

Nama :Nabila Wijaya

NPM :G1F022066

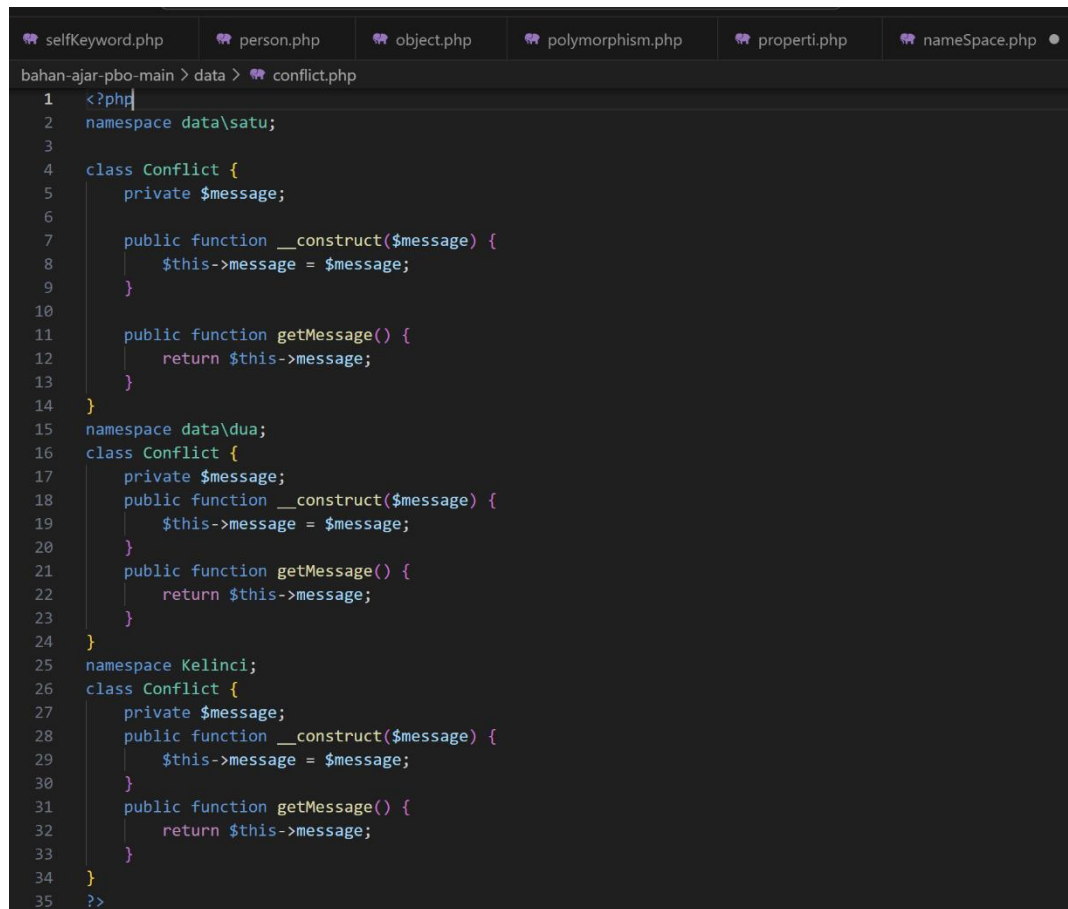
Responsi PBO

SOAL :

1. Silahkan lakukan git clone repositori dari <https://github.com/alzahfariski/bahan-ajar-pbo> (silahkan liat di youtube caranya).
2. lengkapi code php yang belum lengkap sehingga setiap file dapat di run dan tidak memunculkan error.
3. upload atau lakukakan git push ke akun git kalian masing-masing
4. 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.

PENJELASAN

Conflift



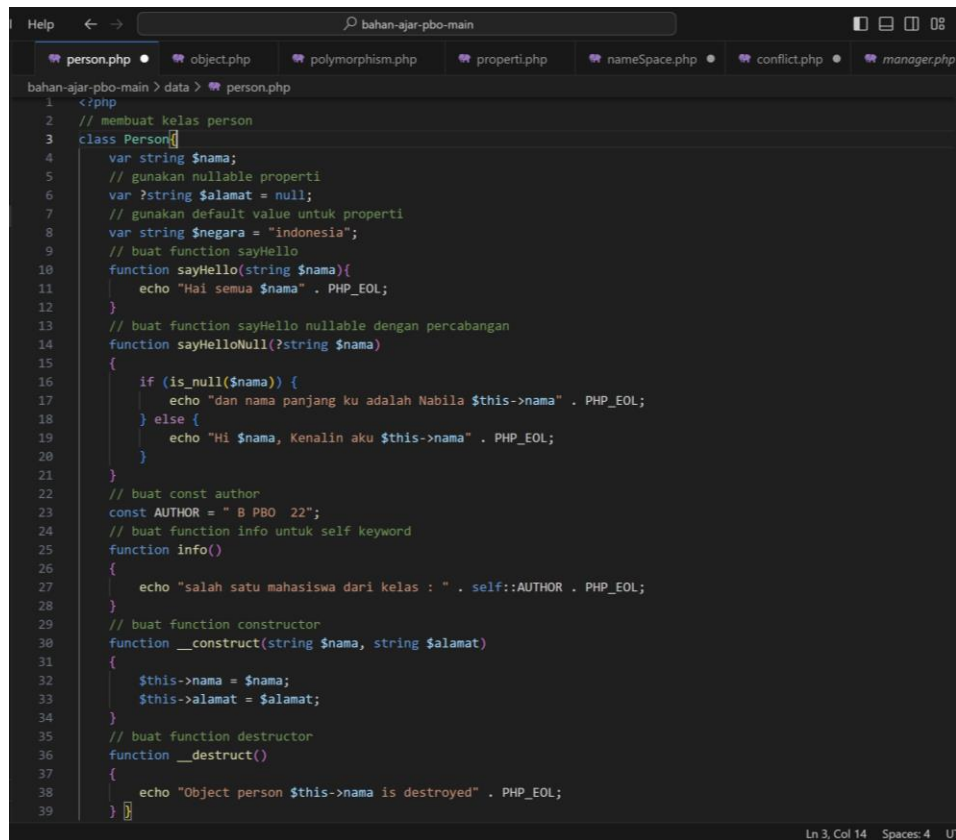
```
bahan-ajar-pbo-main > data > conflict.php
1  <?php
2  namespace data\satu;
3
4  class Conflict {
5      private $message;
6
7      public function __construct($message) {
8          $this->message = $message;
9      }
10
11     public function getMessage() {
12         return $this->message;
13     }
14 }
15 namespace data\dua;
16 class Conflict {
17     private $message;
18     public function __construct($message) {
19         $this->message = $message;
20     }
21     public function getMessage() {
22         return $this->message;
23     }
24 }
25 namespace Kelinci;
26 class Conflict {
27     private $message;
28     public function __construct($message) {
29         $this->message = $message;
30     }
31     public function getMessage() {
32         return $this->message;
33     }
34 }
35 ?>
```

Penjelasan :

Source code tersebut merupakan contoh penggunaan [namespace](#) dalam bahasa pemrograman PHP. Namespace digunakan untuk mengorganisir kode dalam kelompok tertentu, sehingga dapat menghindari konflik nama kelas atau fungsi yang sama.

Berikut adalah penjelasan naratif untuk source code tersebut: [Namespace data\satu](#) Namespace ini berisi kelas "Conflict" dengan properti message, konstruktor, dan metode getMessage. Namespace [data\dua](#) Namespace ini juga berisi kelas "Conflict" yang serupa dengan namespace [data\satu](#), tetapi keduanya terpisah dalam konteks namespace. [Namespace Kelinci](#) Namespace ini memiliki kelas "Conflict" yang mirip dengan dua namespace sebelumnya, tetapi berada dalam namespace yang berbeda yaitu Kelinci. Dengan menggunakan namespace, kita dapat membuat kelas dengan nama yang sama tanpa konflik, karena kelas-kelas tersebut berada dalam namespace yang berbeda. Pemisahan ini memungkinkan kita untuk mengorganisir kode dengan lebih baik dan menghindari kesulitan yang dapat timbul karena konflik nama. Untuk menggunakan kelas-kelas tersebut, kita bisa menggunakan use statement dan memberikan alias jika diperlukan untuk mengatasi ambiguitas nama

Person

A screenshot of a code editor window titled 'bahan-ajar-pbo-main'. The editor shows a PHP file named 'person.php' with the following code:

```
1 <?PHP
2 // membuat kelas person
3 class Person{
4     var string $nama;
5     // gunakan nullable properti
6     var ?string $alamat = null;
7     // gunakan default value untuk properti
8     var string $negara = "Indonesia";
9     // buat function sayHello
10    function sayHello(string $nama){
11        echo "Hai semua $nama" . PHP_EOL;
12    }
13    // buat function sayHello nullable dengan percabangan
14    function sayHelloNull(?string $nama)
15    {
16        if (is_null($nama)) {
17            echo "dan nama panjang ku adalah Nabila $this->nama" . PHP_EOL;
18        } else {
19            echo "Hi $nama, Kenalin aku $this->nama" . PHP_EOL;
20        }
21    }
22    // buat const author
23    const AUTHOR = " B PBO 22";
24    // buat function info untuk self keyword
25    function info()
26    {
27        echo "salah satu mahasiswa dari kelas : " . self::AUTHOR . PHP_EOL;
28    }
29    // buat function constructor
30    function __construct(string $nama, string $alamat)
31    {
32        $this->nama = $nama;
33        $this->alamat = $alamat;
34    }
35    // buat function destructor
36    function __destruct()
37    {
38        echo "Object person $this->nama is destroyed" . PHP_EOL;
39    }
}
```

Penjelasan :

Source code tersebut adalah contoh implementasi dalam bahasa pemrograman PHP untuk membuat sebuah kelas bernama "Person". Berikut penjelasan mengenai beberapa bagian dari source code tersebut: Mendefinisikan kelas dengan nama "Person".

\$nama: Sebuah properti dengan tipe data string yang menyimpan nama.

\$alamat: Sebuah properti nullable (bisa bernilai null) yang menyimpan alamat.

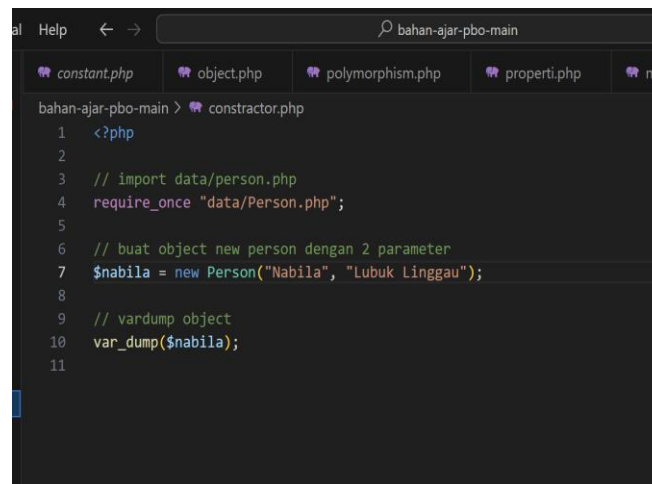
\$negara: Sebuah properti dengan nilai default "Indonesia" yang menyimpan informasi negara.

Fungsi sayHello Sebuah fungsi yang menerima parameter \$nama dan mencetak pesan sapaan.

Fungsi sayHelloNull Sebuah fungsi yang menerima parameter \$nama yang bisa bernilai null. Jika null, mencetak pesan dengan menggunakan properti \$nama dari objek saat ini. Jika tidak null, mencetak pesan sapaan. Konstanta Kelas atau **const AUTHOR** Sebuah konstanta kelas yang menyimpan informasi penulis atau pengguna kelas. **Function info** berfungsi yang mencetak informasi tentang kelas dan konstanta menggunakan self; dan **construct string nama, string Alamat**

Sebuah fungsi konstruktor yang dijalankan saat objek dibuat. Menginisialisasi properti `$nama` dan `$alamat` dengan nilai yang diterima. Dan terakhir `function destruct` Terdapat destruktur `__destruct` yang dijalankan saat objek `Person` dihancurkan (biasanya ketika program selesai dijalankan). Mencetak pesan yang memberi tahu bahwa objek `Person` telah dihancurkan. Dengan kombinasi properti, metode, konstanta, konstruktor, dan destruktur, kelas `"Person"` ini dapat digunakan untuk merepresentasikan individu dengan menyimpan informasi seperti nama, alamat, dan negara. Metode-metode yang ada memungkinkan untuk berinteraksi dan memberikan informasi terkait objek `Person`.

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 $nabila = new Person("Nabila", "Lubuk Linggau");
8
9 // vardump object
10 var_dump($nabila);
11
```

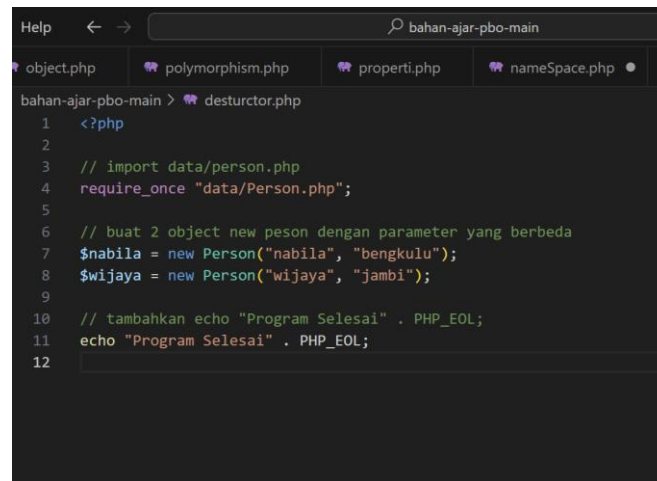
Penjelasan :

Source code tersebut mencakup dua bagian utama: penggunaan `require_once` untuk mengimpor (import) file `Person.php` dan pembuatan objek dari kelas `Person` dengan memberikan dua parameter pada saat instansiasi. Setelah itu, terdapat `var_dump` untuk menampilkan informasi tentang objek tersebut. `Import File Person` Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `Person.php`. `require_once` digunakan agar file hanya diimpor sekali, menghindari konflik jika sebelumnya telah diimpor. Pembuatan Objek `Person` Baris ini menciptakan sebuah objek baru dari kelas `Person` dengan nama variabel `$nabila`. Konstruktor dari kelas `Person` dipanggil dengan memberikan dua parameter: `"Nabila"` untuk nama dan `"Lubuk Linggau"` untuk alamat. `Var_dump Object` Baris ini menggunakan `var_dump` untuk menampilkan informasi rinci tentang objek `$nabila`. `var_dump` digunakan untuk debugging dan menampilkan tipe data dan nilai dari variabel atau ekspresi.

Dalam source code ini, pertama-tama, file `Person.php` diimpor menggunakan `require_once`. Kemudian, sebuah objek baru dari kelas `Person` dengan nama `"Nabila"` dan alamat `"Lubuk Linggau"` dibuat. Objek tersebut disimpan dalam variabel `$nabila`. Terakhir, informasi rinci tentang

objek tersebut ditampilkan menggunakan `var_dump`. Dengan cara ini, kita dapat menggunakan kelas `Person` dan membuat objek dari kelas tersebut dalam skrip PHP ini. [Import file Person.php](#) memungkinkan kita untuk menggunakan kelas tersebut dan mengakses fungsi atau properti yang didefinisikan di dalamnya.

Destructor

A screenshot of a code editor window titled 'bahan-ajar-pbo-main'. The editor shows a PHP file named 'destructor.php' with the following code:

```
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 $nabila = new Person("nabila", "bengkulu");
8 $wijaya = new Person("wijaya", "jambi");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
12
```

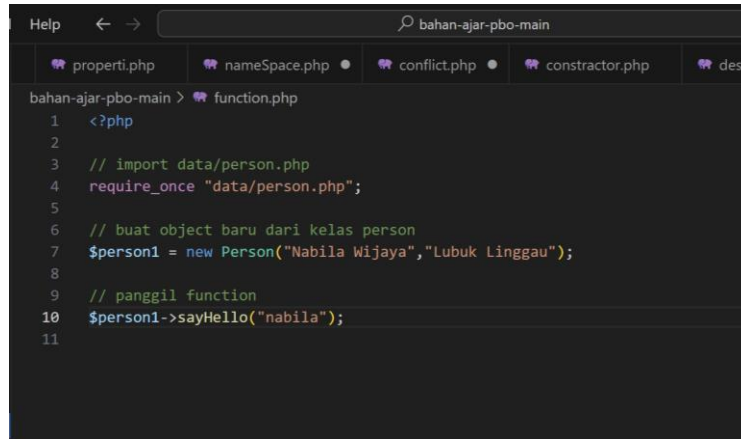
Penjelasan :

Source code tersebut melakukan beberapa langkah penting, yaitu mengimpor (import) file `Person.php`, membuat dua objek dari kelas `Person` dengan parameter yang berbeda, dan terakhir menampilkan pesan "Program Selesai" menggunakan `echo`. [Import File Person](#) Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `Person.php`. Ini memungkinkan penggunaan kelas `Person` yang didefinisikan di dalam file tersebut Pembuatan Dua Objek `Person` Baris ini menciptakan dua objek baru dari kelas `Person`. Objek pertama disimpan dalam variabel `$nabila` dengan parameter "nabila" untuk nama dan "bengkulu" untuk alamat. Objek kedua disimpan dalam variabel `$wijaya` dengan parameter "wijaya" untuk nama dan "jambi" untuk alamat Menampilkan Pesan Program Selesai Baris ini menggunakan `echo` untuk menampilkan pesan "Program Selesai" di layar. `PHP_EOL` digunakan untuk menambahkan karakter baris baru, sehingga pesan akan muncul di baris berikutnya.

Dalam source code ini, file `Person.php` diimpor agar kelas `Person` dapat digunakan. Selanjutnya, dua objek dari kelas `Person` dibuat dengan parameter yang berbeda untuk nama dan alamat. Objek-objek ini disimpan dalam variabel `$nabila` dan `$wijaya`. Terakhir, pesan "Program Selesai" ditampilkan di layar sebagai tanda bahwa program telah menyelesaikan eksekusinya.

Dengan cara ini, kita dapat membuat dan menggunakan objek dari kelas Person dengan memberikan nilai-parameter yang berbeda untuk menciptakan variasi objek.

Function



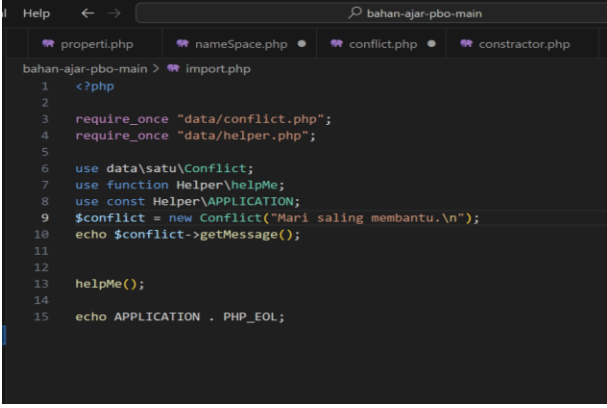
```
Help  <  >  bahan-ajar-pbo-main
property.php  namespace.php  conflict.php  constructor.php  desti
bahan-ajar-pbo-main > function.php
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person1 = new Person("Nabila Wijaya","Lubuk Linggau");
8
9  // panggil function
10 $person1->sayHello("nabila");
11
```

Penjelasan :

Source code ini terdiri dari beberapa langkah kunci yang mencakup penggunaan `require_once` untuk mengimpor file `person.php`, pembuatan objek baru dari kelas `Person` dengan memberikan parameter, dan pemanggilan metode `sayHello` dari objek yang telah dibuat. Import File `Person` Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `person.php`. Dengan demikian, kelas `Person` yang didefinisikan dalam file tersebut dapat digunakan di dalam script ini. Pembuatan Objek `Person` Baris ini menciptakan objek baru dari kelas `Person` dan menyimpannya dalam variabel `$person1`. Saat membuat objek, konstruktor kelas `Person` dipanggil dengan memberikan dua parameter, yaitu `"Nabila Wijaya"` untuk nama dan `"Lubuk Linggau"` untuk alamat. Pemanggilan Metode `sayHello` Pemanggilan Metode `sayHello`

Dalam source code ini, terlebih dahulu file `person.php` diimpor untuk mengakses kelas `Person`. Selanjutnya, sebuah objek baru dari kelas `Person` dibuat dengan nama `"Nabila Wijaya"` dan alamat `"Lubuk Linggau"`. Setelah itu, metode `sayHello` dari objek tersebut dipanggil dengan memberikan parameter `"nabila"`. Sebagai hasilnya, metode `sayHello` akan mencetak pesan sapaan ke layar dengan memanfaatkan parameter yang diberikan. Keseluruhan proses ini memperlihatkan bagaimana menggunakan objek dari kelas `Person` dan memanggil metode di dalamnya untuk berinteraksi dengan objek tersebut.

Import

A screenshot of a code editor window titled 'bahan-ajar-pbo-main'. The editor shows a file named 'import.php' with the following PHP code:

```
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("Mari saling membantu.\n");
10 echo $conflict->getMessage();
11
12
13 helpMe();
14
15 echo APPLICATION . PHP_EOL;
```

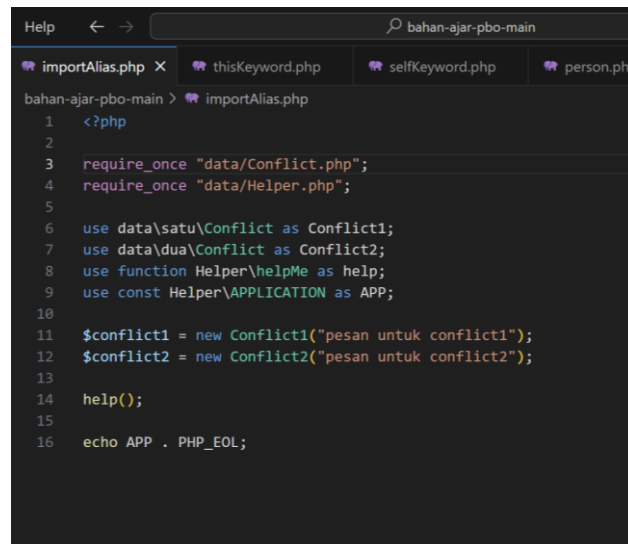
Penjelasan :

Source code ini melibatkan beberapa konsep, termasuk penggunaan `require_once` untuk mengimpor file `conflict.php` dan `helper.php`, penggunaan `use` untuk mengimpor namespace dan elemen-elemen tertentu (fungsi dan konstanta) dari file `helper.php`, serta pembuatan objek dan pemanggilan fungsi serta konstanta yang diimpor. Import File `conflict.php` dan `helper.php` Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `conflict.php` dan `helper.php`. Ini memungkinkan penggunaan kelas, fungsi, dan konstanta yang didefinisikan di dalam kedua file tersebut. Menggunakan Namespace dan Elemen Spesifik dari File `helper.php` Baris ini menggunakan pernyataan `use` untuk mengimpor namespace `data\satu\Conflict` dari file `conflict.php`, fungsi `helpMe` dari namespace `Helper`, dan konstanta `APPLICATION` dari namespace `Helper`. Pembuatan Objek dan Pemanggilan Metode Baris ini menciptakan objek baru dari kelas `Conflict` (dari namespace `data\satu`) dan menyimpannya dalam variabel `$conflict`. Selanjutnya, metode `getMessage` dari objek tersebut dipanggil dan hasilnya ditampilkan ke layar. Pemanggilan Fungsi dari `Helper Namespace` Baris ini memanggil fungsi `helpMe` yang diimpor dari namespace `Helper`. Fungsi ini mungkin berisi implementasi tertentu yang tidak terlihat dalam source code yang diberikan. Pemanggilan Konstanta dari `Helper Namespace` Baris ini menampilkan nilai dari konstanta `APPLICATION` yang diimpor dari namespace `Helper`. `PHP_EOL` digunakan untuk menambahkan karakter baris baru setelah nilai konstanta.

Dalam source code ini, file `conflict.php` dan `helper.php` diimpor untuk digunakan. Selanjutnya, namespace dan elemen-elemen spesifik dari file `helper.php` diimpor menggunakan pernyataan `use`. Objek baru dari kelas `Conflict` dibuat dan metodenya dipanggil, selanjutnya fungsi dan konstanta yang diimpor dari namespace `Helper` juga dipanggil dan ditampilkan hasilnya ke

layar. Keseluruhan proses ini menunjukkan bagaimana menggunakan dan mengimpor elemen-elemen dari berbagai file dan namespace dalam PHP.

ImportAlias



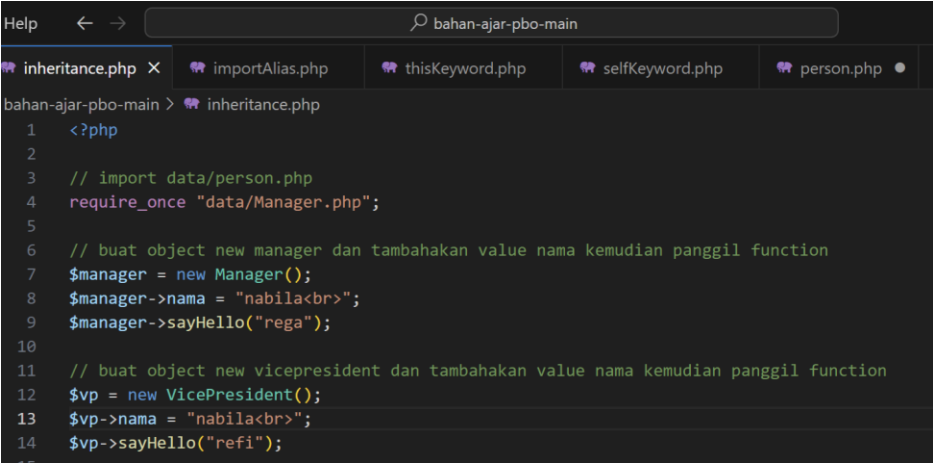
```
1 <?php
2
3 require_once "data/Conflict.php";
4 require_once "data/Helper.php";
5
6 use data\satu\Conflict as Conflict1;
7 use data\dua\Conflict as Conflict2;
8 use function Helper\helpMe as help;
9 use const Helper\APPLICATION as APP;
10
11 $conflict1 = new Conflict1("pesan untuk conflict1");
12 $conflict2 = new Conflict2("pesan untuk conflict2");
13
14 help();
15
16 echo APP . PHP_EOL;
```

Penjelasan :

Source code ini melibatkan beberapa konsep, termasuk penggunaan `require_once` untuk mengimpor file `Conflict.php` dan `Helper.php`, penggunaan `use` untuk mengimpor namespace dan elemen-elemen tertentu (kelas, fungsi, dan konstanta) dari file `Conflict.php` dan `Helper.php`, serta pembuatan objek dan pemanggilan fungsi serta konstanta yang diimpor. Import File `Conflict.php` dan `Helper.php`, Menggunakan Namespace dan Elemen Spesifik dari File `Conflict.php` dan `Helper.php` Baris ini menggunakan pernyataan `use` untuk mengimpor namespace dan elemen-elemen spesifik dari file `Conflict.php` dan `Helper.php`. Alias (alias) juga digunakan untuk menghindari konflik nama, Pembuatan Objek dan Pemanggilan Metode Baris ini menciptakan dua objek, yaitu `$conflict1` dan `$conflict2`, dari dua kelas yang berbeda, `Conflict1` dan `Conflict2`. Masing-masing objek diberikan pesan sebagai parameter pada saat instansiasi, Pemanggilan Fungsi dari `Helper` Namespace Baris ini memanggil fungsi `helpMe` yang diimpor dari namespace `Helper` dengan menggunakan alias `help`, Pemanggilan Konstanta dari `Helper` Namespace Baris ini menampilkan nilai dari konstanta `APPLICATION` yang diimpor dari namespace `Helper` dengan menggunakan alias `APP`. `PHP_EOL` digunakan untuk menambahkan karakter baris baru setelah nilai konstanta.

Dalam source code ini, file [Conflict.php](#) dan [Helper.php](#) diimpor untuk digunakan. Selanjutnya, namespace dan elemen-elemen spesifik dari file tersebut diimpor menggunakan pernyataan [use](#). Dua objek dari kelas yang berbeda dibuat, dan metodenya dipanggil. Fungsi dan konstanta yang diimpor dari namespace Helper juga dipanggil dan hasilnya ditampilkan ke layar. Keseluruhan proses ini menunjukkan bagaimana menggunakan dan mengimpor elemen-elemen dari berbagai file dan namespace dalam PHP, dengan memperhatikan penanganan konflik nama menggunakan alias.

Inheritance



```
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->nama = "nabila<br>";
9 $manager->sayHello("rega");
10
11 // buat object new vicepresident dan tambahkan value nama kemudian panggil function
12 $vp = new VicePresident();
13 $vp->nama = "nabila<br>";
14 $vp->sayHello("refi");
```

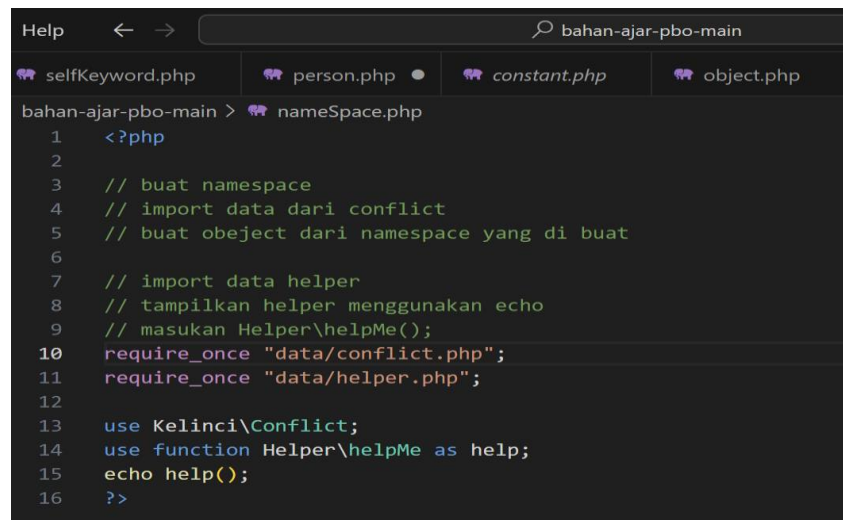
Penjelasan :

Source code tersebut melibatkan pembuatan objek dari dua kelas, yaitu Manager dan VicePresident. Import File [Manager.php](#) Baris ini menggunakan [require_once](#) untuk mengimpor (menggunakan) file [Manager.php](#). Ini memungkinkan penggunaan kelas Manager yang didefinisikan dalam file tersebut. Pembuatan Objek Manager dan Pemanggilan Metode Baris ini menciptakan objek baru dari kelas Manager dan menyimpannya dalam variabel `$manager`. Selanjutnya, nilai properti `$nama` dari objek diisi dengan "nabila", dan metode [sayHello](#) dipanggil dengan parameter "rega". Pembuatan Objek [VicePresident](#) dan Pemanggilan Metode Baris ini menciptakan objek baru dari kelas [VicePresident](#) dan menyimpannya dalam variabel `$vp`. Nilai properti `$nama` dari objek diisi dengan "nabila", dan metode [sayHello](#) dipanggil dengan parameter "refi".

Dalam source code ini, terlebih dahulu file [Manager.php](#) diimpor untuk menggunakan kelas Manager. Selanjutnya, objek baru dari kelas Manager dan VicePresident dibuat, masing-masing

disimpan dalam variabel `$manager` dan `$vp`. Properti `$nama` dari kedua objek diisi dengan nilai "nabila", dan metode `sayHello` dipanggil untuk masing-masing objek dengan parameter yang berbeda ("rega" untuk Manager dan "refi" untuk VicePresident). Kelas-kelas Manager dan VicePresident mungkin memiliki implementasi khusus untuk metode `sayHello` dan mungkin merupakan turunan dari kelas lain. Namun, potongan kode yang diberikan tidak memberikan detail terkait implementasi kelas-kelas tersebut

Namespace



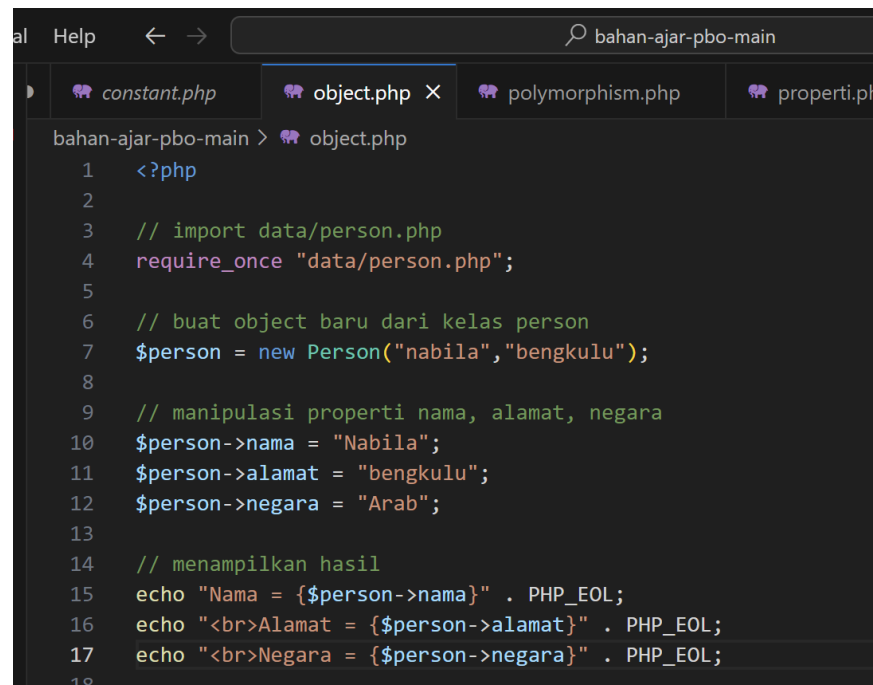
```
Help  <  >  bahan-ajar-pbo-main
selfKeyword.php  person.php  constant.php  object.php
bahan-ajar-pbo-main > namespace.php
1  <?php
2
3  // buat namespace
4  // import data dari conflict
5  // buat oboject dari namespace yang di buat
6
7  // import data helper
8  // tampilkan helper menggunakan echo
9  // masukan Helper\helpMe();
10 require_once "data/conflict.php";
11 require_once "data/helper.php";
12
13 use Kelinci\Conflict;
14 use function Helper\helpMe as help;
15 echo help();
16 ?>
```

Penjelasan :

Source code ini menggunakan namespace dan mengimpor elemen dari dua file, yaitu `conflict.php` dan `helper.php`. Setelah mengimpor, objek dari namespace Kelinci (dari file `conflict.php`) dibuat, dan fungsi `helpMe` dari namespace Helper (dari file `helper.php`) dipanggil. Import File `conflict.php` Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `conflict.php`. Ini memungkinkan penggunaan namespace dan elemen-elemen yang didefinisikan dalam file tersebut Membuat Objek dari Namespace Kelinci Baris ini menggunakan pernyataan `use` untuk mengimpor namespace Kelinci dari file `conflict.php`. Selanjutnya, objek dari kelas `Conflict` (yang terdapat dalam namespace Kelinci) dibuat Import File `helper.php` dan Pemanggilan Fungsi Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `helper.php`. Selanjutnya, pernyataan `use` digunakan untuk mengimpor fungsi `helpMe` dari namespace Helper dan memberikan alias `help` pada fungsi tersebut. Pemanggilan Fungsi dan Tampilkan dengan `echo`, Baris ini memanggil fungsi `helpMe` (yang diimpor dengan alias `help`) dari namespace Helper dan menampilkannya ke layar menggunakan `echo`

Dalam source code ini, file [conflict.php](#) diimpor untuk menggunakan [namespace Kelinci](#) dan kelas Conflict yang ada di dalamnya. Selanjutnya, file [helper.php](#) diimpor untuk menggunakan [namespace Helper](#) dan fungsi [helpMe](#). Fungsi ini dipanggil dan hasilnya ditampilkan ke layar dengan menggunakan [echo](#). Keseluruhan proses ini menunjukkan cara menggunakan namespace dan mengimpor elemen-elemen dari file terpisah dalam PHP

Object



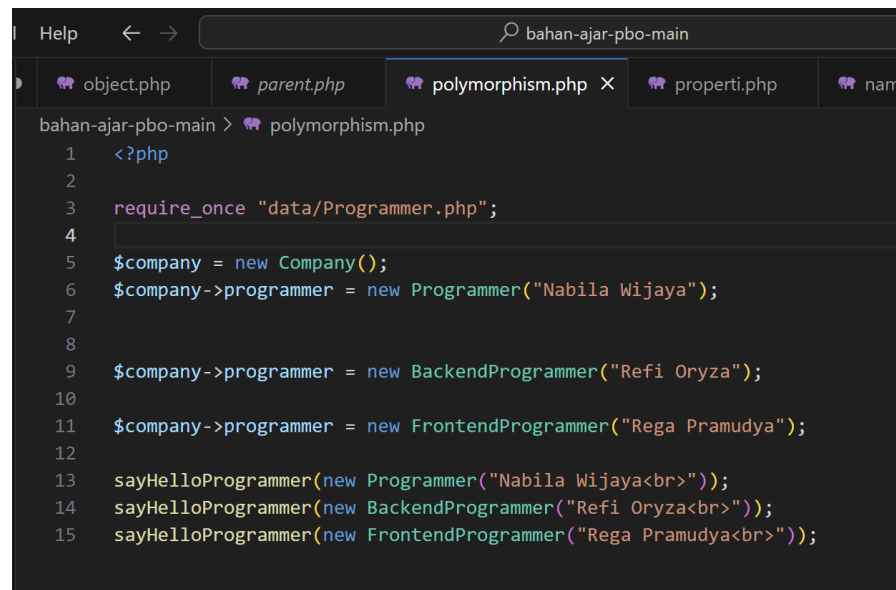
```
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person = new Person("nabila","bengkulu");
8
9  // manipulasi properti nama, alamat, negara
10 $person->nama = "Nabila";
11 $person->alamat = "bengkulu";
12 $person->negara = "Arab";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

Penjelasan :

Source code ini melakukan beberapa langkah dasar dalam pemrograman berorientasi objek menggunakan PHP Import File [person.php](#), Baris ini menggunakan [require_once](#) untuk mengimpor (menggunakan) file [person.php](#). Ini memungkinkan penggunaan kelas [Person](#) yang didefinisikan dalam file tersebut. Pembuatan Objek dari Kelas [Person](#), Baris ini menciptakan objek baru dari kelas [Person](#) dan menyimpannya dalam variabel [\\$person](#). Konstruktork dari kelas [Person](#) dipanggil dengan memberikan dua parameter: "nabila" untuk nama dan "bengkulu" untuk Alamat, Manipulasi Properti Nama, Alamat, dan Negara, Baris ini melakukan manipulasi properti dari objek [\\$person](#). Properti nama diubah menjadi "Nabila", properti [alamat](#) diubah menjadi "bengkulu", dan properti [negara](#) diubah menjadi "Indonesia". Menampilkan Hasil Manipulasi Baris ini menggunakan [echo](#) untuk menampilkan hasil manipulasi properti objek [\\$person](#). Nilai properti [nama](#), [alamat](#), dan [negara](#) ditampilkan ke layer.

Dalam source code ini, file `person.php` diimpor untuk menggunakan kelas `Person`. Selanjutnya, objek baru dari kelas `Person` dibuat dengan memberikan nilai-parameter pada saat instansiasi. Properti dari objek tersebut kemudian dimanipulasi dengan memberikan nilai baru. Akhirnya, hasil manipulasi properti ditampilkan ke layar menggunakan `echo`. Keseluruhan proses ini memberikan gambaran tentang cara membuat objek, mengakses propertinya, dan melakukan manipulasi pada properti objek dalam PHP.

PolyMorphism

A screenshot of a code editor window titled 'bahan-ajar-pbo-main'. The editor shows a file named 'polymorphism.php' with the following PHP code:

```
1 <?php
2
3 require_once "data/Programmer.php";
4
5 $company = new Company();
6 $company->programmer = new Programmer("Nabila Wijaya");
7
8
9 $company->programmer = new BackendProgrammer("Refi Oryza");
10
11 $company->programmer = new FrontendProgrammer("Rega Pramudya");
12
13 sayHelloProgrammer(new Programmer("Nabila Wijaya<br>"));
14 sayHelloProgrammer(new BackendProgrammer("Refi Oryza<br>"));
15 sayHelloProgrammer(new FrontendProgrammer("Rega Pramudya<br>"));
```

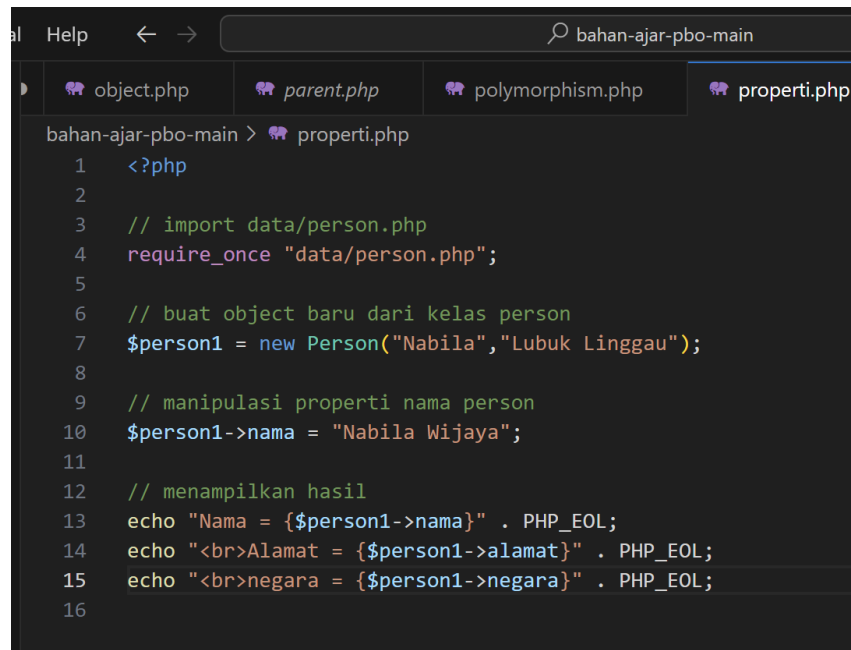
Penjelasan :

Import File `Programmer.php`, Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `Programmer.php`. Ini memungkinkan penggunaan kelas `Programmer`, `BackendProgrammer`, `FrontendProgrammer`, dan `Company` yang didefinisikan dalam file tersebut. Pembuatan Objek `Company` dan `Programmer`, Baris ini menciptakan objek `$company` dari kelas `Company`. Selanjutnya, objek `$programmer` dari kelas `Programmer` (dengan nama "Nabila Wijaya") disimpan dalam properti `programmer` dari objek `$company`, Pembaruan Objek `Programmer` dalam `Company`, Baris ini memperbarui properti `programmer` dari objek `$company` dengan objek baru dari kelas `BackendProgrammer` dan `FrontendProgrammer`. Ini menunjukkan bahwa properti `programmer` dapat mengacu pada objek dari kelas yang berbeda. Pemanggilan Fungsi `sayHelloProgrammer`, Baris ini memanggil fungsi `sayHelloProgrammer` dengan memberikan objek dari kelas `Programmer`, `BackendProgrammer`, dan `FrontendProgrammer`.

sebagai parameter. Fungsi ini mungkin memiliki implementasi tertentu yang tidak terlihat dalam potongan kode yang diberikan.

Dalam source code ini, file Programmer.php diimpor untuk menggunakan beberapa kelas terkait pemrograman. Objek dari kelas Company dan beberapa objek dari kelas yang berbeda (Programmer, BackendProgrammer, FrontendProgrammer) dibuat dan dimanipulasi. Kemudian, fungsi sayHelloProgrammer dipanggil dengan memberikan objek dari berbagai kelas sebagai parameter. Keseluruhan proses ini memberikan gambaran tentang bagaimana menggunakan kelas, objek, dan pemrograman berorientasi objek dalam PHP

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("Nabila", "Lubuk Linggau");
8
9  // manipulasi properti nama person
10 $person1->nama = "Nabila Wijaya";
11
12 // menampilkan hasil
13 echo "Nama = {$person1->nama}" . PHP_EOL;
14 echo "<br>Alamat = {$person1->alamat}" . PHP_EOL;
15 echo "<br>negara = {$person1->negara}" . PHP_EOL;
16
```

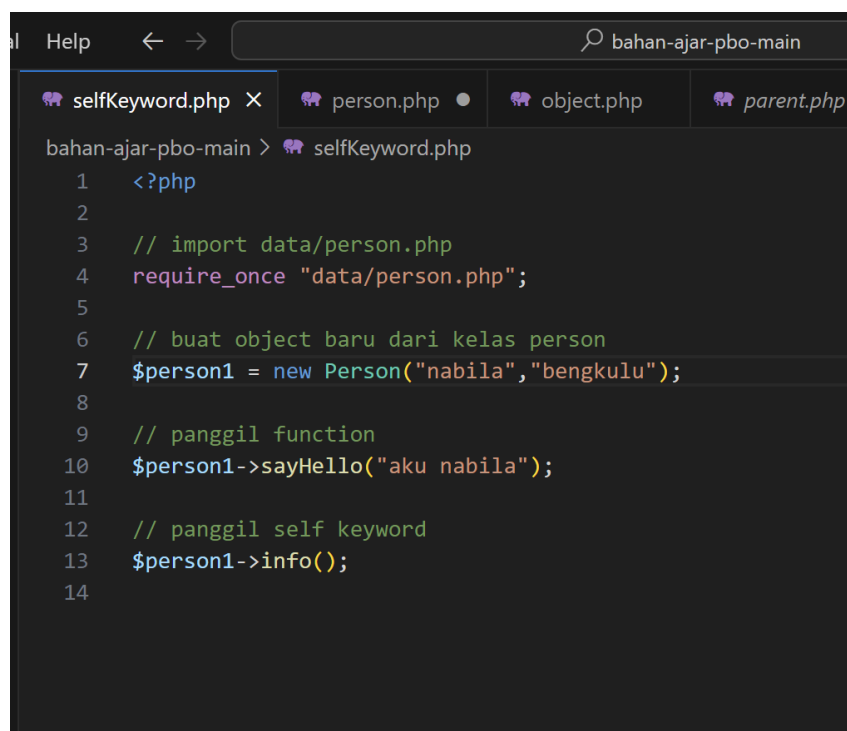
Penjelasan:

Source code di atas adalah contoh implementasi pemrograman berorientasi objek (OOP) dalam PHP, yang melibatkan pembuatan objek dari kelas Person dan manipulasi propertinya. Import File `person.php`, Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `person.php`. Ini memungkinkan penggunaan kelas Person yang didefinisikan dalam file tersebut. Pembuatan Objek dari Kelas Person, Baris ini menciptakan objek baru dari kelas Person dan menyimpannya dalam variabel `$person1`. Konstruktors dari kelas Person dipanggil dengan memberikan dua parameter: "Nabila" untuk nama dan "Lubuk Linggau" untuk Alamat Manipulasi

Properti Nama `Person`, Baris ini melakukan manipulasi properti `nama` dari objek `$person1`. Nilai properti `nama` diubah menjadi "Nabila Wijaya" Menampilkan Hasil Manipulasi Properti, Baris ini menggunakan `echo` untuk menampilkan hasil manipulasi properti objek `$person1`. Nilai properti `nama`, `alamat`, dan `negara` ditampilkan ke layar. Dan penggunaan `
` untuk pembuatan garis baru.

Dalam source code ini, file `person.php` diimpor untuk menggunakan kelas `Person`. Selanjutnya, objek baru dari kelas `Person` dibuat dengan memberikan nilai-parameter pada saat instansiasi. Properti dari objek tersebut kemudian dimanipulasi dengan memberikan nilai baru. Akhirnya, hasil manipulasi properti ditampilkan ke layar menggunakan `echo`. Keseluruhan proses ini memberikan gambaran tentang cara membuat objek, mengakses propertinya, dan melakukan manipulasi pada properti objek dalam PHP

SelfKeyword



```
Help  <  >  bahan-ajar-pbo-main
selfKeyword.php X  person.php ●  object.php  parent.php
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("nabila","bengkulu");
8
9  // panggil function
10 $person1->sayHello("aku nabila");
11
12 // panggil self keyword
13 $person1->info();
14
```

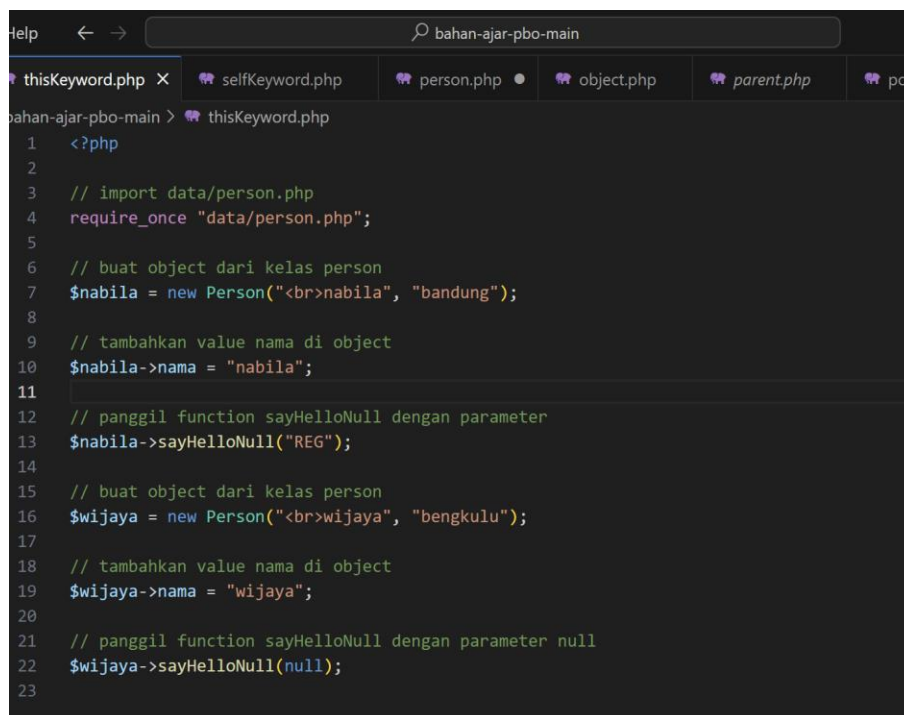
Penjelasan :

Source code di atas merupakan contoh implementasi pemrograman berorientasi objek (OOP) dalam PHP, yang melibatkan pembuatan objek dari kelas `Person` dan pemanggilan beberapa metode. Import File `person.php`, Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `person.php`. Ini memungkinkan penggunaan kelas `Person` yang didefinisikan

dalam file tersebut Pembuatan Objek dari Kelas **Person**, Baris ini menciptakan objek baru dari kelas **Person** dan menyimpannya dalam variabel **\$person1**. Konstruktor dari kelas **Person** dipanggil dengan memberikan dua parameter: "nabila" untuk nama dan "bengkulu" untuk Alamat, Panggilan Metode **sayHello**, Baris ini memanggil metode **sayHello** dari objek **\$person1** dengan memberikan parameter "aku nabila". Metode ini kemungkinan menampilkan pesan sapaan sesuai dengan nilai parameter yang diberikan. Panggilan Metode **info** dengan Self Keyword Baris ini memanggil metode **info** dari objek **\$person1**. Metode ini menggunakan **self**; untuk merujuk ke konstanta **AUTHOR** yang mungkin didefinisikan di dalam kelas **Person**. Konstanta ini mungkin berisi informasi tentang penulis atau informasi kelas.

Dalam source code ini, file **person.php** diimpor untuk menggunakan kelas **Person**. Selanjutnya, objek baru dari kelas **Person** dibuat dengan memberikan nilai-parameter pada saat instansiasi. Metode **sayHello** dipanggil dengan memberikan parameter, dan metode **info** dipanggil untuk menampilkan informasi yang mungkin berisi data kelas atau konstanta. Keseluruhan proses ini memberikan gambaran tentang cara membuat objek, memanggil metode, dan penggunaan **self** keyword dalam PHP

ThisKeyWord



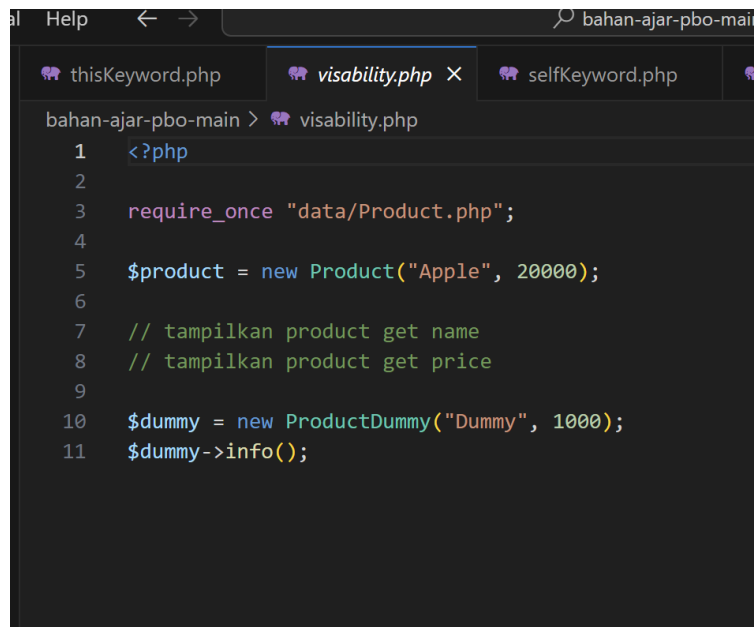
```
Help  <  >  bahan-ajar-pbo-main
thisKeyword.php x  selfKeyword.php  person.php  object.php  parent.php  poly
bahan-ajar-pbo-main > thisKeyword.php
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object dari kelas person
7  $nabila = new Person("<br>nabila", "bandung");
8
9  // tambahkan value nama di object
10 $nabila->nama = "nabila";
11
12 // panggil function sayHelloNull dengan parameter
13 $nabila->sayHelloNull("REG");
14
15 // buat object dari kelas person
16 $wijaya = new Person("<br>wijaya", "bengkulu");
17
18 // tambahkan value nama di object
19 $wijaya->nama = "wijaya";
20
21 // panggil function sayHelloNull dengan parameter null
22 $wijaya->sayHelloNull(null);
23
```

Penjelasan :

Source code di atas adalah contoh implementasi pemrograman berorientasi objek (OOP) dalam PHP, yang melibatkan pembuatan objek dari kelas `Person` dan pemanggilan beberapa metode. Import File `person.php`, Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `person.php`. Ini memungkinkan penggunaan kelas `Person` yang didefinisikan dalam file tersebut, Pembuatan Objek dari Kelas `Person` (`nabila`) Baris ini menciptakan objek baru dari kelas `Person` dan menyimpannya dalam variabel `$nabila`. Konstruktor dari kelas `Person` dipanggil dengan memberikan dua parameter: "`
nabila`" untuk nama dan "`bandung`" untuk Alamat. Penambahan Nilai Nama di Objek `$nabila` Baris ini menambahkan nilai properti nama dari objek `$nabila` menjadi "`nabila`". Panggilan Metode `sayHelloNull` dengan Parameter Baris ini memanggil metode `sayHelloNull` dari objek `$nabila` dengan memberikan parameter "`REG`". Metode ini mungkin memiliki implementasi yang berbeda tergantung pada nilai parameter yang diberikan, Pembuatan Objek dari Kelas `Person` (`wijaya`), Baris ini menciptakan objek baru dari kelas `Person` dan menyimpannya dalam variabel `$wijaya`. Konstruktor dari kelas `Person` dipanggil dengan memberikan dua parameter: "`
wijaya`" untuk nama dan "`bengkulu`" untuk Alamat, Penambahan Nilai Nama di Objek `$wijaya`, Baris ini menambahkan nilai properti nama dari objek `$wijaya` menjadi "`wijaya`". Panggilan Metode `sayHelloNull` dengan Parameter Null, Baris ini memanggil metode `sayHelloNull` dari objek `$wijaya` dengan memberikan parameter null. Metode ini mungkin memiliki implementasi yang berbeda tergantung pada apakah nilai parameter adalah null atau tidak.

Dalam source code ini, file `person.php` diimpor untuk menggunakan kelas `Person`. Selanjutnya, dua objek baru dari kelas `Person` dibuat dengan memberikan nilai-parameter pada saat instansiasi. Nilai properti nama dari kedua objek tersebut kemudian dimanipulasi dengan memberikan nilai baru. Metode `sayHelloNull` dipanggil dengan memberikan parameter pada objek `$nabila` dan dengan parameter null pada objek `$wijaya`. Keseluruhan proses ini memberikan gambaran tentang cara membuat objek, mengakses propertinya, dan melakukan pemanggilan metode dengan parameter berbeda dalam PHP.

Visability



```
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();
```

Penjelasan :

Source code tersebut adalah contoh implementasi pemrograman berorientasi objek (OOP) dalam PHP, yang melibatkan pembuatan objek dari kelas `Product` dan `ProductDummy`, serta pemanggilan beberapa metode. Import File `Product.php`, Baris ini menggunakan `require_once` untuk mengimpor (menggunakan) file `Product.php`. Ini memungkinkan penggunaan kelas `Product` dan `ProductDummy` yang didefinisikan dalam file tersebut, Pembuatan Objek dari Kelas `Product` Baris ini menciptakan objek `$product` dari kelas `Product` dan menyimpannya dalam variabel. Konstruktor dari kelas `Product` dipanggil dengan memberikan dua parameter: "Apple" untuk nama produk dan 20000 untuk harga, Pemanggilan Metode `getName` dan `getPrice` Baris ini menggunakan metode `getName` dan `getPrice` dari objek `$product` untuk menampilkan nama produk dan harga ke layar. Metode ini mungkin mengembalikan nilai yang sesuai dengan properti `name` dan `price` objek `Product`, Pembuatan Objek dari Kelas `ProductDummy`, Baris ini menciptakan objek `$dummy` dari kelas `ProductDummy` dan menyimpannya dalam variabel. Konstruktor dari kelas `ProductDummy` dipanggil dengan memberikan dua parameter: "Dummy" untuk nama produk dan 1000 untuk harga, Pemanggilan Metode `info` pada `ProductDummy`, Baris ini memanggil metode `info` dari objek `$dummy`. Metode ini mungkin memiliki implementasi khusus yang menampilkan informasi tambahan terkait dengan objek `ProductDummy`.

Dalam source code ini, file `Product.php` diimpor untuk menggunakan kelas `Product` dan `ProductDummy`. Objek dari kelas `Product` dan `ProductDummy` dibuat dengan memberikan nilai-parameter pada saat instansiasi. Metode `getName` dan `getPrice` dipanggil untuk menampilkan informasi produk dari objek `$product`. Metode `info` dari objek `$dummy` juga dipanggil, yang mungkin memberikan informasi tambahan tergantung pada implementasinya. Keseluruhan proses ini memberikan gambaran tentang cara membuat objek, mengakses propertinya, dan melakukan pemanggilan metode dalam PHP.