15         echo "Hi $nama" . PHP_EOL;
16     }
17     // buat function sayHello nullable dengan percabangan
18     function sayHelloNull(?string $nama)
19     {
20         if (is_null($nama)) {
21             echo "Hi, Namaku $this->nama" . PHP_EOL;
22         } else {
23             echo "Hi $nama, Namaku $this->nama" . PHP_EOL;
24         }
25     }
26
27     // buat const author
28     const AUTHOR = "Arief";
29     // buat function info untuk self keyword
30     function info()
31     {
32         echo "Author : " . self::AUTHOR . PHP_EOL;
33     }
34     // buat function constructor
35     function __construct(string $nama, ?string $alamat)
36     {
37         $this->nama = $nama;
38         $this->alamat = $alamat;
39     }
40
41     // buat function destructor
42     function __destruct()
43     {
44         echo "Object person $this->nama is destroyed" . PHP_EOL;
45     }
46 }
47
```

## Penjelasan:

Kode PHP di atas mendefinisikan kelas Person yang memiliki properti seperti \$nama, \$alamat, dan \$negara. Beberapa properti menggunakan fitur-fitur khusus, seperti properti \$alamat yang dapat bernilai null dan properti \$negara dengan nilai default "Indonesia". Terdapat juga metode seperti sayHello untuk mencetak pesan sapaan, sayHelloNull dengan

penanganan nullable parameter menggunakan percabangan, info untuk mencetak informasi penulis melalui konstanta kelas AUTHOR dengan kata kunci self. Selain itu, terdapat constructor `__construct` untuk inialisasi objek dan destructor `__destruct` untuk memberikan pesan saat objek dihancurkan. Konsep nullable, default value, constant, self keyword, constructor, dan destructor digunakan dalam kode ini untuk mengilustrasikan fitur-fitur dasar dalam pemrograman berorientasi objek menggunakan PHP.

## Product.php

```
product.php x
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 }
```

### Penjelasan:

Kode PHP di atas mendefinisikan dua kelas, yaitu Product dan ProductDummy.

- Kelas Product:

- Kelas ini memiliki dua properti proteksi, yaitu \$name (string) dan \$price (int), yang mewakili nama dan harga produk.
- Terdapat metode konstruktor \_\_construct yang dijalankan saat objek dibuat untuk menginisialisasi nilai properti \$name dan \$price berdasarkan parameter yang diterima.
- Metode getName() dan getPrice() digunakan untuk mengakses nilai properti \$name dan \$price, masing-masing dengan tipe data kembalian string dan int.
- Kelas ProductDummy:
  - Kelas ini mewarisi kelas Product, sehingga memiliki akses ke properti proteksi dan metode-metode dari kelas tersebut.
  - Terdapat metode info() yang mencetak informasi nama dan harga produk menggunakan properti yang diwarisi dari kelas Product.

Kelas Product dirancang sebagai kelas dasar untuk merepresentasikan produk dengan properti dasar seperti nama dan harga, sementara ProductDummy menunjukkan bagaimana kelas dapat diwarisi untuk menambahkan metode tambahan atau mengubah perilaku kelas dasar. Pemahaman pewarisan dan hak akses properti dalam konteks ini adalah kunci untuk memahami struktur dan hubungan antar kelas dalam pemrograman berorientasi objek menggunakan PHP.

# Programmer.php

```
programmer.php X
data > programmer.php > ...
1  <?php
2
3  class Programmer
4  {
5
6      public string $name;
7
8      public function __construct(string $name)
9      {
10         $this->name = $name;
11     }
12 }
13
14
15 class BackendProgrammer extends Programmer
16 {
17 }
18
19 class FrontendProgrammer extends Programmer
20 {
21 }
22
23 class Company
24 {
25     public Programmer $programmer;
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 }
```

## Penjelasan:

Kode PHP di atas menetapkan hierarki kelas yang terkait dengan pemrograman, dimulai dengan kelas dasar Programmer yang memiliki properti nama. Dua kelas turunan, BackendProgrammer dan FrontendProgrammer, mewarisi sifat-sifat kelas dasar tanpa penambahan atau perubahan. Kelas Company memiliki properti programmer yang bertipe Programmer, menunjukkan kemampuan untuk menyimpan objek dari kelas apapun yang mewarisi atau setara dengan kelas Programmer. Fungsi sayHelloProgrammer menggambarkan polimorfisme dengan menerima parameter bertipe Programmer dan mencetak pesan sapaan yang berbeda berdasarkan jenis pemrogram yang diberikan sebagai argumen, menggunakan pernyataan if dan instanceof. Ini menggambarkan konsep pewarisan, polimorfisme, dan penggunaan tipe data objek dalam pemrograman berorientasi objek dengan PHP.

## Shape.com

```
shape.php X
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 }
```

### Penjelasan:

Kode PHP di atas mendefinisikan dua kelas, yaitu Shape dan Rectangle, dalam namespace Data.

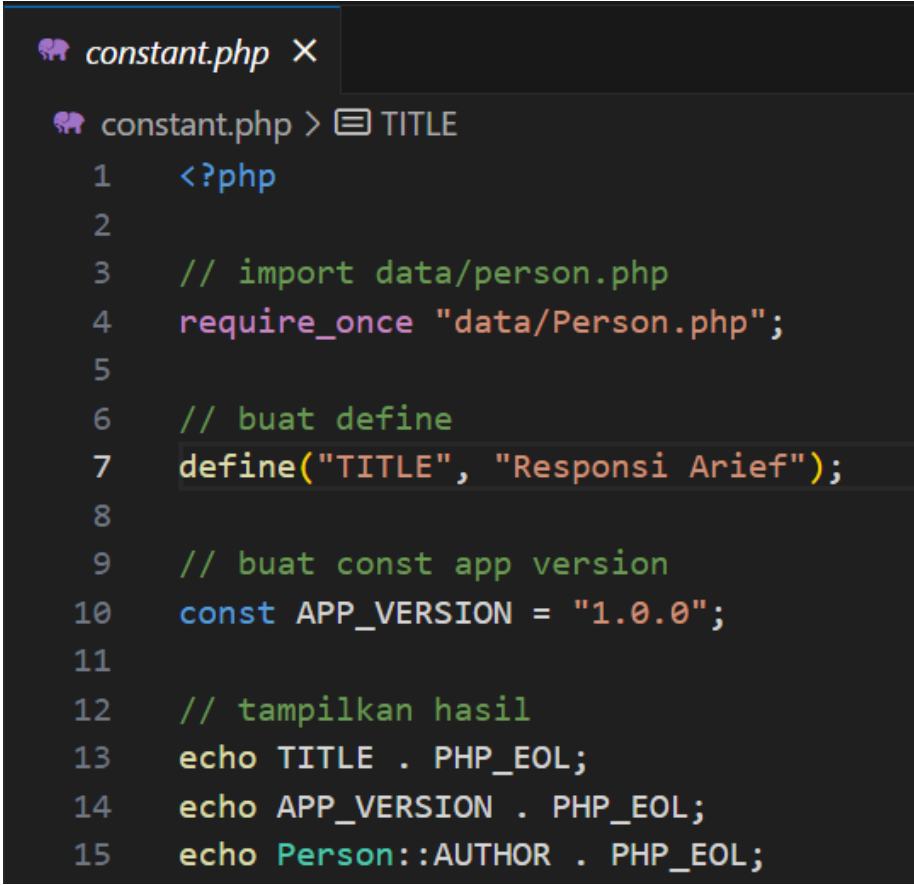
- Kelas Shape:
  - Kelas ini memiliki satu metode publik `getCorner()`, yang mengembalikan nilai -1. Metode ini memberikan implementasi default untuk mendapatkan jumlah sudut suatu bentuk (shape).
- Kelas Rectangle:



- Kelas ini mewarisi dari kelas Shape, menunjukkan konsep pewarisan dalam pemrograman berorientasi objek.
- Kelas ini memiliki metode `getCorner()` sendiri yang menggantikan implementasi dari kelas induk (Shape) dan mengembalikan nilai 4, mengindikasikan bahwa persegi panjang memiliki 4 sudut.
- Metode `getParentCorner()` digunakan untuk mendemonstrasikan pemanggilan metode dari kelas induk menggunakan kata kunci `parent::`. Metode ini mengembalikan nilai dari metode `getCorner` dari kelas Shape, yang seharusnya -1.

Dengan kata lain, kelas Rectangle memperluas atau mengkhususkan perilaku kelas Shape dengan mendefinisikan kembali (override) metode `getCorner`. Konsep pewarisan dan penggunaan `parent::` memungkinkan kelas turunan untuk mengakses atau memodifikasi perilaku yang sudah ada dari kelas induk.

## Constant.php



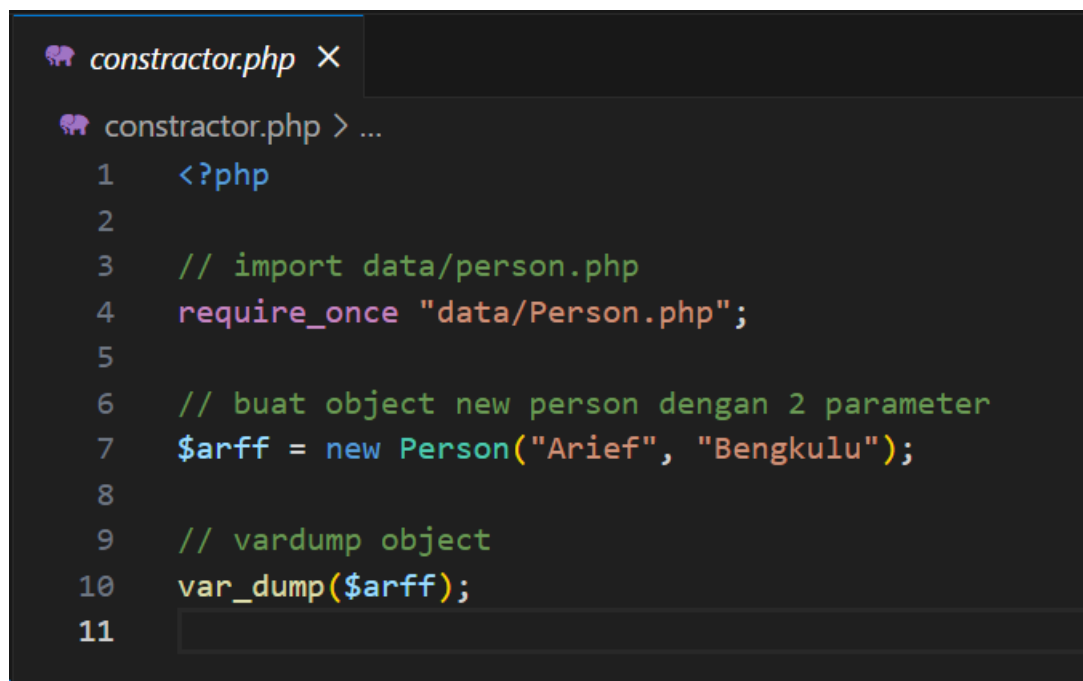
```

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

Penjelasan:

Kode PHP di atas pertama-tama mengimpor file eksternal Person.php menggunakan pernyataan `require_once`, memungkinkan akses ke kelas atau kode yang terdapat dalam file tersebut. Selanjutnya, kode mendefinisikan konstanta `TITLE` dengan nilai "Responsi Arief" menggunakan pernyataan `define`, dan konstanta `APP_VERSION` dengan nilai "1.0.0" menggunakan kata kunci `const`. Konstanta-konstanta tersebut bersifat global dan dapat diakses di seluruh skrip. Hasilnya kemudian ditampilkan menggunakan pernyataan `echo`, menampilkan nilai dari konstanta `TITLE`, `APP_VERSION`, dan konstanta kelas `Person` yang disebut `AUTHOR`. Konsep-konsep ini membantu dalam pengaturan nilai tetap yang digunakan secara konsisten di seluruh aplikasi PHP.

## 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  $arff = new Person("Arief", "Bengkulu");
8
9  // vardump object
10 var_dump($arff);
11
```

### Penjelasan:

Kode PHP di atas melakukan beberapa langkah. Pertama, dengan menggunakan pernyataan `require_once`, file eksternal Person.php diimpor untuk memberikan akses ke kelas Person. Selanjutnya, objek baru dengan nama `$arff` dibuat dari kelas Person dengan menggunakan konstruktor yang mengharapakan dua parameter, yaitu nama ("Arief") dan alamat ("Bengkulu"). Setelahnya, fungsi `var_dump` digunakan untuk menampilkan informasi rinci tentang struktur dan nilai dari objek `$arff`. Keseluruhan kodenya menunjukkan proses pembuatan objek dari kelas Person dengan nilai-nilai tertentu dan penggunaan fungsi `var_dump` untuk inspeksi struktur objek tersebut.

## Destuctor.php

```
desturctor.php X
desturctor.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Person.php";
5
6  // buat 2 object new person dengan parameter yang berbeda
7  $sarff = new Person("Arief", "Bengkulu");
8  $prass = new Person("Prasetyo", "Jogja");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
12
```

### Penjelasan:

Kode PHP di atas pertama-tama mengimpor file eksternal Person.php untuk menggunakan kelas Person yang terdapat di dalamnya. Selanjutnya, dua objek baru, \$sarff dan \$prass, dibuat dari kelas Person menggunakan konstruktor dengan parameter yang berbeda untuk masing-masing objek. Objek \$sarff memiliki nama "Arief" dan alamat "Bengkulu", sementara objek \$prass memiliki nama "Prasetyo" dan alamat "Jogja". Terakhir, sebuah pernyataan echo digunakan untuk menampilkan pesan "Program Selesai" di layar. Keseluruhan kodenya menciptakan dua objek Person dengan data yang berbeda dan menunjukkan pesan bahwa program telah selesai dijalankan.

## Function.php

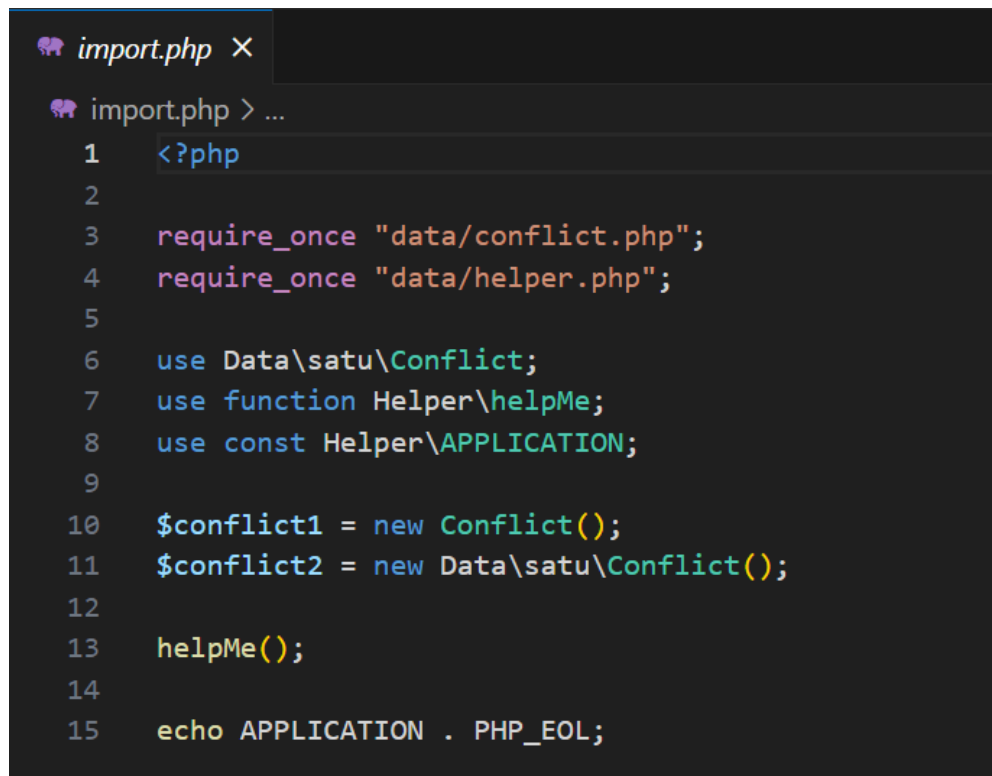
```
function.php X
function.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new Person("Arief", "Bengkulu");
8
9  // panggil function
10 $person1->sayHello("Arief");
```

### Penjelasan:

Kode PHP di atas pertama-tama mengimpor file eksternal person.php untuk mengakses kelas Person yang terdapat di dalamnya. Selanjutnya, sebuah objek baru dengan nama \$person1

dibuat dari kelas Person menggunakan konstruktor dengan dua parameter, yaitu nama "Arief" dan alamat "Bengkulu". Setelah objek dibuat, dipanggil metode sayHello dari objek \$person1 dengan menyertakan parameter "Arief". Metode ini mencetak pesan sapaan menggunakan nilai parameter dan properti nama dari objek, sehingga hasilnya adalah mencetak "Hi Arief, Namaku Arief" ke layar. Keseluruhan kodenya menggambarkan proses pembuatan objek dan pemanggilan metode dari kelas Person, mengilustrasikan konsep dasar dalam pemrograman berorientasi objek.

## Import.php

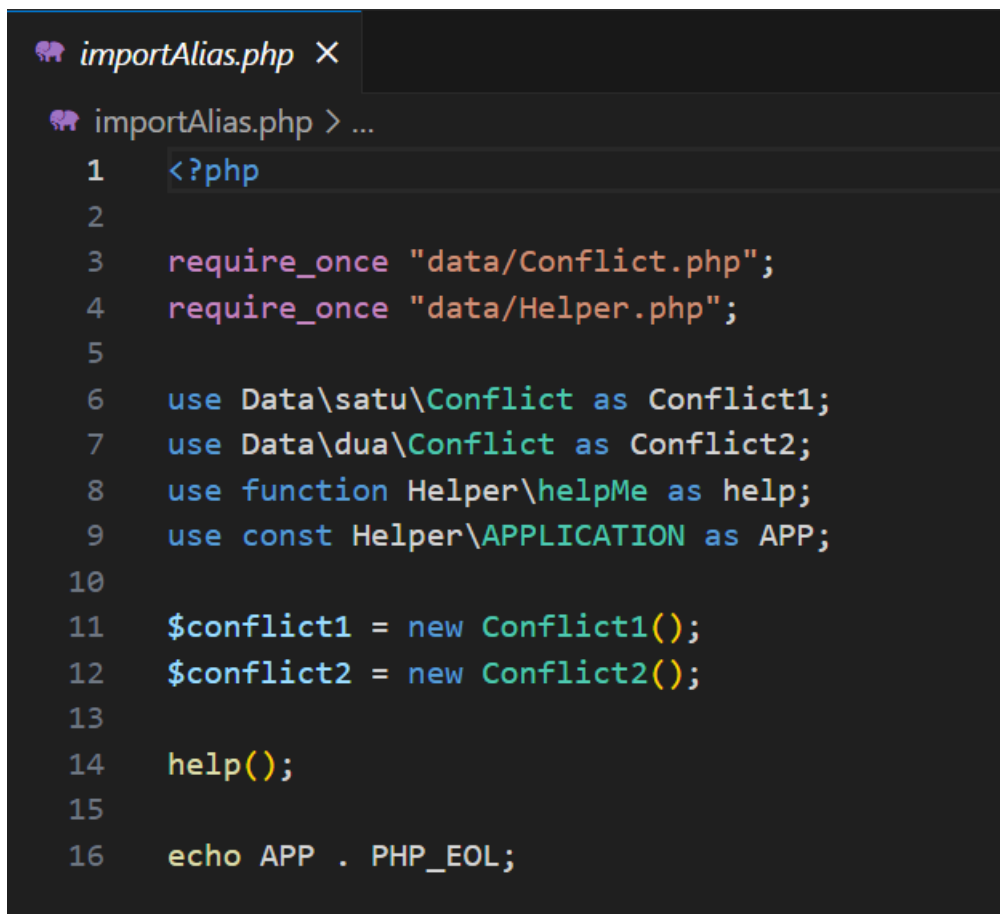


```
import.php X
import.php > ...
1  <?php
2
3  require_once "data/conflict.php";
4  require_once "data/helper.php";
5
6  use Data\satu\Conflict;
7  use function Helper\helpMe;
8  use const Helper\APPLICATION;
9
10 $conflict1 = new Conflict();
11 $conflict2 = new Data\satu\Conflict();
12
13 helpMe();
14
15 echo APPLICATION . PHP_EOL;
```

### Penjelasan:

Kode PHP di atas dimulai dengan mengimpor file eksternal person.php untuk mengakses kelas Person. Selanjutnya, objek baru dengan nama \$person1 dibuat dari kelas Person menggunakan konstruktor dengan dua parameter, yaitu "Arief" sebagai nama dan "Bengkulu" sebagai alamat. Setelah objek dibuat, fungsi sayHello dipanggil dari objek \$person1 dengan parameter "Arief". Fungsi ini mencetak pesan sapaan yang menggabungkan nilai parameter dan properti nama dari objek, sehingga hasilnya adalah mencetak "Hi Arief, Namaku Arief" ke layar. Keseluruhan kodenya mengilustrasikan konsep pembuatan objek, pemanggilan metode, dan penggunaan parameter dalam pemrograman berorientasi objek dengan PHP

## importAlias.php



```
importAlias.php X
importAlias.php > ...
1  <?php
2
3  require_once "data/Conflict.php";
4  require_once "data/Helper.php";
5
6  use Data\satu\Conflict as Conflict1;
7  use Data\dua\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;
```

### Penjelasan:

Kode PHP di atas mengimpor dua file eksternal, yaitu "conflict.php" dan "helper.php", menggunakan pernyataan `require_once`. Selanjutnya, dilakukan penggunaan namespace dengan kata kunci `use` untuk menghindari konflik nama antar-namespace. Objek `$conflict1` dibuat dari kelas `Conflict` yang terdapat dalam namespace global, sementara objek `$conflict2` dibuat dari kelas `Conflict` yang terdapat dalam namespace "Data\satu". Fungsi `helpMe()` dan konstanta `APPLICATION` diakses menggunakan kata kunci `use` untuk mempersingkat penulisan. Akhirnya, program mencetak hasil pemanggilan fungsi dan nilai konstanta ke layar. Keseluruhan kodenya menggambarkan penggunaan namespace untuk mengorganisir kode, mencegah konflik nama, dan memudahkan penggunaan fungsi dan konstanta dari namespace tertentu.

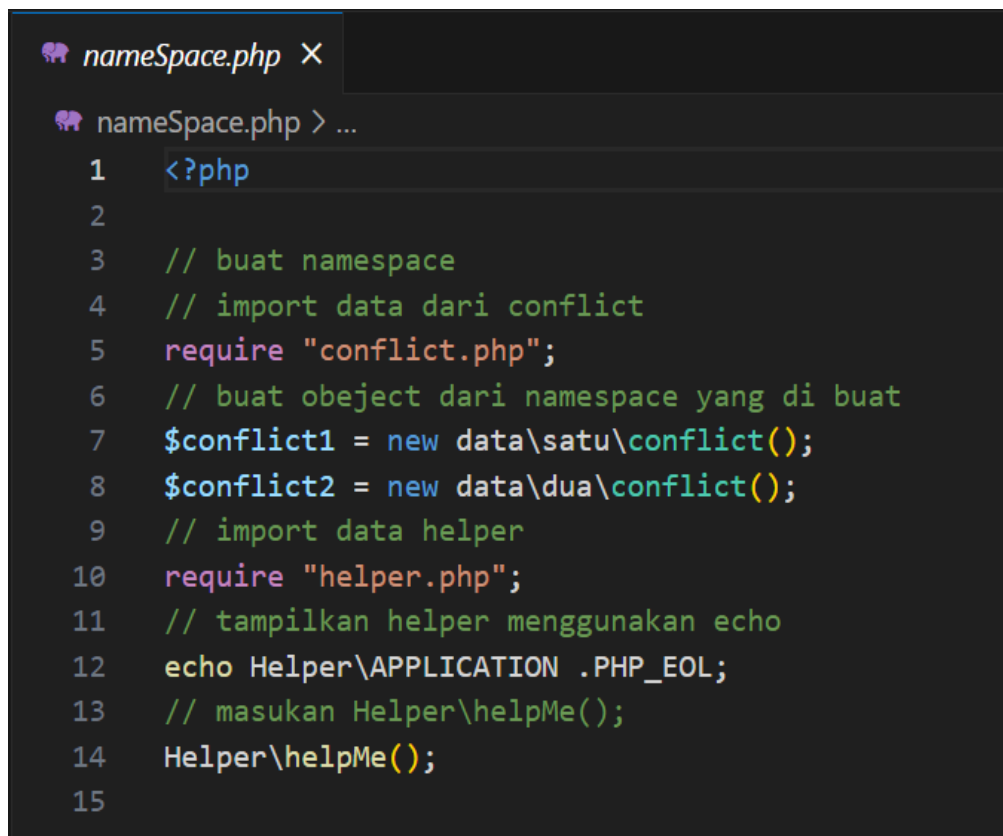
## inheritance.php

```
inheritance.php X
inheritance.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/Manager.php";
5  // buat object new manager dan tambahkan value nama kemudian panggil function
6  $manager1 = new Manager();
7  $manager1->nama = "Arief";
8  $manager1->sayHello("Arief Hariss");
9
10 // buat object new vicepresident dan tambahkan value nama kemudian panggil function
11 $vicePresident1 = new VicePresident();
12 $vicePresident1->nama = "Prass";
13 $vicePresident1->sayHello("Prasetyooo");
14
```

### Penjelasan:

Kode PHP di atas dimulai dengan mengimpor file eksternal Manager.php, yang berisi definisi kelas Manager dan VicePresident. Selanjutnya, objek baru \$manager1 dibuat dari kelas Manager, dan properti nama diisi dengan nilai "Arief". Fungsi sayHello kemudian dipanggil pada objek \$manager1 dengan parameter "Arief Hariss", yang mencetak pesan sapaan. Selanjutnya, objek \$vicePresident1 dibuat dari kelas VicePresident, dan properti nama diisi dengan nilai "Prass". Fungsi sayHello juga dipanggil pada objek \$vicePresident1 dengan parameter "Prasetyooo", yang mencetak pesan sapaan. Keseluruhan kodenya menggambarkan pembuatan objek dari kelas Manager dan VicePresident, pengisian nilai properti nama, dan pemanggilan fungsi sayHello untuk menampilkan pesan sapaan sesuai dengan nilai properti yang telah diatur.

## nameSpace.php



```
nameSpace.php X
nameSpace.php > ...
1  <?php
2
3  // buat namespace
4  // import data dari conflict
5  require "conflict.php";
6  // buat oobject dari namespace yang di buat
7  $conflict1 = new data\satu\conflict();
8  $conflict2 = new data\dua\conflict();
9  // import data helper
10 require "helper.php";
11 // tampilkan helper menggunakan echo
12 echo Helper\APPLICATION .PHP_EOL;
13 // masukan Helper\helpMe();
14 Helper\helpMe();
15
```

### Penjelasan:

Kode PHP di atas memulai dengan mendefinisikan dua namespace menggunakan kata kunci namespace, yaitu data\satu dan data\dua. Selanjutnya, file eksternal "conflict.php" diimpor untuk mengakses kelas conflict dari kedua namespace tersebut. Objek \$conflict1 dan \$conflict2 kemudian dibuat dari kelas conflict masing-masing dalam namespace data\satu dan data\dua. Selain itu, file eksternal "helper.php" diimpor untuk mengakses konstanta APPLICATION dan fungsi helpMe dari namespace Helper. Konstanta tersebut dan fungsi helpMe kemudian dicetak ke layar menggunakan pernyataan echo. Keseluruhan kodenya mencerminkan penggunaan namespace untuk mengorganisir kode, menghindari konflik nama, dan menunjukkan cara menggunakan elemen-elemen dari berbagai namespace dalam skrip PHP.

## object.php

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

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "person.php" dan membuat objek baru dari kelas Person dengan memberikan nilai "Arief" sebagai nama dan "Bengkulu" sebagai alamat melalui konstruktor. Selanjutnya, properti objek tersebut (nama, alamat, dan negara) dimanipulasi secara langsung dengan memberikan nilai baru. Pemanggilan pernyataan echo digunakan untuk menampilkan hasil manipulasi properti, mencetak nilai yang telah diubah untuk properti nama, alamat, dan negara ke layar. Keseluruhan kodenya menggambarkan proses pembuatan objek dan penggunaan properti dalam pemrograman berorientasi objek dengan PHP, di mana nilai properti dapat dimanipulasi setelah objek dibuat.



## parent.php

```
parent.php X
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;
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "Shape.php" yang berisi definisi kelas Shape dan Rectangle dalam namespace "Data". Objek \$shape dibuat dari kelas Shape, dan metode getCorner dari kelas tersebut dipanggil untuk menampilkan hasil (-1) ke layar. Selanjutnya, objek \$rectangle dibuat dari kelas Rectangle, yang merupakan turunan dari kelas Shape. Metode getCorner dari kelas Rectangle dipanggil, dan hasilnya (4) ditampilkan ke layar. Pemanggilan metode getParentCorner dari objek \$rectangle digunakan untuk memanggil metode yang diwarisi dari kelas Shape, dan hasilnya (-1) juga ditampilkan ke layar. Keseluruhan kodenya menggambarkan pembuatan objek, pemanggilan metode, dan pewarisan dalam pemrograman berorientasi objek dengan PHP, menunjukkan cara mengakses metode dari kelas dasar menggunakan pewarisan.

## polymorphism.php

```
polymorphism.php X
polymorphism.php > ...
1  <?php
2
3  require_once "data/programmer.php";
4
5  $company = new Company();
6  $company->programmer = new Programmer("Arief");
7  var_dump($company);
8
9  $company->programmer = new BackendProgrammer("Arief");
10 var_dump($company);
11
12 $company->programmer = new FrontendProgrammer("Arief");
13 var_dump($company);
14
15 sayHelloProgrammer(new Programmer("Arief Hariss"));
16 sayHelloProgrammer(new BackendProgrammer("Arief Hariss"));
17 sayHelloProgrammer(new FrontendProgrammer("Arief Hariss"));
18
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "programmer.php", yang berisi definisi kelas Programmer, BackendProgrammer, FrontendProgrammer, dan Company. Pertama, sebuah objek \$company dibuat dari kelas Company, dan properti \$programmer diisi dengan objek Programmer yang memiliki nama "Arief". Penggunaan var\_dump digunakan untuk menampilkan informasi detail tentang struktur objek dan propertinya. Selanjutnya, properti \$programmer diisi ulang dengan objek BackendProgrammer dan FrontendProgrammer secara berturut-turut, yang menunjukkan kemampuan polimorfisme dalam penggunaan objek dengan tipe yang berbeda-beda. Pemanggilan fungsi sayHelloProgrammer dilakukan untuk objek Programmer, BackendProgrammer, dan FrontendProgrammer dengan memberikan parameter nama "Arief Hariss". Keseluruhan kodenya mengilustrasikan konsep polimorfisme, pewarisan, dan penggunaan objek dengan tipe yang berbeda dalam pemrograman berorientasi objek dengan PHP.

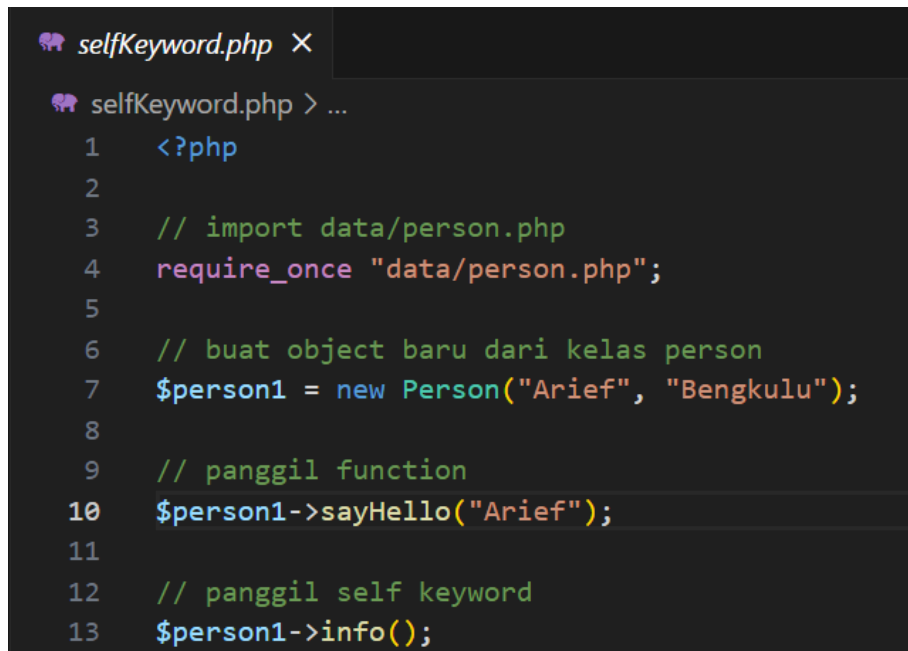
## property.php

```
property.php X
property.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("Arief", "Bengkulu");
8
9  // manipulasi properti nama person
10 $person1->nama = "Arief";
11 $person1->alamat = "Bengkulu";
12 $person1->negara = "Indonesia";
13 // menampilkan hasil
14 echo "nama = {$person1->nama}" . PHP_EOL;
15 echo "alamat = {$person1->alamat}" . PHP_EOL;
16 echo "negara = {$person1->negara}" . PHP_EOL;
17
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "person.php" dan membuat objek baru dari kelas Person dengan memberikan nilai "Arief" sebagai nama dan "Bengkulu" sebagai alamat melalui konstruktor. Selanjutnya, properti objek tersebut (nama, alamat, dan negara) dimanipulasi secara langsung dengan memberikan nilai baru. Pemanggilan pernyataan echo digunakan untuk menampilkan hasil manipulasi properti, mencetak nilai yang telah diubah untuk properti nama, alamat, dan negara ke layar. Keseluruhan kodenya menggambarkan proses pembuatan objek dan penggunaan properti dalam pemrograman berorientasi objek dengan PHP, di mana nilai properti dapat dimanipulasi setelah objek dibuat.

## selfKeyword.php



```
selfKeyword.php X
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("Arief", "Bengkulu");
8
9  // panggil function
10 $person1->sayHello("Arief");
11
12 // panggil self keyword
13 $person1->info();
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "person.php" dan membuat objek baru dari kelas Person dengan memberikan nilai "Arief" sebagai nama dan "Bengkulu" sebagai alamat melalui konstruktor. Selanjutnya, metode sayHello dipanggil pada objek \$person1 dengan parameter "Arief", yang mencetak pesan sapaan. Selain itu, metode info yang menggunakan kata kunci self:: dipanggil pada objek yang mencetak informasi tentang konstanta kelas AUTHOR. Keseluruhan kodenya mengilustrasikan pemanggilan metode pada objek, termasuk penggunaan kata kunci self:: untuk mengakses elemen kelas itu sendiri, dalam pemrograman berorientasi objek dengan PHP.

## thisKeyword.php

```
thisKeyword.php X
thisKeyword.php > ...
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object dari kelas person
7  $sarff = new Person("Arief", "Bengkulu");
8
9  // tambahkan value nama di object
10 $sarff->nama = "Arief";
11
12 // panggil function sayHelloNull dengan parameter
13 $sarff->sayHelloNull("Prasetyo");
14
15 // buat object dari kelas person
16 $prass = new Person("Prass", "Jogja");
17
18 // tambahkan value nama di object
19 $prass->nama = "Sumbul";
20
21 // panggil function sayHelloNull dengan parameter null
22 $prass->sayHelloNull(null);
23
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "person.php" dan membuat dua objek dari kelas Person, yaitu \$sarff dan \$prass, dengan memberikan nilai nama dan alamat melalui konstruktor. Properti nama pada objek \$sarff kemudian diubah nilainya menjadi "Arief". Selanjutnya, metode sayHelloNull dipanggil pada objek \$sarff dengan parameter "Prasetyo", yang mencetak pesan sapaan dengan mempertimbangkan nilai parameter yang tidak null. Objek \$prass juga mengalami perubahan nilai pada properti nama, kemudian metode sayHelloNull dipanggil dengan memberikan parameter null, sehingga mencetak pesan sapaan dengan memeriksa apakah nilai parameter adalah null atau tidak. Keseluruhan kodenya menggambarkan pembuatan objek, manipulasi properti, dan pemanggilan metode yang memperhitungkan nilai null dalam pemrograman berorientasi objek dengan PHP.

## visibility.php

```
visibility.php X
visibility.php > ...
1  <?php
2
3  require_once "data/Product.php";
4
5  $product = new Product("Apple", 20000);
6
7  // tampilkan product get name
8  echo $product->getName(). PHP_EOL;
9  // tampilkan product get price
10 echo $product->getPrice(). PHP_EOL;
11
12 $dummy = new ProductDummy("Dummy", 1000);
13 $dummy->info();
```

### Penjelasan:

Kode PHP di atas mengimpor file eksternal "Product.php", yang berisi definisi kelas Product dan ProductDummy. Objek \$product kemudian dibuat dari kelas Product dengan memberikan nilai "Apple" sebagai nama dan 20000 sebagai harga melalui konstruktor. Pemanggilan metode getName dan getPrice pada objek \$product digunakan untuk menampilkan nama dan harga produk ke layar. Selanjutnya, objek \$dummy dibuat dari kelas turunan ProductDummy dengan memberikan nilai "Dummy" sebagai nama dan 1000 sebagai harga melalui konstruktor. Pemanggilan metode info pada objek \$dummy digunakan untuk menampilkan informasi produk, yang mencetak nama dan harga ke layar. Keseluruhan kodenya menggambarkan pembuatan objek, penggunaan metode, dan pewarisan dalam pemrograman berorientasi objek dengan PHP, menunjukkan cara mengakses properti dan metode dari kelas turunan.