

**NAMA : Dian Khairani**

**NPM : G1F022004**

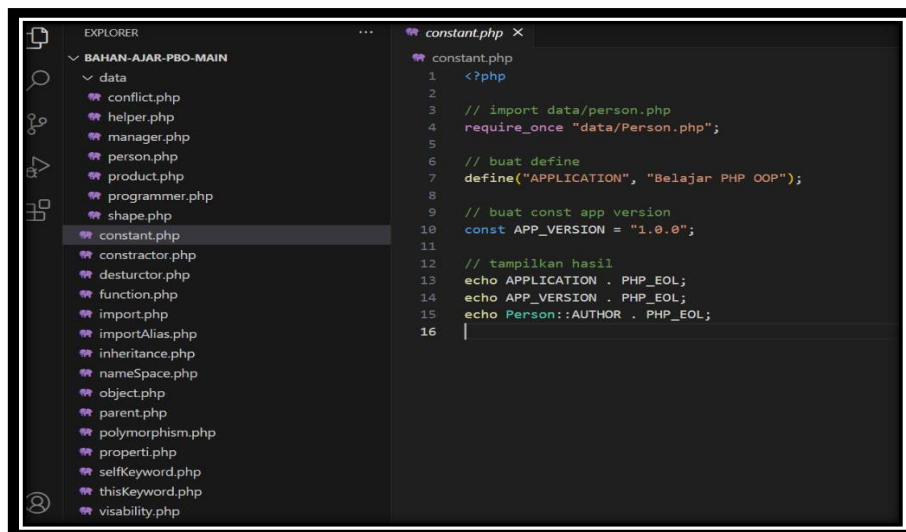
## **Responsi PBO**

### **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 !**

#### **1. File constant.php**

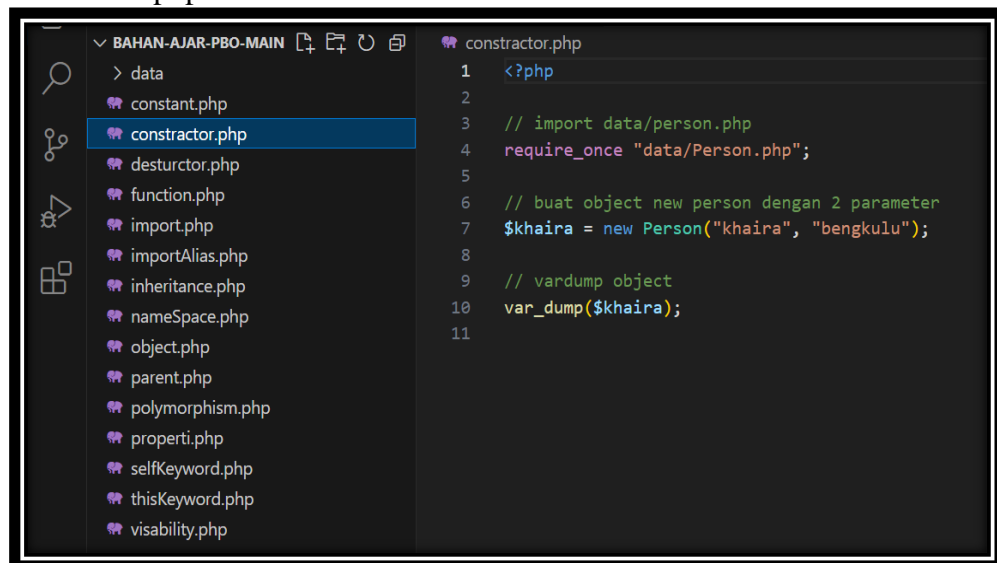


### **Penjelasan :**

1. **require\_once "data/Person.php";**: Mengimpor (include) file "Person.php" yang terletak di direktori "data". Menggunakan require\_once untuk memastikan file hanya diimpor satu kali.

2. **define("APPLICATION", "Belajar PHP OOP");** Mendefinisikan konstanta dengan nama "APPLICATION" dan nilai "Belajar PHP OOP". Konstanta dapat diakses di seluruh skrip dan nilainya tidak dapat diubah.
3. **const APP\_VERSION = "1.0.0";** Mendefinisikan konstanta dengan nama "APP\_VERSION" dan nilai "1.0.0". Konstanta menggunakan kata kunci **const** dan juga dapat diakses di seluruh skrip.
4. **echo APPLICATION . PHP\_EOL;** Mencetak nilai konstanta "APPLICATION" ke layar, diikuti dengan karakter baris baru (PHP\_EOL).
5. **echo APP\_VERSION . PHP\_EOL;** Mencetak nilai konstanta "APP\_VERSION" ke layar, diikuti dengan karakter baris baru.
6. **echo Person::AUTHOR . PHP\_EOL;** Mencetak nilai konstanta "AUTHOR" dari kelas "Person" ke layar. Menggunakan **::** untuk mengakses konstanta atau properti statis dari kelas tanpa membuat instansi kelas.

## 2. File constactor.php

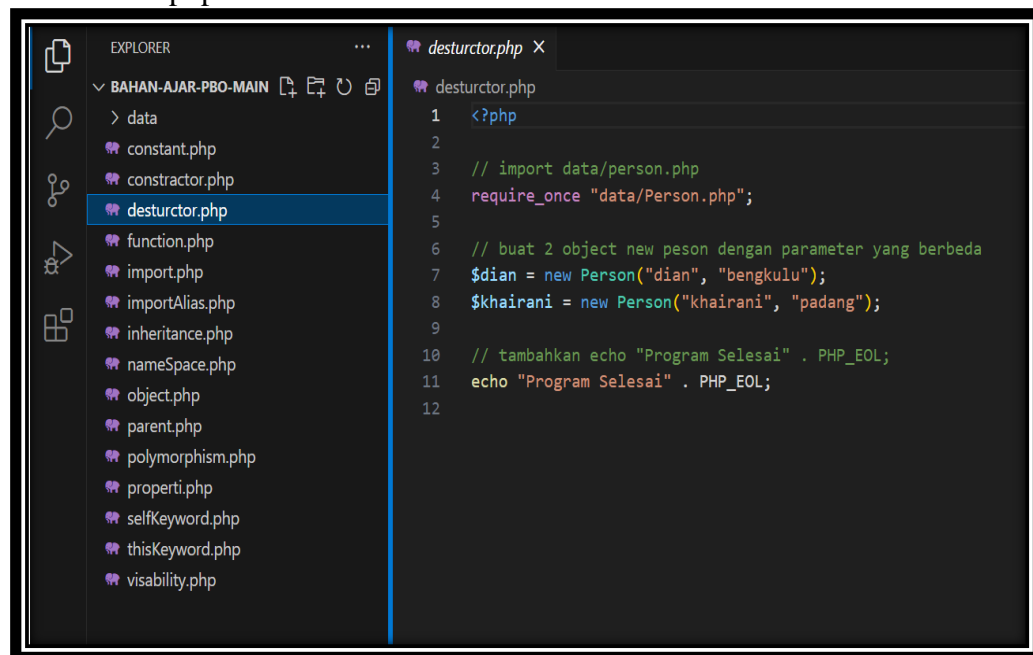


### Penjelasan :

1. **require\_once "data/Person.php";** Menyertakan (import) file "Person.php" yang terletak di direktori "data".

2. **\$khaira = new Person("khaira", "bengkulu");**: Membuat objek baru dari kelas "Person" dengan memberikan dua parameter ("khaira" dan "bengkulu") ke konstruktor kelas.
3. **var\_dump(\$khaira);**: Menggunakan fungsi `var_dump` untuk menampilkan informasi terstruktur tentang objek yang baru dibuat, dalam hal ini, objek dengan nama "\$khaira".

### 3. File destrutor.php



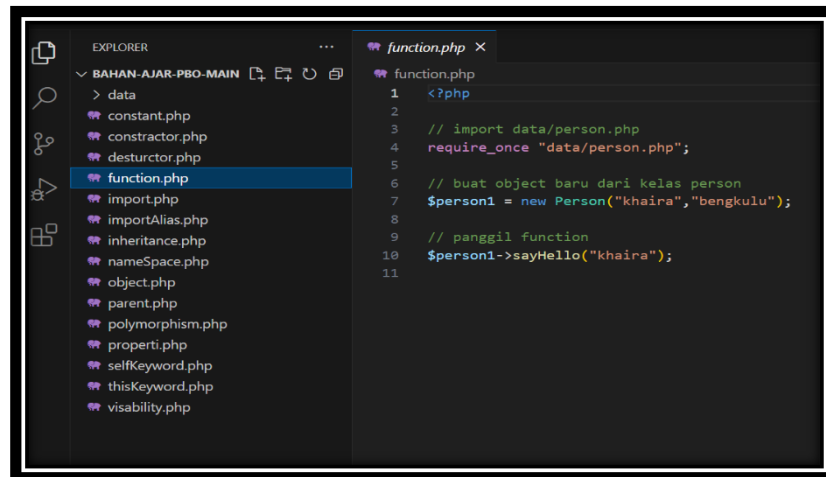
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with a file named `destrutor.php` selected. The code editor shows the following PHP code:

```
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 $dian = new Person("dian", "bengkulu");
8 $khairani = new Person("khairani", "padang");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
12
```

#### Penjelasan :

1. **require\_once "data/Person.php";**: Menyertakan (import) file "Person.php" yang berisi definisi kelas "Person" dari direktori "data".
2. **\$gita = new Person("dian", "bengkulu");**: Membuat objek baru dari kelas "Person" dengan nama "\$dian" dan memberikan dua parameter ("dian" dan "bengkulu") ke konstruktor kelas.
3. **\$pita = new Person("khairani", "padang");**: Membuat objek baru lagi dari kelas "Person" dengan nama "\$khairani" dan parameter yang berbeda ("khairani" dan "padang").
4. **echo "Program Selesai" . PHP\_EOL;**: Mencetak teks "Program Selesai" ke layar, diikuti dengan karakter baris baru (PHP\_EOL untuk memastikan hasil cetakan terformat dengan baik).

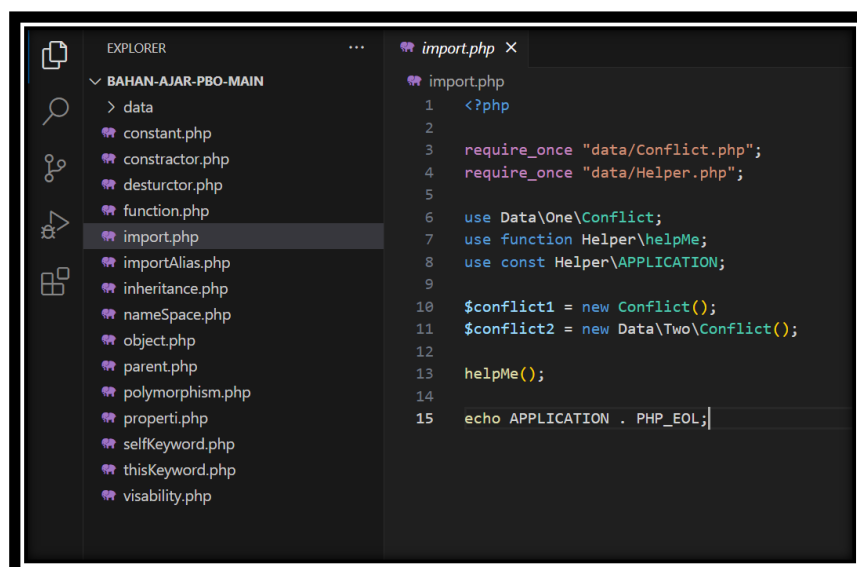
#### 4. File function.php



#### Penjelasan :

1. **require\_once "data/person.php";**: Mengimpor (include) file "person.php" yang berisi definisi kelas "Person". Menggunakan **require\_once** untuk memastikan file hanya diimpor satu kali.
2. **\$person1 = new Person("khaira", "bengkulu");**: Membuat objek baru dari kelas "Person" dengan nama "\$person1" dan memberikan dua argumen, yaitu "khaira" sebagai nama dan "bengkulu" sebagai lokasi, kepada konstruktor kelas "Person".
3. **\$person1->sayHello("khaira");**: Memanggil metode "sayHello" dari objek "\$person1" dengan memberikan argumen "khaira".

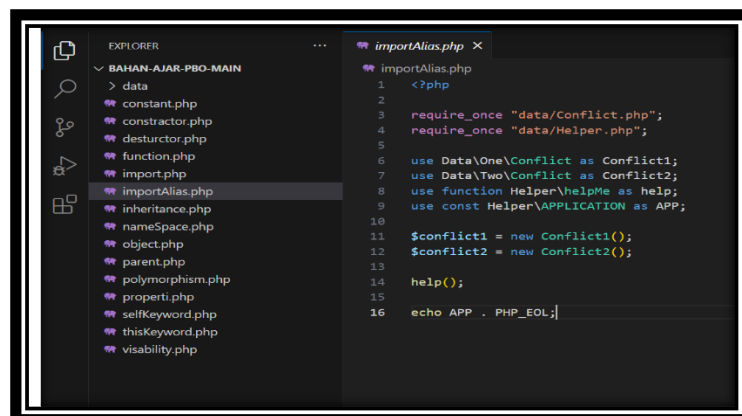
#### 5. File import.php



### Penjelasan :

1. **require\_once "data/Conflict.php";** Mengimpor (include) file "Conflict.php" dari direktori "data".
2. **require\_once "data/Helper.php";** Mengimpor (include) file "Helper.php" dari direktori "data".
3. **use Data\One\Conflict;** Menggunakan namespace "Data\One" dan mengimpor kelas "Conflict" ke dalam skrip.
4. **use function Helper\helpMe;** Menggunakan namespace "Helper" dan mengimpor fungsi "helpMe" ke dalam skrip.
5. **use const Helper\APPLICATION;** Menggunakan namespace "Helper" dan mengimpor konstanta "APPLICATION" ke dalam skrip.
6. **\$conflict1 = new Conflict();** Membuat objek baru dari kelas "Conflict" yang sudah diimpor menggunakan **use**. Objek ini disimpan dalam variabel **\$conflict1**.
7. **\$conflict2 = new Data\Two\Conflict();** Membuat objek baru dari kelas "Conflict" yang berada dalam namespace "Data\Two".
8. **helpMe();** Memanggil fungsi "helpMe()" yang sudah diimpor menggunakan **use**. Fungsi ini mungkin melakukan tugas tertentu.
9. **echo APPLICATION . PHP\_EOL;** Mencetak nilai konstanta "APPLICATION" ke layar, diikuti dengan karakter baris baru.

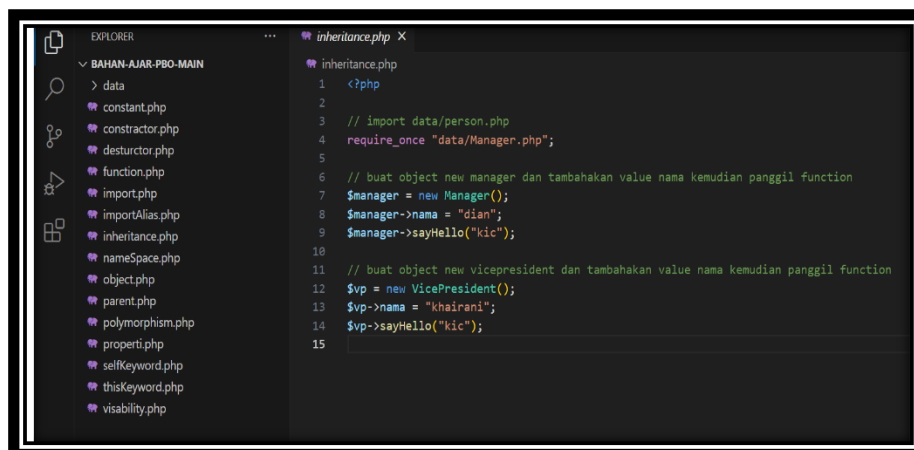
### 6. File importAlias.php



## Penjelasan :

1. **require\_once "data/Conflict.php";** Mengimpor (include) file "Conflict.php" dari direktori "data".
2. **require\_once "data/Helper.php";** Mengimpor (include) file "Helper.php" dari direktori "data".
3. **use Data\One\Conflict as Conflict1;** Menggunakan namespace "Data\One" dan mengimpor kelas "Conflict" ke dalam skrip dengan alias "Conflict1".
4. **use Data\Two\Conflict as Conflict2;** Menggunakan namespace "Data\Two" dan mengimpor kelas "Conflict" ke dalam skrip dengan alias "Conflict2".
5. **use function Helper\helpMe as help;** Menggunakan namespace "Helper" dan mengimpor fungsi "helpMe" ke dalam skrip dengan alias "help".
6. **use const Helper\APPLICATION as APP;** Menggunakan namespace "Helper" dan mengimpor konstanta "APPLICATION" ke dalam skrip dengan alias "APP".
7. **\$conflict1 = new Conflict1();** Membuat objek baru dari kelas "Conflict" yang sudah diimpor menggunakan **use** dengan alias "Conflict1".
8. **\$conflict2 = new Conflict2();** Membuat objek baru dari kelas "Conflict" yang sudah diimpor menggunakan **use** dengan alias "Conflict2".
9. **help();** Memanggil fungsi "help" yang sudah diimpor menggunakan **use**.
10. **echo APP . PHP\_EOL;** Mencetak nilai konstanta "APPLICATION" yang sudah diimpor menggunakan **use** dengan alias "APP".

## 7. File inheritance.php



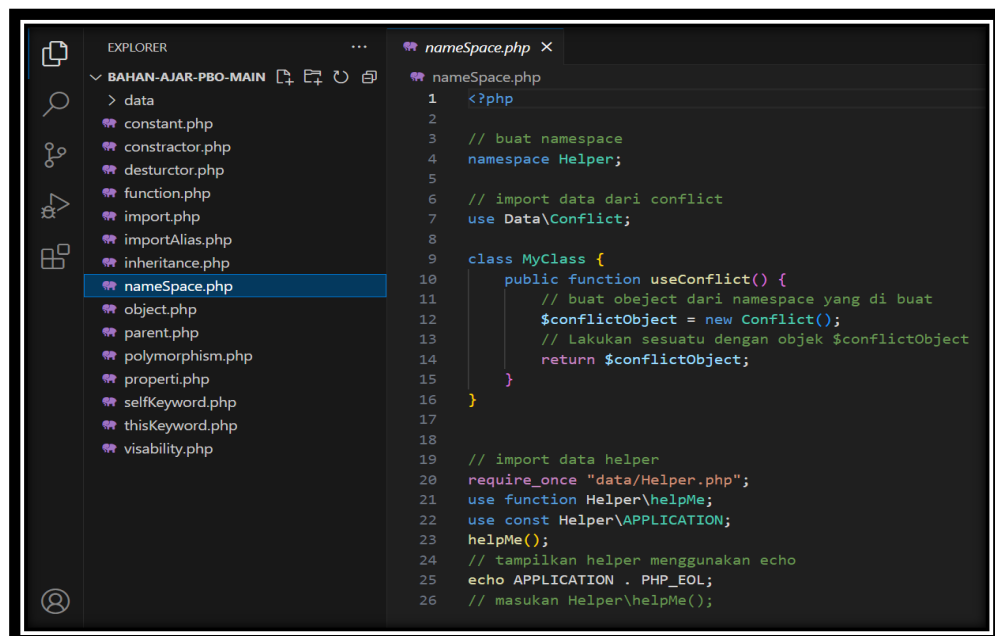
```
EXPLORER
  BAHAN-AJAR-PBO-MAIN
    > data
      constant.php
      constructor.php
      destructor.php
      function.php
      import.php
      importAlias.php
      inheritance.php
      namespace.php
      object.php
      parent.php
      polymorphism.php
      properti.php
      selfKeyword.php
      thisKeyword.php
      visibility.php

  inheritance.php X
  inheritance.php
  1  <?php
  2
  3  // import data/person.php
  4  require_once "data/Manager.php";
  5
  6  // buat object new manager dan tambahkan value nama kemudian panggil function
  7  $manager = new Manager();
  8  $manager->nama = "dian";
  9  $manager->sayHello("kic");
  10
  11 // buat object new Vicepresident dan tambahkan value nama kemudian panggil function
  12 $svp = new VicePresident();
  13 $svp->nama = "khairani";
  14 $svp->sayHello("kic");
  15
```

## Penjelasan :

1. **require\_once "data/Manager.php";** Mengimpor (include) file "Manager.php" yang berisi definisi kelas "Manager" dan "VicePresident".
2. **\$manager = new Manager();** Membuat objek baru dari kelas "Manager" dan menyimpannya dalam variabel **\$manager**.
3. **\$manager->nama = "dian";** Menetapkan nilai properti "nama" dari objek "\$manager" menjadi "dian".
4. **\$manager->sayHello("kic");** Memanggil metode "sayHello" dari objek "\$manager" dengan memberikan argumen "kic".
5. **\$vp = new VicePresident();** Membuat objek baru dari kelas "VicePresident" dan menyimpannya dalam variabel "\$vp".
6. **\$vp->nama = "khairani";** Menetapkan nilai properti "nama" dari objek "\$vp" menjadi "pita".
7. **\$vp->sayHello("kic");** Memanggil metode "sayHello" dari objek "\$vp" dengan memberikan argumen "kic".

## 8. File nameSpace.php



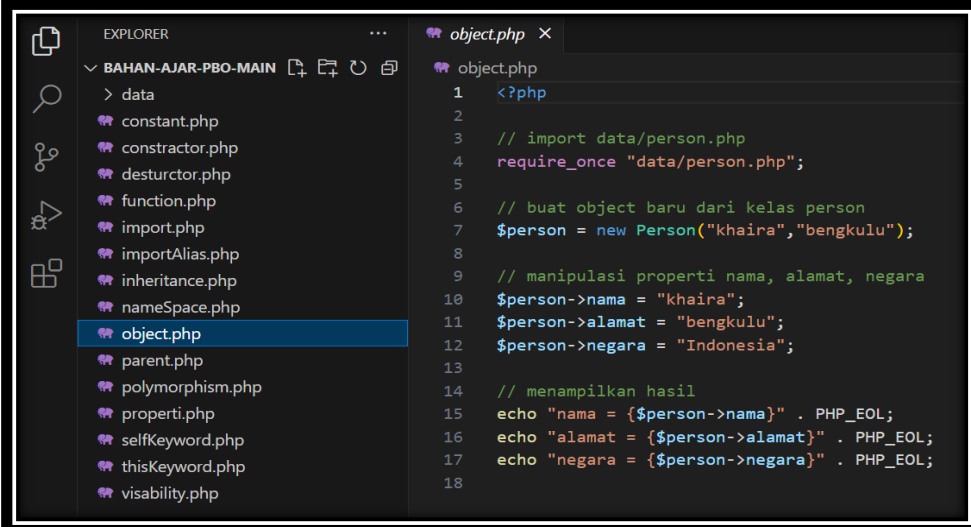
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with a file named `nameSpace.php` selected. The code editor shows the content of `nameSpace.php`, which includes a namespace declaration, a class definition, and a function call.

```
1 <?php
2
3 // buat namespace
4 namespace Helper;
5
6 // import data dari conflict
7 use Data\Conflict;
8
9 class MyClass {
10     public function useConflict() {
11         // buat obeejct dari namespace yang di buat
12         $conflictObject = new Conflict();
13         // Lakukan sesuatu dengan objek $conflictObject
14         return $conflictObject;
15     }
16 }
17
18
19 // import data helper
20 require_once "data/Helper.php";
21 use function Helper\helpMe;
22 use const Helper\APPLICATION;
23 helpMe();
24 // tampilkan helper menggunakan echo
25 echo APPLICATION . PHP_EOL;
26 // masukan Helper\helpMe();
```

## Penjelasan :

1. **namespace Helper;**: Mendefinisikan namespace dengan nama "Helper".
2. **use Data\Conflict;**: Menggunakan namespace "Data" dan mengimpor kelas "Conflict" ke dalam namespace "Helper".
3. **class MyClass { ... };**: Mendefinisikan kelas "MyClass" di dalam namespace "Helper".
4. **require\_once "data/Helper.php";**: Mengimpor (include) file "Helper.php" yang mungkin berisi definisi fungsi atau konstanta.
5. **use function Helper\helpMe;**: Menggunakan namespace "Helper" dan mengimpor fungsi "helpMe" ke dalam skrip.
6. **use const Helper\APPLICATION;**: Menggunakan namespace "Helper" dan mengimpor konstanta "APPLICATION" ke dalam skrip.
7. **helpMe();**: Memanggil fungsi "helpMe()" yang sudah diimpor. Fungsi ini mungkin melakukan tugas tertentu.
8. **echo APPLICATION . PHP\_EOL;**: Mencetak nilai konstanta "APPLICATION" ke layar, diikuti dengan karakter baris baru (**PHP\_EOL**).
9. **Helper\helpMe();**: Pernyataan ini seharusnya tidak diperlukan karena fungsi "helpMe()" sudah diimpor sebelumnya dengan pernyataan **use function**.

## 9. File object.php



The screenshot shows a code editor with a dark theme. On the left, the 'EXPLORER' sidebar displays a file tree for 'BAHAN-AJAR-PBO-MAIN'. The 'data' folder is expanded, showing several PHP files. 'object.php' is selected and highlighted. The main editor area displays the code for 'object.php'.

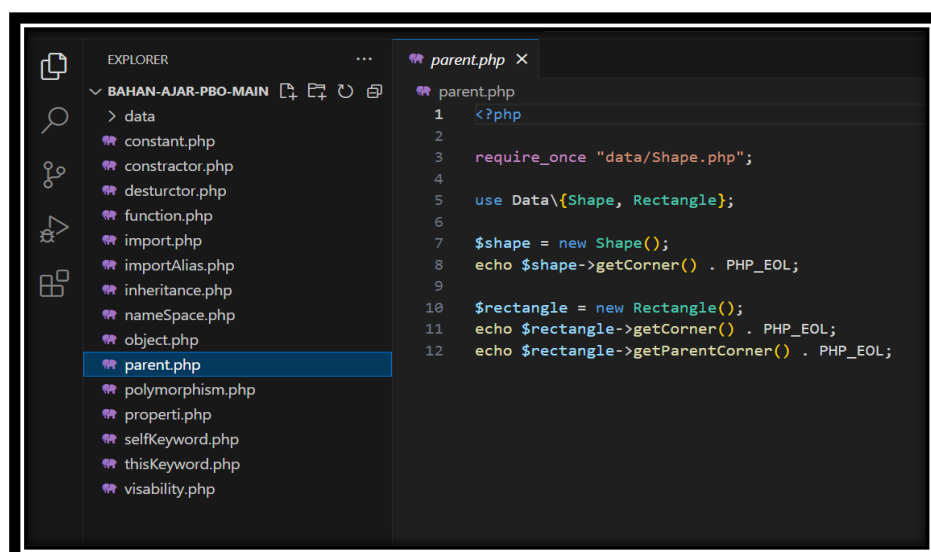
```
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object baru dari kelas person
7  $person = new Person("khaira","bengkulu");
8
9  // manipulasi properti nama, alamat, negara
10 $person->nama = "khaira";
11 $person->alamat = "bengkulu";
12 $person->negara = "Indonesia";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



### Penjelasan :

1. **require\_once "data/person.php";** Mengimpor (include) file "person.php" yang berisi definisi kelas "Person".
2. **\$person = new Person("khaira","bengkulu");** Membuat objek baru dari kelas "Person" dengan nama "\$person" dan memberikan dua argumen, yaitu "khaira" sebagainama dan "bengkulu" sebagai alamat, kepada konstruktor kelas "Person".
3. **\$person->nama = "khaira";** Mengubah nilai properti "nama" dari objek "\$person" menjadi "khaira".
4. **\$person->alamat = "bengkulu";** Mengubah nilai properti "alamat" dari objek "\$person" menjadi "bengkulu".
5. **\$person->negara = "Indonesia";** Mengubah nilai properti "negara" dari objek "\$person" menjadi "Indonesia".
6. **echo "nama = {\$person->nama}" . PHP\_EOL;** Menampilkan nilai properti "nama" dari objek "\$person" ke layar, diikuti dengan karakter baris baru.
7. **echo "alamat = {\$person->alamat}" . PHP\_EOL;** Menampilkan nilai properti "alamat" dari objek "\$person" ke layar, diikuti dengan karakter baris baru.
8. **echo "negara = {\$person->negara}" . PHP\_EOL;** Menampilkan nilai properti "negara" dari objek "\$person" ke layar, diikuti dengan karakter baris baru.

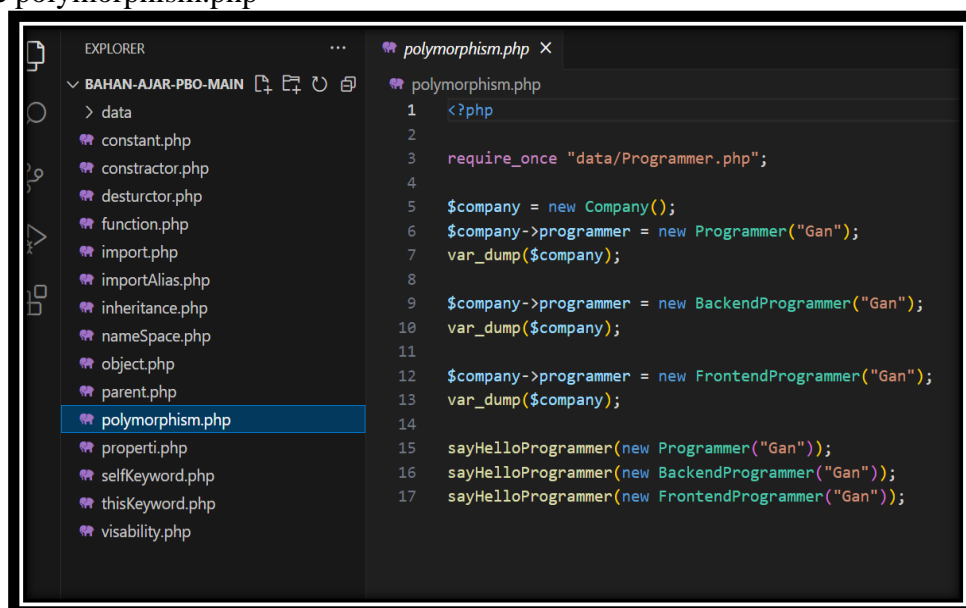
### 10. File parent.php



### Penjelasan :

1. **require\_once "data/Shape.php";** Mengimpor (include) file "Shape.php" yang berisi definisi kelas "Shape" dan "Rectangle".
2. **use Data\{Shape, Rectangle};** Menggunakan pernyataan **use** untuk mengimpor namespace "Data" dan kelas "Shape" serta "Rectangle" ke dalam skrip.
3. **\$shape = new Shape();** Membuat objek baru dari kelas "Shape" dan menyimpannya dalam variabel "\$shape".
4. **echo \$shape->getCorner() . PHP\_EOL;** Memanggil metode "getCorner()" dari objek "\$shape" dan menampilkan hasilnya ke layar, diikuti dengan karakter baris baru (PHP\_EOL).
5. **\$rectangle = new Rectangle();** Membuat objek baru dari kelas "Rectangle" dan menyimpannya dalam variabel "\$rectangle".
6. **echo \$rectangle->getCorner() . PHP\_EOL;** Memanggil metode "getCorner()" dari objek "\$rectangle" (kelas "Rectangle") dan menampilkan hasilnya ke layar, diikuti dengan karakter baris baru.
7. **echo \$rectangle->getParentCorner() . PHP\_EOL;** Memanggil metode "getParentCorner()" dari objek "\$rectangle" (kelas "Rectangle").

### 11. File polymorphism.php



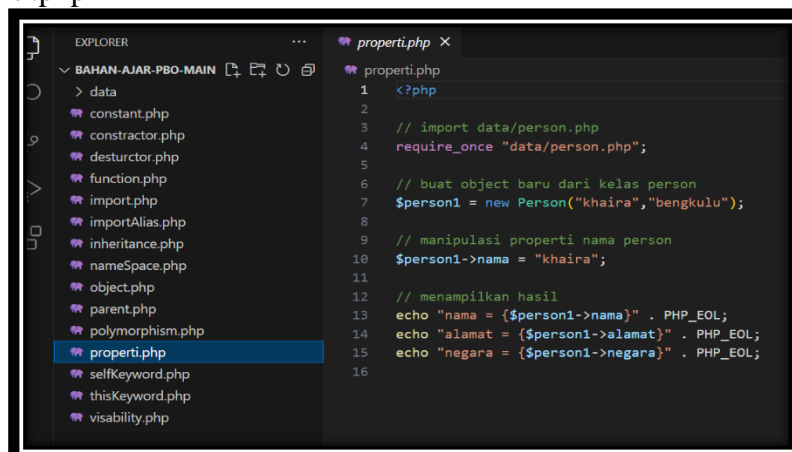
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with a file named `polymorphism.php` selected. The code editor shows the following PHP code:

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

### Penjelasan :

1. **require\_once "data/Programmer.php";** Mengimpor (include) file "Programmer.php" yang berisi definisi kelas "Programmer", "BackendProgrammer", "FrontendProgrammer", dan fungsi "sayHelloProgrammer".
2. **\$company = new Company();** Membuat objek baru dari kelas "Company" dan menyimpannya dalam variabel "\$company".
3. **\$company->programmer = new Programmer("Gan");** Mengisi properti "programmer" dari objek "\$company" dengan objek baru dari kelas "Programmer" yang memiliki nama "Gan".
4. **\$company->programmer = new BackendProgrammer("Gan");** Mengisi properti "programmer" dari objek "\$company" dengan objek baru dari kelas "BackendProgrammer" yang memiliki nama "Gan".
5. **\$company->programmer = new FrontendProgrammer("Gan");** Mengisi properti "programmer" dari objek "\$company" dengan objek baru dari kelas "FrontendProgrammer" yang memiliki nama "Gan".
6. **sayHelloProgrammer(new Programmer("Gan"));** Memanggil fungsi "sayHelloProgrammer" dengan memberikan objek baru dari kelas "Programmer" yang memiliki nama "Gan".
7. **sayHelloProgrammer(new BackendProgrammer("Gan"));** Memanggil fungsi "sayHelloProgrammer" dengan memberikan objek baru dari kelas "BackendProgrammer" yang memiliki nama "Gan".

### 12. File properti.php



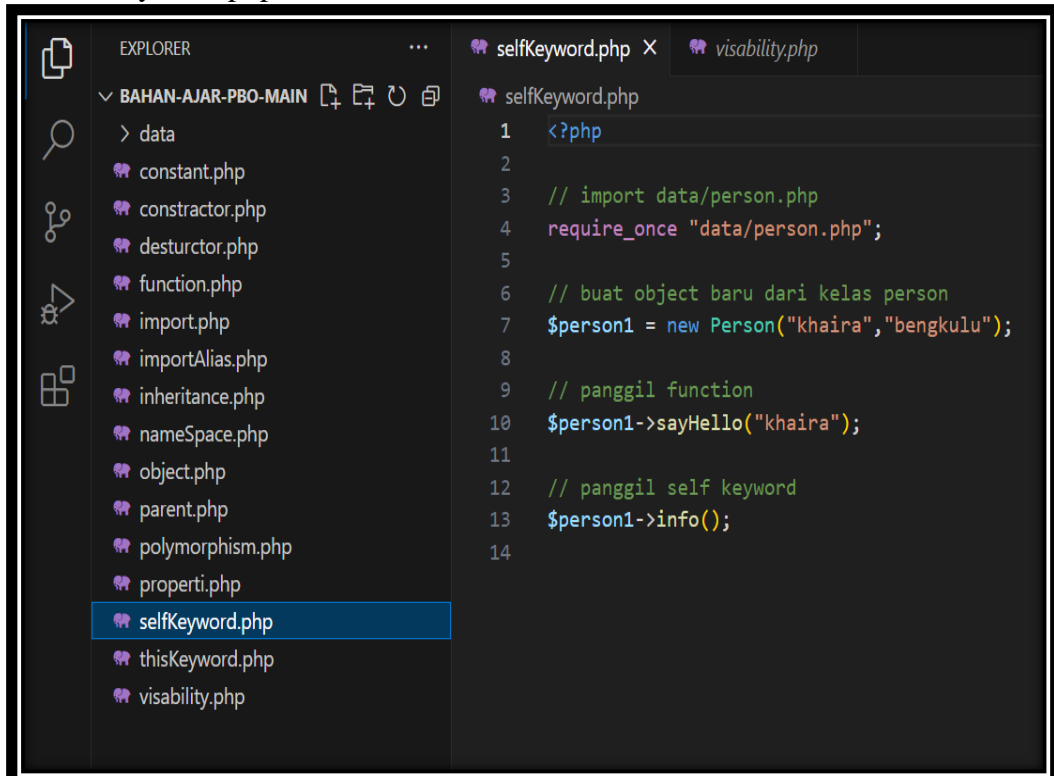
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with a file named `properti.php` selected. The code editor shows the content of `properti.php`, which is a PHP script that imports `data/person.php`, creates a new `Person` object, manipulates its properties, and displays the results.

```
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("khaira","bengkulu");
8
9 // manipulasi properti nama person
10 $person1->nama = "khaira";
11
12 // menampilkan hasil
13 echo "nama = {$person1->nama}" . PHP_EOL;
14 echo "alamat = {$person1->alamat}" . PHP_EOL;
15 echo "negara = {$person1->negara}" . PHP_EOL;
16
```

### Penjelasan :

1. **require\_once "data/person.php";** Mengimpor (include) file "person.php" yang berisi definisi kelas "Person".
2. **\$person1 = new Person("khaira","bengkulu");** Membuat objek baru dari kelas "Person" dengan nama "\$person1" dan memberikan dua argumen, yaitu "khaira" sebagai nama dan "bengkulu" sebagai alamat, kepada konstruktor kelas "Person".
3. **\$person1->nama = "khaira";** Memanipulasi (mengganti) nilai properti "nama" dari objek "\$person1" menjadi "khaira".
4. **echo "nama = {\$person1->nama}" . PHP\_EOL;** Menampilkan nilai properti "nama" dari objek "\$person1" ke layar, diikuti dengan karakter baris baru (PHP\_EOL).
5. **echo "alamat = {\$person1->alamat}" . PHP\_EOL;** Menampilkan nilai properti "alamat" dari objek "\$person1" ke layar, diikuti dengan karakter baris baru.
6. **echo "negara = {\$person1->negara}" . PHP\_EOL;** Menampilkan nilai properti "negara" dari objek "\$person1" ke layar, diikuti dengan karakter baris baru.

### 13. File selfKeyword.php



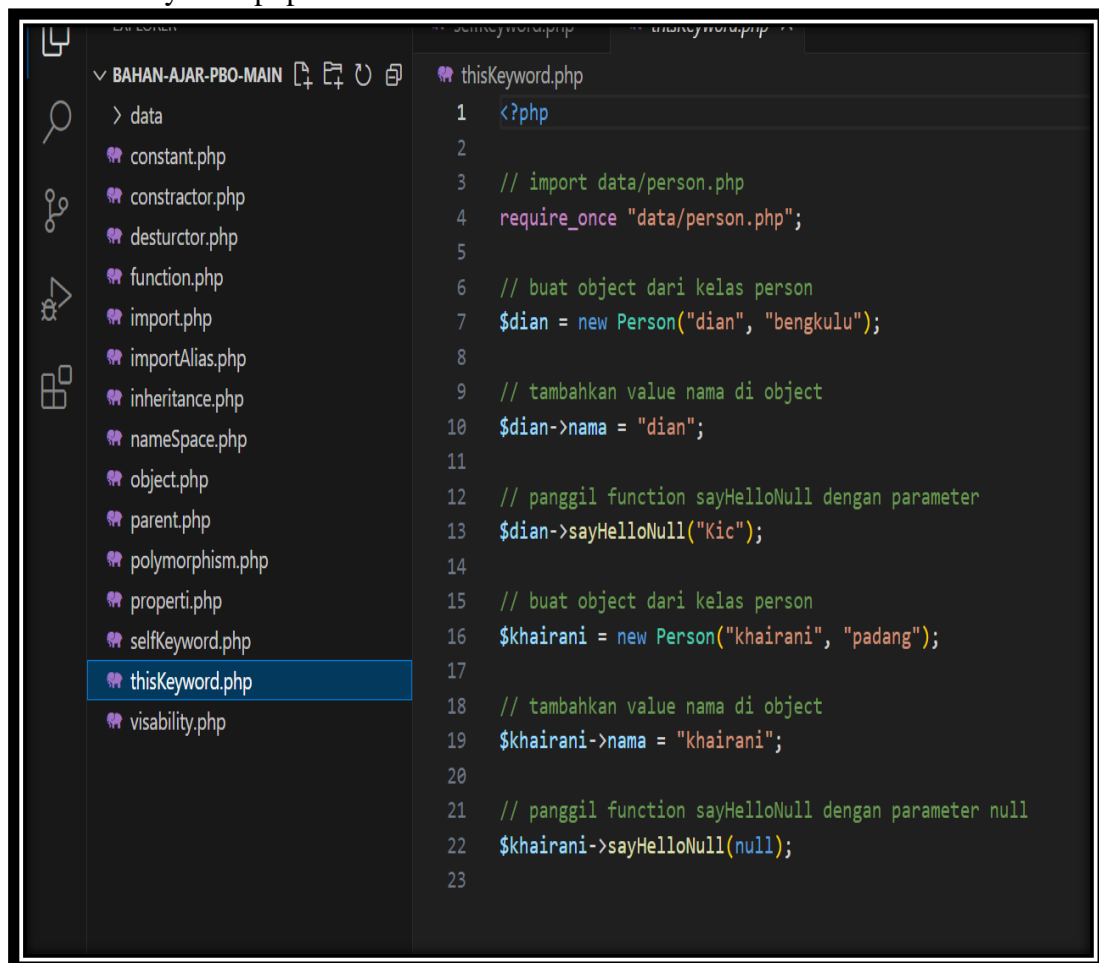
The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a directory structure with a folder named "BAHAN-AJAR-PBO-MAIN" containing a subfolder "data" and several PHP files. The file "selfKeyword.php" is selected. The code editor shows the content of "selfKeyword.php", which includes a PHP opening tag, a comment, and code to import "data/person.php", create a new "Person" object, call a "sayHello" method, and call an "info" method using the "self" keyword.

```
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("khaira","bengkulu");
8
9 // panggil function
10 $person1->sayHello("khaira");
11
12 // panggil self keyword
13 $person1->info();
14
```

### Penjelasan :

1. **require\_once "data/person.php";**: Mengimpor (include) file "person.php" yang berisi definisi kelas "Person".
2. **\$person1 = new Person("khaira","bengkulu");**: Membuat objek baru dari kelas "Person" dengan nama "\$person1" dan memberikan dua argumen, yaitu "khairani" sebagaimana dan "bengkulu" sebagai alamat, kepada konstruktor kelas "Person".
3. **\$person1->sayHello("khaira");**: Memanggil metode "sayHello" dari objek "\$person1" dengan memberikan argumen "khaira".
4. **\$person1->info();**: Memanggil metode "info" dari objek "\$person1". Metode ini mungkin menampilkan informasi tambahan yang terkait dengan objek "Person", dan jika metode "info" menggunakan **self**.

### 14. File thisKeyword.php

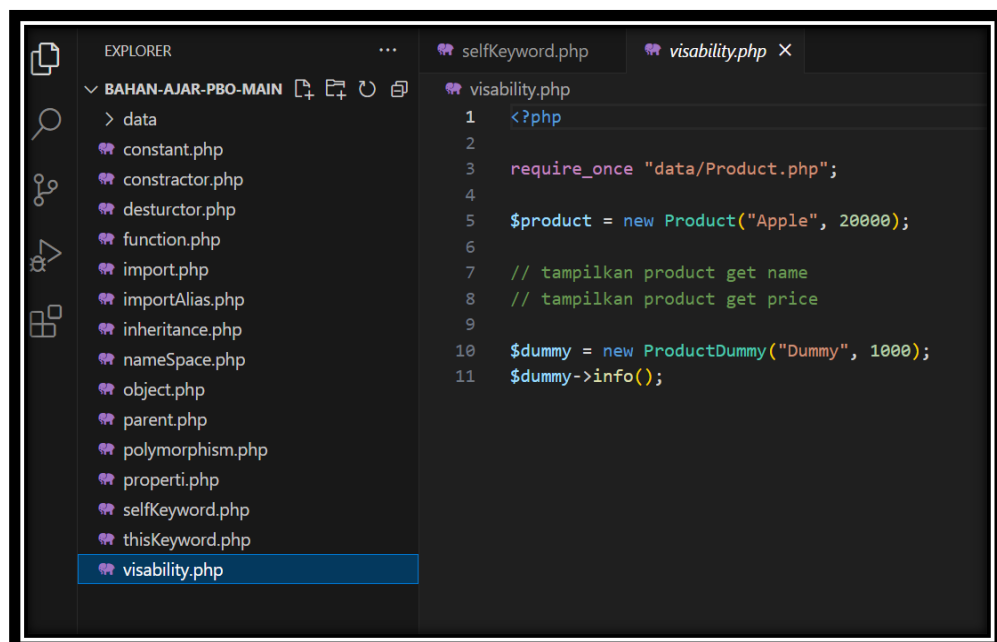


```
1  <?php
2
3  // import data/person.php
4  require_once "data/person.php";
5
6  // buat object dari kelas person
7  $dian = new Person("dian", "bengkulu");
8
9  // tambahkan value nama di object
10 $dian->nama = "dian";
11
12 // panggil function sayHelloNull dengan parameter
13 $dian->sayHelloNull("Kic");
14
15 // buat object dari kelas person
16 $khairani = new Person("khairani", "padang");
17
18 // tambahkan value nama di object
19 $khairani->nama = "khairani";
20
21 // panggil function sayHelloNull dengan parameter null
22 $khairani->sayHelloNull(null);
23
```

### Penjelasan :

1. **require\_once "data/person.php";** Mengimpor (include) file "person.php" yang berisi definisi kelas "Person".
2. **\$dian = new Person("dian", "bengkulu");** Membuat objek baru dari kelas "Person" dengan nama "\$dian" dan memberikan dua argumen, yaitu "dian" sebagai nama dan "bengkulu" sebagai alamat, kepada konstruktor kelas "Person".
3. **\$dian->nama = "dian";** Menetapkan nilai properti "nama" dari objek "\$dian" menjadi "dian".
4. **\$dian->sayHelloNull("Kic");** Memanggil metode "sayHelloNull" dari objek "\$dian" dengan memberikan argumen "Kic".
5. **\$khairani = new Person("khairani", "padang");** Membuat objek baru dari kelas "Person" dengan nama "\$khairani" dan memberikan dua argumen, yaitu "khairani" sebagai nama dan "padang" sebagai alamat, kepada konstruktor kelas "Person".
6. **\$khairani->nama = "khairani";** Menetapkan nilai properti "nama" dari objek "\$khairani" menjadi "khairani".
7. **\$khairani->sayHelloNull(null);** Memanggil metode "sayHelloNull" dari objek "\$khairani" dengan memberikan argumen null.

### 15. File visibility.php



### Penjelasan :

1. **require\_once "data/Product.php";**: Mengimpor (include) file "Product.php" yang berisi definisi kelas "Product" dan "ProductDummy".
2. **\$product = new Product("Apple", 20000);**: Membuat objek baru dari kelas "Product" dengan nama "\$product" dan memberikan dua argumen, yaitu "Apple" sebagai nama produk dan 20000 sebagai harga produk, kepada konstruktor kelas "Product".
3. **// tampilkan product get name**: Komentar yang menunjukkan niat untuk menampilkan nilai dari metode "getName()" pada objek "\$product". Namun, pernyataan ini sebenarnya tidak ada dalam kode.
4. **// tampilkan product get price**: Komentar yang menunjukkan niat untuk menampilkan nilai dari metode "getPrice()" pada objek "\$product". Namun, pernyataan ini sebenarnya tidak ada dalam kode.
5. **\$dummy = new ProductDummy("Dummy", 1000);**: Membuat objek baru dari kelas "ProductDummy" dengan nama "\$dummy" dan memberikan dua argumen, yaitu "Dummy" sebagai nama produk dan 1000 sebagai harga produk, kepada konstruktor kelas "ProductDummy".
6. **\$dummy->info();**: Memanggil metode "info()" dari objek "\$dummy". Metode ini mungkin menampilkan informasi tambahan tentang objek "ProductDummy", seperti nama dan harga produk, ke layar.

**Link github :** [https://github.com/khaira04/G1F022004\\_Dian-Khairani\\_responsi-pbo.git](https://github.com/khaira04/G1F022004_Dian-Khairani_responsi-pbo.git)