

RESPONSI PRAKTIKUM PEMPROGRAMAN BERORIENTASI OBJEK

Nama : Refi Oryza
NPM : G1F022068
Soal :

- Silahkan lakukan git clone repositori dari <https://github.com/alzahfariski/bahan-ajar-pbo> (silahkan liat di youtube caranya)
- lengkapi code php yang belum lengkap sehingga setiap file dapat di run dan tidak memunculkan error
- upload atau lakukan git push ke akun git kalian masing-masing
- salin url lalu kumpulkan dengan berikan penjelasan mengenai pemahaman kalian secara descriptive (contoh penjelasan mengenai file object.php menggunakan code apa saja dan berfungsi untuk apa) penjelasan kalian akan mempengaruhi penilaian

Pembahasan :

a. conflict

```
1  <?php
2  // buat namespace data\satu
3  // dengan class conflict
4  // class sample
5  // class dummy
6  // buat namespace data\dua
7  // dengan class conflict
8  namespace data\satu;
9
10 class Conflict {
11     private $message;
12
13     public function __construct($message) {
14         $this->message = $message;
15     }
16
17     public function getMessage() {
18         return $this->message;
19     }
20 }
21
22 namespace data\dua;
23
24 class Conflict {
25     private $message;
26
27     public function __construct($message) {
28         $this->message = $message;
29     }
30
31     public function getMessage() {
32         return $this->message;
33     }
34 }
35
36 namespace H1student;
37
38 class Conflict {
39     private $message;
40
41     public function __construct($message) {
42         $this->message = $message;
43     }
44
45     public function getMessage() {
46         return $this->message;
47     }
48 }
49 >>
```

Gambar 1.1 source code conflict

Pembahasan :

Pada gambar 1.1 merupakan Source Code dari data [conflict](#). Dalam namespace `data\satu`, mendefinisikan kelas `Conflict` yang memiliki property `message`, `konstruktor __construct` untuk menginisialisasi nilai properti saat objek dibuat, dan metode `getMessage` untuk mengembalikan nilai properti `message`. Dalam namespace `data\dua`, mendefinisikan kelas `Conflict` dengan struktur yang sama seperti yang ada dalam namespace `data\satu`. Ini menunjukkan penggunaan namespace yang berbeda untuk kelas yang memiliki nama yang sama. Dalam namespace `Hlstudent`, mendefinisikan kelas `Conflict` dengan struktur yang sama. Ini menunjukkan bahwa namespace dapat digunakan untuk mencegah konflik antara kelas dengan nama yang sama yang mungkin berasal dari bagian-bagian yang berbeda dari kode. Penggunaan namespace di sini membantu mengorganisir kode dan menghindari konflik nama kelas.

b. Person

```
1 <?php
2
3 // membuat kelas person
4 class Person{
5     // membuat properti
6     var string $nama;
7
8     // gunakan nullable properti
9     var ?string $alamat = null;
10
11     // gunakan default value untuk properti
12     var string $negara = "Turkey";
13
14     // buat function sayHello
15     function sayHello(string $nama){
16         echo "Hello $nama" . PHP_EOL;
17     }
18
19     // buat function sayHello nullable dengan percabangan
20     function sayHelloNull(?string $nama)
21     {
22         if (is_null($nama)) {
23             echo "I hope u okey" . PHP_EOL;
24         } else {
25             echo "Annyeong $nama, how are you there?" . PHP_EOL;
26         }
27     }
28
29     // buat const author
30     const AUTHOR = "Kelas PBO B 23";
31
32     // buat function info untuk self keyword
33
34     function info()
35     {
36         echo "Author : " . self::AUTHOR . PHP_EOL;
37     }
38
39     // buat function constructor
40     function __construct(string $nama, ?string $alamat)
41     {
42         $this->nama = $nama;
43         $this->alamat = $alamat;
44     }
45
46     // buat function destructor
47     function __destruct()
48     {
49         echo "Object person $this->nama is destroyed" . PHP_EOL;
50     }
51 }
```

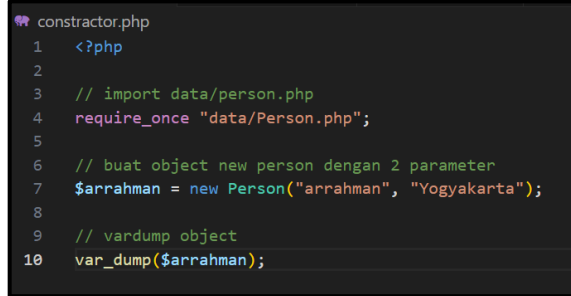
Gambar 1.2 Source Code Person

Pembahasan :

Pada gambar 1.2 merupakan Source code person. awal dari definisi kelas Person. Semua kode yang terkait dengan kelas Person akan ditempatkan di dalam blok kurung kurawal ini. pada data ini pendeklarasikan beberapa properti kelas (`$nama`, `$alamat`, dan `$negara`). Properti `$nama` dan `$negara` memiliki tipe data yang ditentukan (`string`), sedangkan `$alamat` memiliki tipe data yang dapat null dan diberikan nilai default null. Pada data ini di definisikan sebuah konstanta kelas dengan nama `'AUTHOR'` yang memiliki nilai "Kelas PBO B 23". Fungsi ini mencetak informasi menggunakan konstanta kelas `'AUTHOR'` dengan menggunakan kata kunci `self` untuk merujuk pada kelas itu sendiri. Konstruktor kelas yang dijalankan ketika objek dibuat. Menerima

parameter \$nama dan \$alamat dan menginisialisasi properti kelas. Destruktor kelas yang dijalankan ketika objek dihancurkan.

c. Constructor



```
1  <?php
2
3  // import data/person.php
4  require_once "data/Person.php";
5
6  // buat object new person dengan 2 parameter
7  $sarrahan = new Person("arrahan", "Yogyakarta");
8
9  // vardump object
10 var_dump($sarrahan);
```

Gambar 1.3 Source Code Constructor

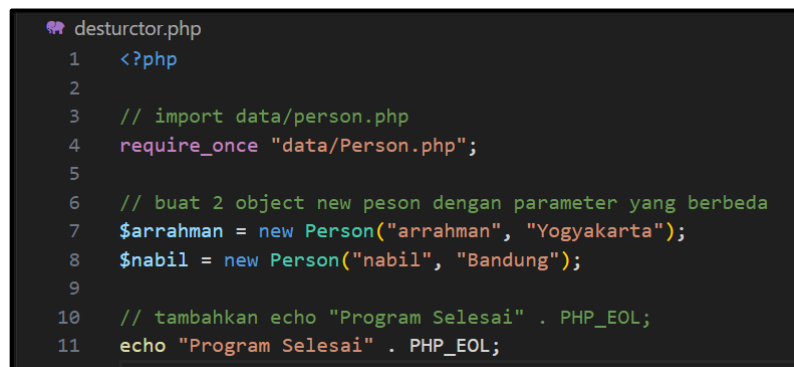
Pembahasan :

Source code tersebut adalah contoh penggunaan kelas Person yang diimpor dari file data/Person.php. `require_once "data/Person.php";` Baris ini digunakan untuk mengimpor file Person.php yang berada di direktori data. Fungsi `require_once` akan memastikan bahwa file hanya diimpor sekali, dan jika sudah diimpor sebelumnya, tidak akan diimpor lagi.

`$sarrahan = new Person("arrahan", "Yogyakarta");` Baris ini membuat objek baru dengan nama \$sarrahan dari kelas Person. Konstruktor kelas Person membutuhkan dua parameter, yaitu nama dan alamat. Dalam hal ini, memberikan nilai "arrahan" sebagai nama dan "Yogyakarta" sebagai alamat.

`var_dump($sarrahan);` Baris ini menggunakan fungsi `var_dump` untuk mencetak informasi lengkap tentang objek \$sarrahan. Fungsi `var_dump` digunakan untuk debugging dan menampilkan informasi tentang tipe dan nilai dari variabel atau objek. Dalam hal ini, dapat melihat struktur dan nilai properti dari objek \$sarrahan.

d. Destructor



```
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  $sarrahan = new Person("arrahan", "Yogyakarta");
8  $nabil = new Person("nabil", "Bandung");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
```

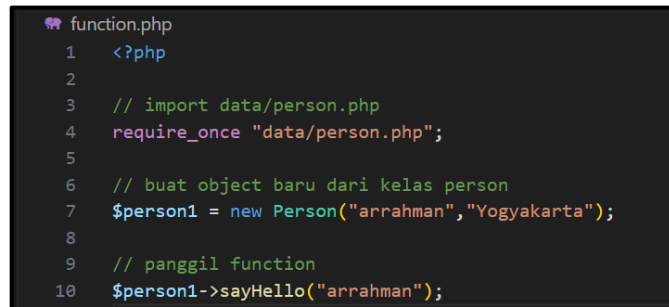
Gambar 1.4 Source code Destructor

Pembahasan :

`require_once "data/Person.php";` Baris ini mengimpor file Person.php yang berada dalam direktori data. `require_once` digunakan untuk memastikan bahwa file hanya diimpor satu kali, bahkan jika dipanggil beberapa kali. `new Person("arrahan", "Yogyakarta");` Baris ini membuat objek baru dari class Person. Objek ini disimpan dalam variabel \$sarrahan. Constructor Person dipanggil dengan memberikan dua parameter, yaitu "arrahan" sebagai nama dan "Yogyakarta" sebagai alamat. `new Person("nabil", "Bandung");` Baris ini mirip dengan baris sebelumnya, tetapi objek

disimpan dalam variabel \$nabil dengan parameter yang berbeda, yaitu "nabil" sebagai nama dan "Bandung" sebagai alamat. `echo "Program Selesai" . PHP_EOL;` Baris ini menampilkan pesan "Program Selesai" ke layar dengan menggunakan fungsi `echo`. `PHP_EOL` digunakan untuk memasukkan newline (baris baru) sesuai dengan konvensi sistem operasi. Untuk keseluruhan, source code tersebut mengimport class `Person` dari file `Person.php`, membuat dua objek dari class tersebut (masing-masing untuk "arrahan" dan "nabil"), dan akhirnya menampilkan pesan "Program Selesai". Ini adalah contoh sederhana penggunaan objek dan class dalam PHP.

e. Function



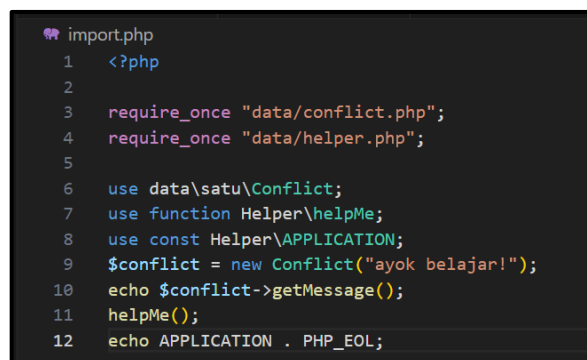
```
function.php
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new Person("arrahan","Yogyakarta");
8
9  // panggil function
10 $person1->sayHello("arrahan");
```

Gambar 1.5 Source code Function

Pembahasan :

`require_once "data/person.php";` Ini adalah pernyataan untuk mengimpor (include) file `person.php`. Dengan menggunakan `require_once`, file ini hanya akan diimpor sekali, bahkan jika pernyataan ini dipanggil berkali-kali. File ini diasumsikan berisi definisi kelas `Person`. `$person1 = new Person("arrahan","Yogyakarta");` Baris ini membuat objek baru dari kelas `Person`. Objek tersebut disimpan dalam variabel `$person1`. Saat membuat objek, konstruktor kelas `Person` dipanggil dengan memberikan dua parameter, yaitu nama "arrahan" dan lokasi "Yogyakarta". `$person1->sayHello("arrahan");` Baris ini memanggil metode `sayHello` dari objek `$person1`. Metode ini menerima satu parameter, yaitu nama, dan kemudian mencetak pesan sapaan menggunakan fungsi `echo`. Di dalam file `person.php`, mendefinisikan kelas `Person` dengan dua properti (`$name` dan `$location`), sebuah konstruktor untuk menginisialisasi objek, dan metode `sayHello` yang mencetak pesan sapaan. Objek yang dibuat dalam file utama (main.php) kemudian menggunakan kelas ini untuk menampilkan pesan sapaan.

f. Import



```
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  $conflict = new Conflict("ayok belajar!");
10 echo $conflict->getMessage();
11 helpMe();
12 echo APPLICATION . PHP_EOL;
```

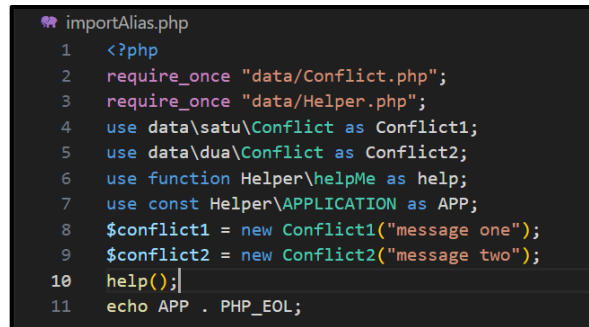
Gambar 1.6 Source code Import

Pembahasan :

Pertama-tama, kode ini mengimpor kelas Conflict dari file conflict.php dan seluruh fungsi dan konstanta dari file helper.php. Setelah itu, kode menggunakan `use` untuk mengimpor kelas, fungsi, dan konstanta dari namespace yang sesuai. Dengan menggunakan `use data\satu\Conflict`, memberitahu PHP bahwa akan menggunakan kelas Conflict dari namespace data\satu. Dengan demikian, dapat membuat objek dari kelas tersebut. Dengan `use function Helper\helpMe`, memberitahu PHP bahwa akan menggunakan fungsi helpMe dari namespace Helper. Dengan `use const Helper\APPLICATION`, memberitahu PHP bahwa akan menggunakan konstanta APPLICATION dari namespace Helper. Membuat objek \$conflict dari kelas Conflict dengan mengirimkan pesan "ayok belajar!" ke konstruktor. Kemudian, kode memanggil metode getMessage() dari objek tersebut dan menampilkan pesan. Memanggil fungsi `helpMe()` yang telah diimpor. Menampilkan nilai konstanta APPLICATION yang telah diimpor, diikuti dengan newline (PHP_EOL).

UNTuk keseluruhan, kode ini mengimpor kelas, fungsi, dan konstanta dari beberapa namespace, membuat objek, dan menjalankan beberapa fungsi dan metode tersebut.

g. ImportAlias

A screenshot of a code editor showing the source code for a file named 'importAlias.php'. The code is written in PHP and includes comments in Indonesian. It demonstrates the use of 'require_once' to include 'data/Conflict.php' and 'data/Helper.php'. It then uses 'use' statements to import classes 'Conflict' from namespaces 'data\satu' and 'data\dua' as 'Conflict1' and 'Conflict2' respectively. It also imports a function 'helpMe' from the 'Helper' namespace as 'help' and a constant 'APPLICATION' as 'APP'. The code creates two objects, '\$conflict1' and '\$conflict2', using the imported classes. It calls the 'help()' function and echoes the 'APP' constant followed by a newline character (PHP_EOL).

```
importAlias.php
1  <?php
2  require_once "data/Conflict.php";
3  require_once "data/Helper.php";
4  use data\satu\Conflict as Conflict1;
5  use data\dua\Conflict as Conflict2;
6  use function Helper\helpMe as help;
7  use const Helper\APPLICATION as APP;
8  $conflict1 = new Conflict1("message one");
9  $conflict2 = new Conflict2("message two");
10 help();
11 echo APP . PHP_EOL;
```

Gambar 1.7 Source Code ImportAlias

Pemabahasan :

`require_once "data/Conflict.php";` Mengimpor file Conflict.php yang berisi definisi namespace data\satu dan data\dua. `require_once "data/Helper.php";` Mengimpor file Helper.php yang berisi definisi namespace Helper. `use data\satu\Conflict as Conflict1;` Menggunakan alias Conflict1 untuk menyederhanakan pemanggilan kelas Conflict dari namespace data\satu. Dengan menggunakan alias, dapat merujuk ke kelas tersebut dengan nama yang lebih singkat. `use data\dua\Conflict as Conflict2;` Menggunakan alias Conflict2 untuk menyederhanakan pemanggilan kelas Conflict dari namespace data\dua. `use function Helper\helpMe as help;` Menggunakan alias help untuk menyederhanakan pemanggilan fungsi helpMe dari namespace Helper. `use const Helper\APPLICATION as APP;` Menggunakan alias APP untuk menyederhanakan pemanggilan konstanta APPLICATION dari namespace Helper. `$conflict1 = new Conflict1("message one");` Membuat objek dari kelas `Conflict` dalam namespace data\satu dengan menggunakan alias Conflict1. `$conflict2 = new Conflict2("message two");` Membuat objek dari kelas Conflict dalam namespace data\dua dengan menggunakan alias Conflict2. `help();` Memanggil fungsi helpMe dari namespace Helper menggunakan alias help. `echo APP . PHP_EOL;` Menampilkan nilai konstanta APPLICATION dari namespace Helper menggunakan alias APP, disertai dengan ganti baris. Untuk keseluruhan, source code ini menunjukkan cara menggunakan namespace, alias, dan elemen dari namespace (kelas, fungsi, dan konstanta) dalam PHP untuk

menghindari konflik nama (name conflict) dan membuat kode lebih mudah dimengerti. Alih-alih menggunakan nama penuh namespace, dapat menggunakan alias untuk membuat kode lebih singkat dan jelas.

h. Inheritance

```
inheritance.php
1  <?php
2  // import data/person.php
3  require_once "data/Manager.php";
4  // buat object new manager dan tambahkan value nama kemudian panggil function
5  $manager = new Manager();
6  $manager->nama = "Arrahman";
7  $manager->sayHello("dude");
8  // buat object new vicepresident dan tambahkan value nama kemudian panggil function
9  $vp = new VicePresident();
10 $vp->nama = "Nabil";
11 $vp->sayHello("bro");
```

Gambar 1.8 Source Code Inheritance

Pembahasan :

`require_once "data/Manager.php";` Baris ini mengimpor file Manager.php yang terletak dalam folder data. `require_once` digunakan untuk memastikan bahwa file hanya diimpor sekali agar tidak terjadi duplikasi. `$manager = new Manager();` Membuat objek baru dari kelas Manager dan menyimpannya dalam variabel \$manager. Ini menciptakan instance dari kelas Manager yang dapat digunakan untuk mengakses properti dan metode yang didefinisikan dalam kelas tersebut. `$manager->nama = "Arrahman";` Memberikan nilai "Arrahman" ke properti nama dari objek `\$manager`. Properti ini mungkin merupakan atribut dari kelas Manager. `$manager->sayHello("dude");` Memanggil metode `sayHello` dari objek \$manager dengan memberikan parameter "dude". Fungsi `sayHello` ini mungkin merupakan metode yang didefinisikan dalam kelas Manager dan digunakan untuk menampilkan pesan sapaan. `$vp = new VicePresident();` Membuat objek baru dari kelas VicePresident dan menyimpannya dalam variabel \$vp. Ini menciptakan instance dari kelas VicePresident yang dapat digunakan untuk mengakses properti dan metode yang didefinisikan dalam kelas tersebut. `$vp->nama = "Nabil";` Memberikan nilai "Nabil" ke properti nama dari objek \$vp. Properti ini mungkin merupakan atribut dari kelas VicePresident. `$vp->sayHello("bro");` Memanggil metode `sayHello` dari objek \$vp dengan memberikan parameter "bro". Fungsi `sayHello` ini mungkin merupakan metode yang didefinisikan dalam kelas `VicePresident` dan digunakan untuk menampilkan pesan sapaan.

Kesimpulannya, source code tersebut membuat dua objek (\$manager dan \$vp), mengatur nilai properti nama untuk keduanya, dan memanggil fungsi `sayHello` pada masing-masing objek dengan parameter yang berbeda.

i. NameSpace

```
nameSpace.php
1  <?php
2  // buat namespace
3  // import data dari conflict
4  // buat oobject dari namespace yang di buat
5  // import data helper
6  // tampilkan helper menggunakan echo
7  // masukan Helper\helpMe();
8  require_once "data/conflict.php";
9  require_once "data/helper.php";
10 use HIstudent\Conflict;
11 use function Helper\helpMe as help;
12 echo help();
13 ?>
```

Gambar 1.9 Source code NameSpace

Pembahasan :

Pada bagian ini, sebuah namespace dengan nama HIstudent dibuat. Namespace ini akan digunakan untuk mengelompokkan kelas-kelas dan fungsi-fungsi yang ada dalam konteks tertentu. file conflict.php yang berisi definisi namespace dan kelas-kelas dalam namespace HIstudent\Conflict diimpor. Hal ini berarti semua kelas dan fungsi dalam file tersebut dapat digunakan dalam konteks file saat ini. kelas Data dari namespace HIstudent\Conflict diinstansiasi menjadi objek \$conflictData. Dengan menggunakan `use`, memberitahu PHP bahwa akan menggunakan kelas Data dalam namespace HIstudent\Conflict. Baris mengimpor file helper.php yang berisi namespace Helper dengan kelas-kelas dan fungsi-fungsi yang ada di dalamnya. Pada source code ini menggunakan use function untuk mengimpor fungsi helpMe dari namespace Helper dan memberikan alias help untuk fungsi tersebut. Setelah itu, menampilkan hasil dari pemanggilan fungsi `help()` menggunakan perintah `echo`.

j. Object

```
object.php
1  <?php
2  // import data/person.php
3  require_once "data/person.php";
4  // buat object baru dari kelas person
5  $person = new Person("Arrahman","Yogyakarta");
6  // manipulasi properti nama, alamat, negara
7  $person->nama = "Arrahman";
8  $person->alamat = "Yogyakarta";
9  $person->negara = "Turkey";
10 // menampilkan hasil
11 echo "nama = {$person->nama}" . PHP_EOL;
12 echo "<br>alamat = {$person->alamat}" . PHP_EOL;
13 echo "<br>negara = {$person->negara}" . PHP_EOL;
```

Gambar 1.10 SourceCode Object

Pembahasan :

Pada Source code menggunakan pernyataan `require_once` untuk mengimpor file person.php ke dalam skrip utama. File ini diharapkan berisi definisi kelas Person yang akan digunakan nanti dalam source code. sebuah objek baru dari kelas Person, Objek ini diberi nama \$person dan diinisialisasi dengan dua parameter: "Arrahman" sebagai nama dan "Yogyakarta" sebagai alamat. Ini menunjukkan bahwa kelas Person mungkin

memiliki konstruktor yang menerima dua parameter. mengubah nilai properti dari objek \$person. Properti nama diubah menjadi "Arrahman", properti alamat diubah menjadi "Yogyakarta", dan properti negara diubah menjadi "Turkey". menampilkan hasil dari manipulasi objek. Pernyataan echo digunakan untuk mencetak nilai properti nama, alamat, dan negara dari objek \$person. `PHP_EOL` digunakan untuk menciptakan baris baru dalam output. Dalam konteks web, `
` juga digunakan untuk memberikan baris baru di HTML.

Secara keseluruhan, source code tersebut menggambarkan penggunaan kelas Person untuk membuat objek yang mewakili seseorang dan mengubah serta menampilkan beberapa propertinya.

k. PolyMorphism



```
polymorphism.php
1  <?php
2
3  require_once "data/Programmer.php";
4
5  $company = new Company();
6  $company->programmer = new Programmer("Nabil Arrahman");
7  $company->programmer = new BackendProgrammer("Refi Oryza");
8  $company->programmer = new FrontendProgrammer("Nabila Wijaya");
9  sayHelloProgrammer(new Programmer("Nabil Arrahman<br>"));
10 sayHelloProgrammer(new BackendProgrammer("Refi Oryza<br>"));
11 sayHelloProgrammer(new FrontendProgrammer("Nabila Wijaya<br>"));
```

Gambar 1.11 Source Code PolyMorphism

Pembahasan :

`require_once "data/Programmer.php";` Baris ini mengimpor file `Programmer.php` yang berada di dalam direktori `data`. File ini kemungkinan besar berisi definisi class untuk Programmer, BackendProgrammer, dan FrontendProgrammer.

`$company = new Company();` Membuat objek `\$company` dari class `Company`. Namun, perlu diingat bahwa dalam source code yang diberikan, tidak ada definisi untuk class `Company`. Mungkin file `Programmer.php` juga berisi definisi class `Company`, atau seharusnya file tersebut diimpor sebelumnya.

`$company->programmer = new Programmer("Nabil Arrahman");` Membuat objek baru dari class `Programmer` dengan nama "Nabil Arrahman" dan menyimpannya dalam property `programmer` dari objek `\$company`.

`$company->programmer = new BackendProgrammer("Refi Oryza");` Mengganti nilai property `programmer` dari objek `\$company` dengan objek baru dari class `BackendProgrammer` dengan nama "Refi Oryza".

`$company->programmer = new FrontendProgrammer("Nabila Wijaya");` Mengganti lagi nilai property `programmer` dari objek `\$company` dengan objek baru dari class `FrontendProgrammer` dengan nama "Nabila Wijaya".

`sayHelloProgrammer(new Programmer("Nabil Arrahman
"));` Memanggil fungsi `sayHelloProgrammer` dengan parameter berupa objek baru dari class `Programmer` dengan nama "Nabil Arrahman". Fungsi ini kemungkinan besar menampilkan pesan sapaan untuk seorang programmer.

`sayHelloProgrammer(new BackendProgrammer("Refi Oryza
"));` Memanggil lagi fungsi `sayHelloProgrammer`, kali ini dengan parameter berupa objek baru dari class `BackendProgrammer` dengan nama "Refi Oryza".

`sayHelloProgrammer(new FrontendProgrammer("Nabila Wijaya
"))`; Memanggil lagi fungsi `sayHelloProgrammer`, kali ini dengan parameter berupa objek baru dari class `FrontendProgrammer` dengan nama "Nabila Wijaya".

l. Properti

```
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("arrahan", "Yogyakarta");
8
9  // manipulasi properti nama person
10 $person1->nama = "arrahan";
11
12 // menampilkan hasil
13 echo "nama = {$person1->nama}" . PHP_EOL;
14 echo "<br>alamat = {$person1->alamat}" . PHP_EOL;
15 echo "<br>negara = {$person1->negara}" . PHP_EOL;
```

Gambar 1.12 Source Code Properti

Pembahasan :

mengimport file `person.php` ke dalam script PHP. File ini kemungkinan berisi definisi kelas `Person` dan mungkin beberapa logika terkait. membuat objek baru dari kelas `Person` dengan nama `$person1`. Constructor kelas `Person` dipanggil dengan dua parameter: "arrahan" untuk properti nama dan "Yogyakarta" untuk properti alamat. mengganti nilai properti nama dari objek `$person1` dengan nilai baru yaitu "arrahan". Properti ini mungkin bisa diakses karena memiliki tingkat visibilitas yang memungkinkan akses dari luar kelas.

mencetak hasil manipulasi ke layar. Dengan menggunakan `echo`, menampilkan nilai properti nama, alamat, dan negara dari objek `$person1`. Perlu diperhatikan bahwa properti negara tidak diinisialisasi sebelumnya, jadi nilai default (mungkin `null` atau nilai lainnya) akan ditampilkan.

m. SelfKeyword

```
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("arrahan", "Yogyakarta");
8
9  // panggil function
10 $person1->sayHello("arrahan");
11
12 // panggil self keyword
13 $person1->info();
```

Gambar 1.13 Source Code selfKeyword

Pembahasan :

tag PHP yang menandakan awal dari kode PHP. kedua menggunakan `require_once` untuk mengimpor (memasukkan) file `person.php` yang berada dalam folder `data`. Fungsi `require_once` digunakan untuk memastikan bahwa file hanya diimpor sekali agar tidak ada duplikat. membuat objek baru dari kelas `Person`. Ini dilakukan dengan

menggunakan kata kunci `new` diikuti oleh nama kelas (`Person`) dan disertai dengan parameter `"arrahman"` dan `"Yogyakarta"`. Dengan ini, objek `$person1` dibuat berdasarkan definisi kelas `Person` yang mungkin terdapat di dalam file `person.php`.

memanggil metode `sayHello()` dari objek `$person1`. Metode ini menerima parameter `"arrahman"` dan kemungkinan melakukan operasi tertentu. Namun, detail implementasi metode ini tidak terlihat dalam potongan kode yang diberikan.

memanggil metode `info()` dari objek `$person1`. Dalam metode `info()`, mungkin terdapat penggunaan kata kunci `self`, yang biasanya digunakan untuk merujuk ke kelas saat ini. Oleh karena itu, pemanggilan `info()` mungkin melakukan operasi yang melibatkan properti atau metode statis di dalam kelas `Person`.

Kesimpulannya pada source code ini menciptakan objek dari kelas `Person`, memanggil beberapa metodenya, dan mungkin memberikan output atau efek tertentu. Detail implementasi kelas `Person` dan metodenya harus dicari dalam file `person.php` untuk pemahaman yang lebih lengkap.

n. ThisKeyword

```
thisKeyword.php
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object dari kelas person
7  $arrahman = new Person("arrahman", "Yogyakarta");
8
9  // tambahkan value nama di object
10 $arrahman->nama = "arrahman";
11
12 // panggil function sayHelloNull dengan parameter
13 $arrahman->sayHelloNull("Bul");
14
15 // buat object dari kelas person
16 $nabil = new Person("nabil", "Bandung");
17
18 // tambahkan value nama di object
19 $nabil->nama = "nabil";
20
21 // panggil function sayHelloNull dengan parameter null
22 $nabil->sayHelloNull(null);
```

Gambar 1.14 Source Code ThisKeyword

Pembahasan :

file `person.php` diimpor menggunakan `require_once`. Ini memastikan bahwa definisi kelas `Person` diambil dari file tersebut. Objek `$arrahman` dibuat dari kelas `Person` dengan dua parameter konstruktor: `"arrahman"` sebagai nama dan `"Yogyakarta"` sebagai kota. Ini menginisialisasi objek dengan nilai-nilai awal. Nama objek `$arrahman` diperbarui menjadi `"arrahman"` menggunakan properti `nama`. Metode `sayHelloNull` pada objek `$arrahman` dipanggil dengan parameter `"Bul"`. Metode ini menerima satu parameter dan mencetak pesan `"Hello, Bul!"` jika parameter tidak `null`. Kesimpulannya source code ini menggambarkan cara membuat objek dari kelas `Person`, mengatur nilai propertinya, dan memanggil metode pada objek tersebut dengan memberikan parameter yang berbeda.

o. Visibility

```
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  // tampilkan product get price
9
10 $dummy = new ProductDummy("Dummy", 1000);
11 $dummy->info();
```

Gambar 1.15 SourceCode Visibility

Pembahasan :

`require_once "data/Product.php";` Baris ini mengimpor (meng-include) file `Product.php` yang berisi definisi kelas `Product` dan `ProductDummy`.

`$product = new Product("Apple", 20000);` Baris ini membuat objek baru dari kelas `Product` dengan memberikan nilai "Apple" sebagai nama produk dan 20000 sebagai harga produk.

`echo "Product Name: " . $product->getName() . PHP_EOL;` Baris ini menampilkan nama produk dengan memanggil method `getName()` dari objek `$product`. `PHP_EOL` digunakan untuk memberikan newline agar hasil tampilan lebih rapi.

`echo "Product Price: " . $product->getPrice() . PHP_EOL;` Baris ini menampilkan harga produk dengan memanggil method `getPrice()` dari objek `$product`.

`$dummy = new ProductDummy("Dummy", 1000);` Baris ini membuat objek baru dari kelas `ProductDummy` dengan memberikan nilai "Dummy" sebagai nama produk dan 1000 sebagai harga produk.

`$dummy->info();` Baris ini memanggil method `info()` dari objek `$dummy`. Method `info()` mungkin berisi logika tertentu yang menampilkan informasi produk.