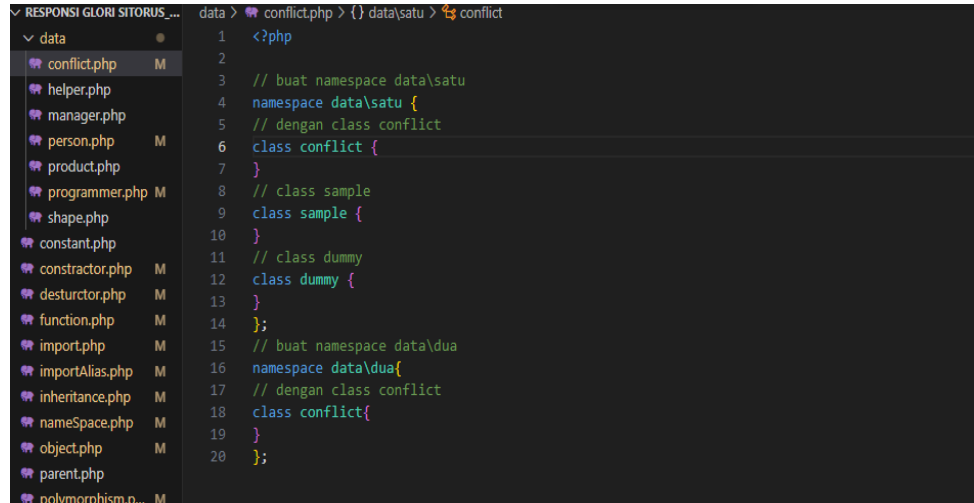


Nama : Glori Pesta Pince Sitorus
NPM : G1F022018
Responsi : Proyek Pemrograman Berorientasi Objek

1. Conflict.php

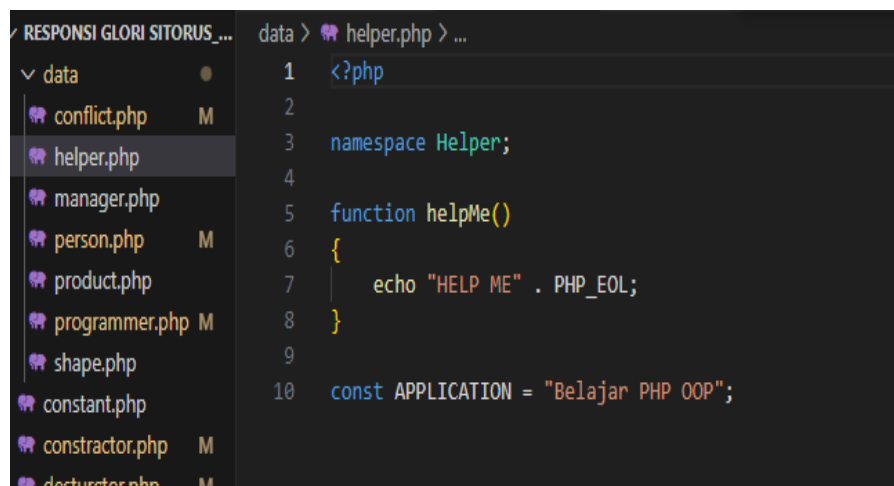


```
1 <?php
2
3 // buat namespace data\satu
4 namespace data\satu {
5 // dengan class conflict
6 class conflict {
7 }
8 // class sample
9 class sample {
10 }
11 // class dummy
12 class dummy {
13 }
14 };
15 // buat namespace data\dua
16 namespace data\dua{
17 // dengan class conflict
18 class conflict{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

Penjelasan:

Pada gambar di atas merupakan kode PHP yang menunjukkan bagaimana menggunakan namespace. Terdapat beberapa kelas yang dibuat di dalam beberapa namespace. Secara keseluruhan, kode ini menunjukkan cara mengatur struktur kode dengan menggunakan namespace dan mengatasi masalah duplikasi nama kelas. Kode ini juga menunjukkan cara mengirim data melalui method POST ke dalam file PHP.

2. Helper.php



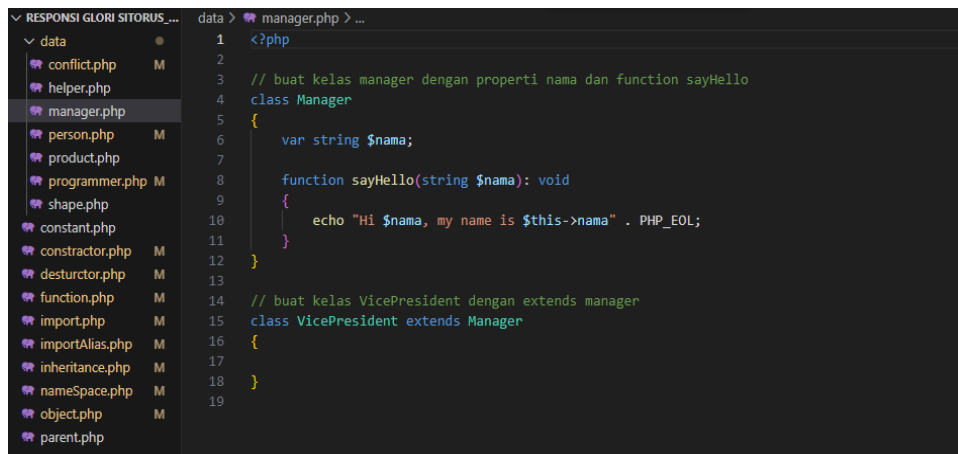
```
1 <?php
2
3 namespace Helper;
4
5 function helpMe()
6 {
7     echo "HELP ME" . PHP_EOL;
8 }
9
10 const APPLICATION = "Belajar PHP OOP";
```

Penjelasan:

- Definisikan konstanta APPLICATION yang bernilai “Belajar PHP OOP”.
- Definisikan fungsi helpMe() yang menggunakan echo untuk menampilkan teks “HELP ME”.

Dalam hal ini, konstanta APPLICATION digunakan untuk mendefinisikan nama aplikasi, sementara fungsi bantuan helpMe() digunakan untuk memberikan informasi tambahan. Setelah kodenya di jalankan, maka hasilnya akan menampilkan “HELP ME”.

3. Manager.php

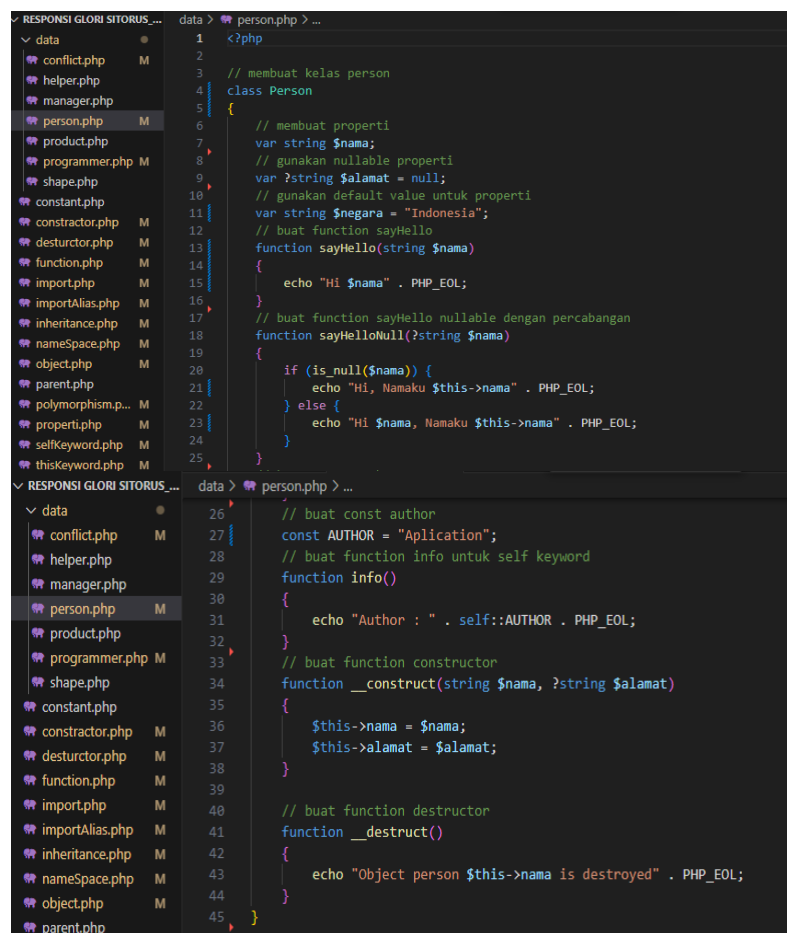


```
1 <?php
2
3 // buat kelas manager dengan properti nama dan function sayHello
4 class Manager
5 {
6     var string $nama;
7
8     function sayHello(string $nama): void
9     {
10         echo "Hi $nama, my name is $this->nama" . PHP_EOL;
11     }
12 }
13
14 // buat kelas VicePresident dengan extends manager
15 class VicePresident extends Manager
16 {
17 }
18 }
19
```

Penjelasan:

Pada gambar di atas merupakan kode dari kelas manager.php yang memiliki property \$nama dengan tipe data string. Di dalam kelas Manager, ada function sayHello yang menerima parameter berupa string \$nama. Function sayHello akan menampilkan kalimat "Hi nama, my name is this->nama" di layar, dimana nama adalah parameter yang diterima dan this->nama adalah nama Manager. Buat kelas VicePresident yang melakukan extends pada kelas Manager. Hal ini menunjukkan bahwa kelas VicePresident merupakan turunan dari kelas Manager dan memiliki fitur yang sama dengan kelas Manager

4. Person.php



```
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 = "Aplication";
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 }
```

Penjelasan:

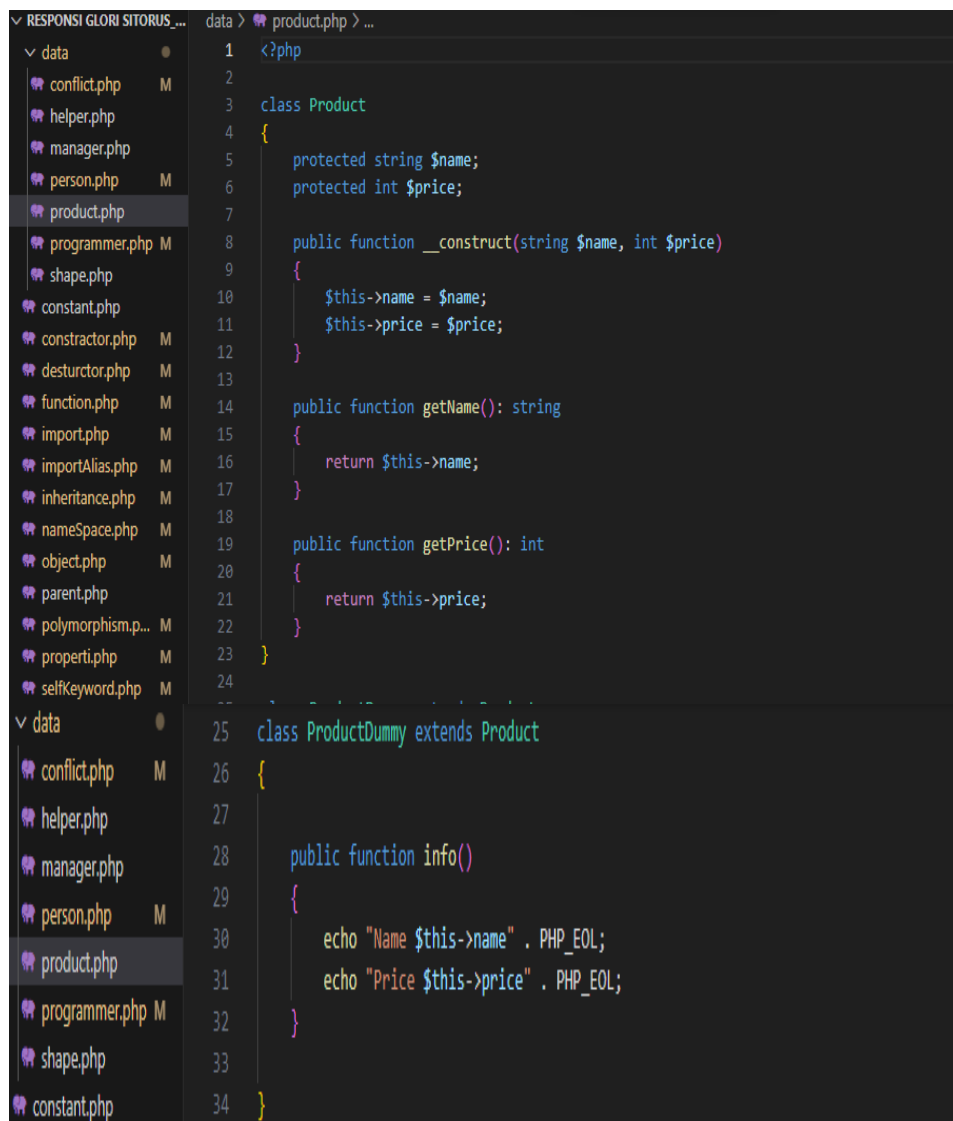
Gambar di atas merupakan kode dari kelas person. Kelas person memiliki beberapa

property, yaitu \$nama, \$alamat, dan \$negara. Kelas person memiliki beberapa fungsi, yaitu:

- sayHello, menerima parameter berupa nama dan menampilkan pesan "Hello\$nama".
- sayHelloNull, menerima parameter berupa nama yang mungkin kosong (null). Jika nama kosong, maka fungsi akan menampilkan pesan "Hi, my nama is this->nama". Jika nama ada, maka fungsi akan menampilkan pesan "Hi nama, my nama is \$this->nama".
- Info, menggunakan kata kunci self untuk mengakses nilai konstan author dan menampilkannya di layar.

Konstanta author didefinisikan untuk menyimpan nama author kelas, dan fungsi konstruktor_construct dibuat untuk menginisialisasi objek kelas person dengan nilai property \$nama dan \$alamat.. fungsi destructor dibuat untuk menjalankan kode saat objek kelas person dihancurkan. Fungsi konstruktor dan destructor ini dianggap opsional namun berguna untuk mengatur pembuatan dan penghancuran objek.

5. Produk.php



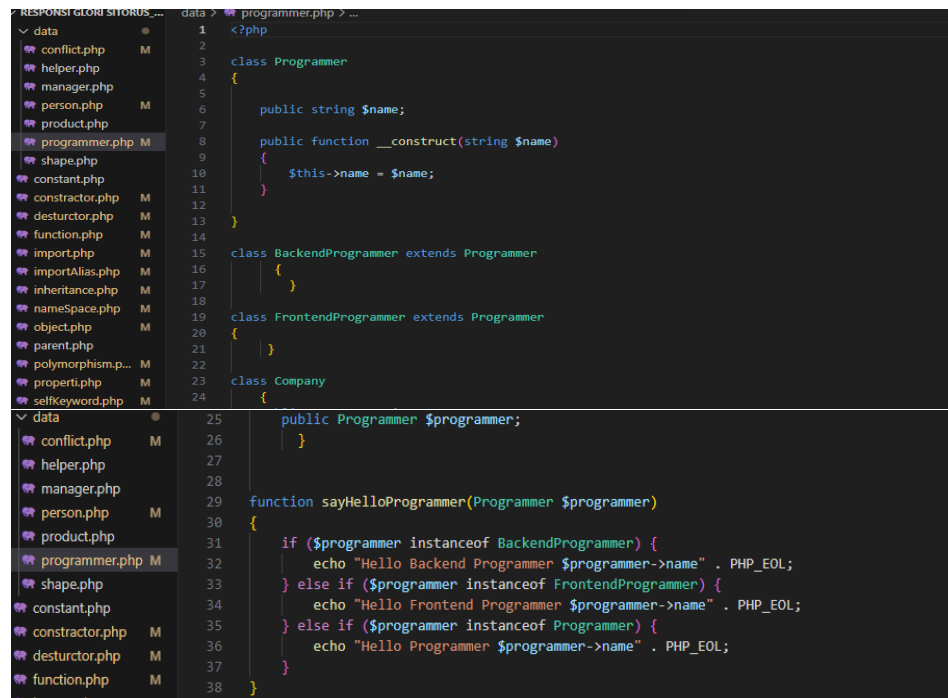
```
1 <?php
2
3 class Product
4 {
5     protected string $name;
6     protected int $price;
7
8     public function __construct(string $name, int $price)
9     {
10         $this->name = $name;
11         $this->price = $price;
12     }
13
14     public function getName(): string
15     {
16         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 ini menggambarkan dua kelas yaitu Product dan ProductDummy. Product merupakan kelas dasar yang mengatur beberapa properti, seperti name dan price. Properti ini dilindungi oleh sistem aksesnya agar hanya bisa diakses oleh kelas itu sendiri dan turunannya. Selain itu, kelas ini

juga memiliki konstruktor dan dua fungsi getter (getName dan getPrice) yang digunakan untuk mengambil nilai dari properti tersebut.

6. Programmer.php

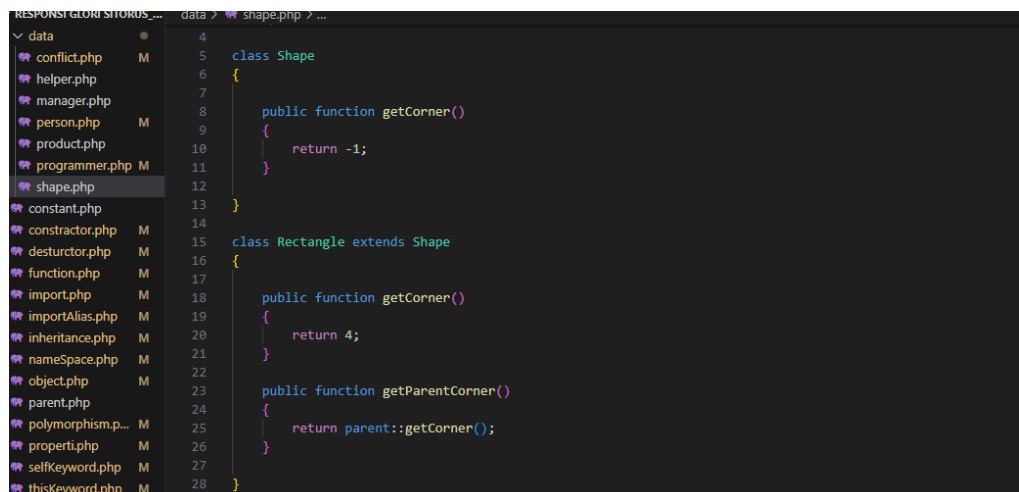


```
1 <?php
2
3 class Programmer
4 {
5     public string $name;
6
7     public function __construct(string $name)
8     {
9         $this->name = $name;
10    }
11
12    class BackendProgrammer extends Programmer
13    {
14    }
15
16    class FrontendProgrammer extends Programmer
17    {
18    }
19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

Penjelasan:

Pada gambar di atas merupakan kelas dari programmer.php. Programmer adalah kelas dasar yang memiliki properti \$name. Selain itu, kelas ini juga memiliki konstruktor yang mengambil satu parameter \$name dan menginisialisasi properti \$name dengan nilai yang diberikan. BackendProgrammer dan FrontendProgrammer adalah kelas yang mewarisi dari kelas Programmer. Artinya, kelas-kelas ini menggunakan properti dan fungsi yang telah ada di kelas Programmer. Company adalah kelas yang memiliki properti \$programmer yang berasal dari kelas Programmer. Dengan demikian, objek Company dapat menyimpan referensi ke objek programmer.sayHelloProgrammer adalah fungsi yang mengambil parameter berupa objek Programmer.

7. Shape.php

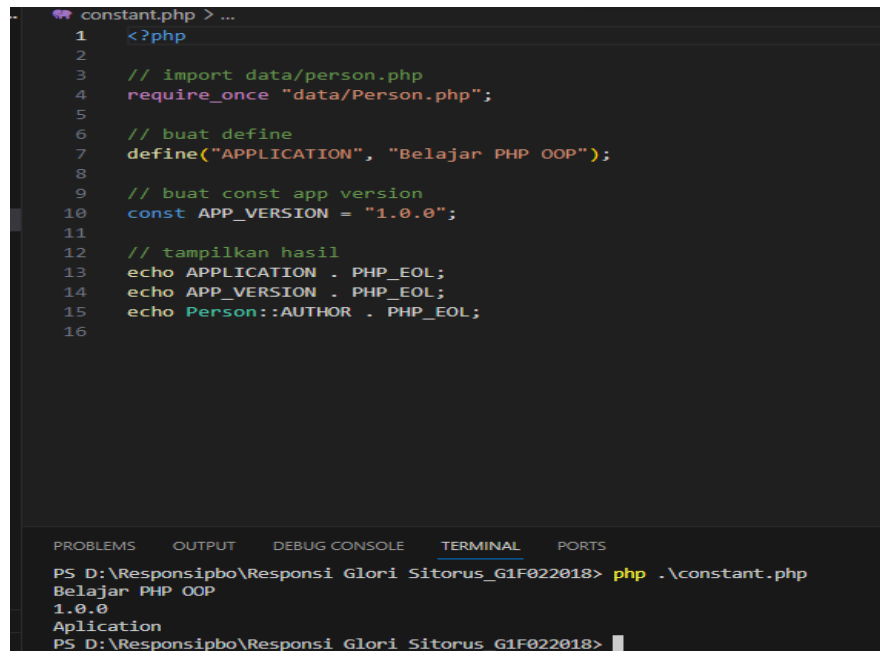


```
4 class Shape
5 {
6     public function getCorner()
7     {
8         return -1;
9     }
10 }
11
12 class Rectangle extends Shape
13 {
14     public function getCorner()
15     {
16         return 4;
17     }
18
19     public function getParentCorner()
20     {
21         return parent::getCorner();
22     }
23 }
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kelas shape dan rectangle. Kelas Shape merupakan kelas induk (parent) yang memiliki properti \$name, \$age, dan \$email. Selain itu, kelas ini juga memiliki metode getCorner() yang mengembalikan nilai -1. Kelas Rectangle merupakan kelas turunan (child) dari kelas Shape. Di dalam kelas ini, metode getCorner() telah di-override (ditimpa) sehingga mengembalikan nilai 4. Selain itu, kelas ini juga memiliki metode getParentCorner() yang akan memanggil metode getCorner() dari kelas induknya, yaitu kelas Shape.

8. Constant.php



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

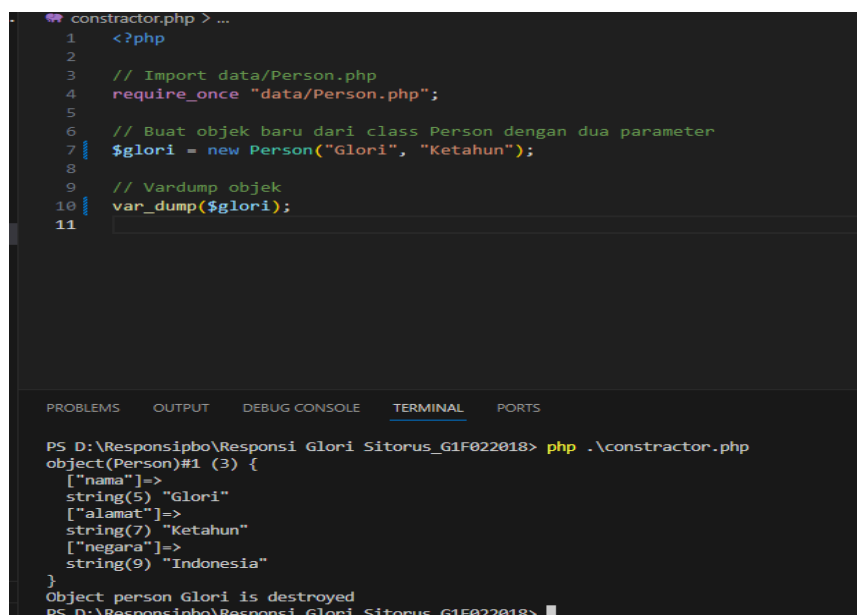
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\constant.php
Belajar PHP OOP
1.0.0
Application
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kelas constant. “data/person.php” Ini merupakan baris kode yang digunakan untuk mengimpor file define() adalah fungsi yang digunakan untuk menciptakan konstanta. Dalam hal ini, konstanta APPLICATION dibuat dengan nilai "Belajar PHP OOP"

9. Constractor.php



```
constructor.php > ...
1  <?php
2
3  // Import data/Person.php
4  require_once "data/Person.php";
5
6  // Buat objek baru dari class Person dengan dua parameter
7  $glori = new Person("Glori", "Ketahun");
8
9  // Vardump objek
10 var_dump($glori);
11
```

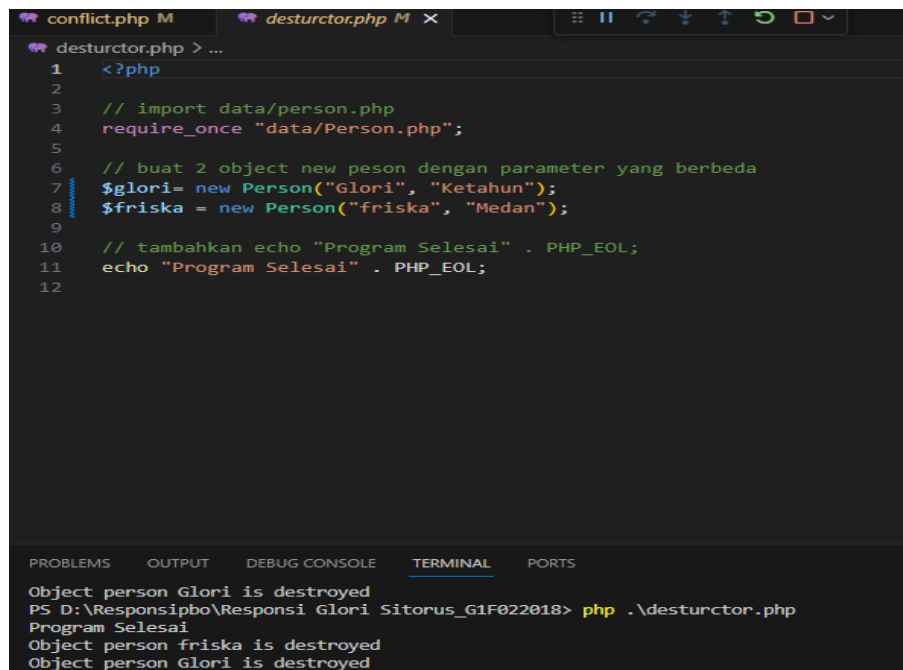
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\constructor.php
object(Person)#1 (3) {
  ["nama"]=>
  string(5) "Glori"
  ["alamat"]=>
  string(7) "Ketahun"
  ["negara"]=>
  string(9) "Indonesia"
}
Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan kelas dari constructor. Hal pertama yang dilakukan adalah mengimpor file person.php dari direktori data. Kemudian membuat objek baru dari kelas person dengan memberikan dua parameter yaitu "Glori" sebagai nama, dan "Ketahun" sebagai alamat. Kemudian kita menggunakan `var_dump($esra);` untuk menampilkan informasi lengkap tentang objek, termasuk tipe data, nilai properti, dan informasi lainnya.

10. Destructor.php



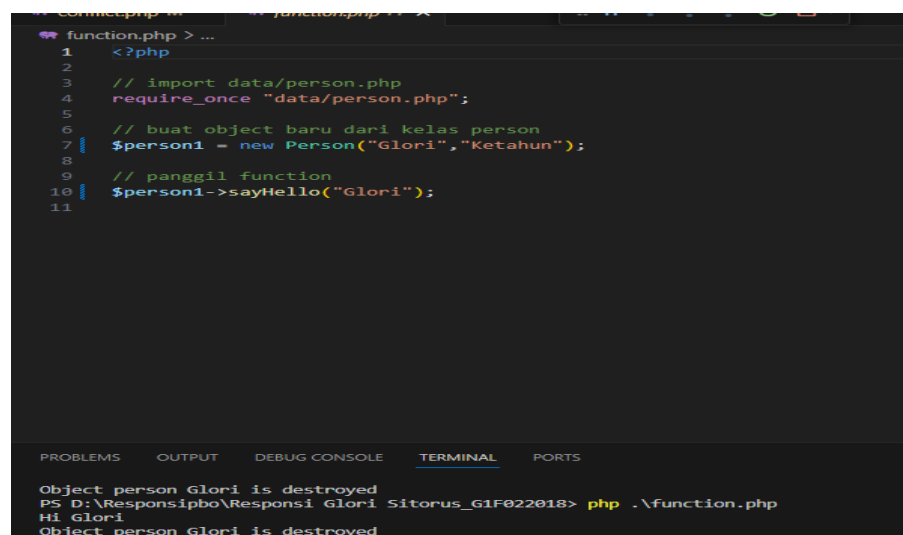
```
1 <?php
2
3 // import data/person.php
4 require_once "data/Person.php";
5
6 // buat 2 object new peson dengan parameter yang berbeda
7 $glori= new Person("Glori", "Ketahun");
8 $friska = new Person("friska", "Medan");
9
10 // tambahkan echo "Program Selesai" . PHP_EOL;
11 echo "Program Selesai" . PHP_EOL;
12
```

Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\desturctor.php
Program Selesai
Object person friska is destroyed
Object person Glori is destroyed

Penjelasan:

Pada kode diatas, langkah pertama yang dilakukan adalah mengimpor file Person.php yang berisi kelas Person. Kelas ini digunakan untuk membuat objek yang mewakili orang. Selanjutnya, kita membuat dua objek baru bernama \$seglori dan \$friska menggunakan kelas Person. Objek ini dibuat dengan dua parameter: "Glori" untuk nama dan "Ketahun" untuk tempat lahir. Perintah ini akan mencetak "Program Selesai" pada layar, dan "PHP_EOL" akan menambahkan karakter baris baru yang sesuai dengan sistem operasi yang digunakan.

11. Function.php



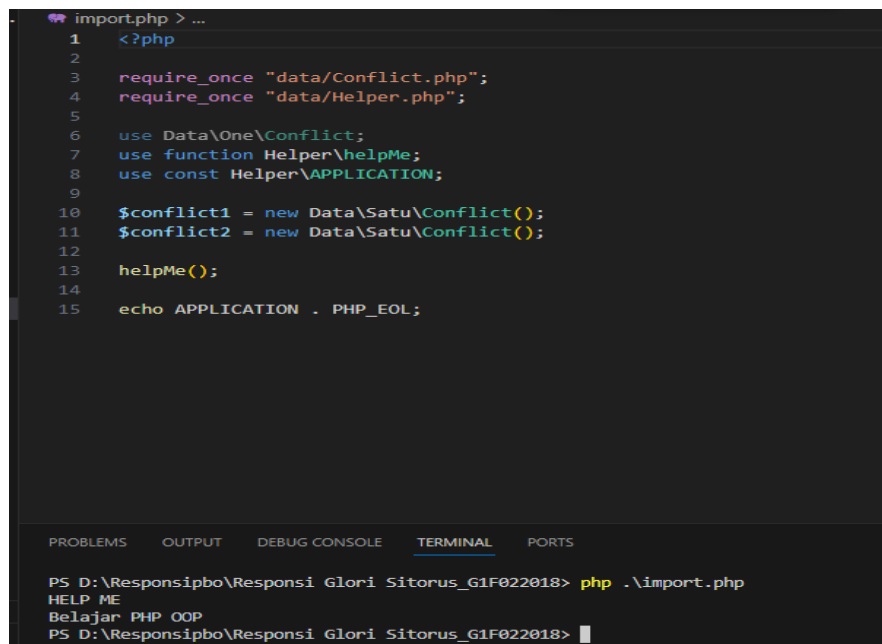
```
1 <?php
2
3 // import data/person.php
4 require_once "data/person.php";
5
6 // buat object baru dari kelas person
7 $person1 = new Person("Glori", "Ketahun");
8
9 // panggil function
10 $person1->sayHello("Glori");
11
```

Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\function.php
Hi Glori
Object person Glori is destroyed

Penjelasan:

Pada gambar di atas merupakan kode dari function.php. :Import data/person.php: Dalam baris pertama, kita menggunakan require_once untuk mengimport file data/person.php. Baris kedua dan ketiga membuat objek baru bernama \$person1. Objek ini dibuat dari kelas Person dan diberi parameter "Glori" dan "Ketahun". Panggil function: Baris keempat memanggil function sayHello yang ada di dalam kelas Person. Function ini akan mencetak "Hello, Esra!" pada layar

12. Import.php



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

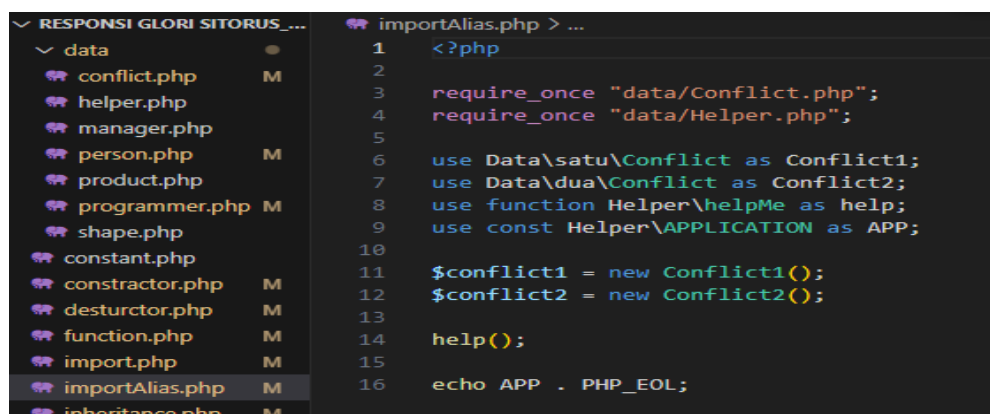
PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\import.php
HELP ME
Belajar PHP OOP
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode import.php. Perintah use dan use_once digunakan untuk mengimpor kelas, fungsi, dan konstanta dari file terkait. Kelas Conflict diimpor dari namespace Data\One menggunakan use_once, sehingga memudahkan pembuatan objek kelas ini di seluruh kode kita Selanjutnya, impor fungsi helpMe dan konstanta APPLICATION yang digunakan sehingga fungsi dan nilai konstanta ini tersedia tanpa menentukan jalur lengkap. Selanjutnya, kode tersebut membuat dua objek kelas conflict. Hal ini menunjukkan bahwa setelah mengimpor kelas, kita dapat membuat objeknya hanya dengan menggunakan nama kelas.

13. importAlias.php



```
RESPONSI GLORI SITURUS_...
└─ data
   ├── conflict.php M
   ├── helper.php
   ├── manager.php
   ├── person.php M
   ├── product.php
   ├── programmer.php M
   ├── shape.php
   ├── constant.php
   ├── constructor.php M
   ├── destructor.php M
   ├── function.php M
   ├── import.php M
   ├── importAlias.php M
   └── inheritance.php M
```

```
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:

Pada gambar di atas merupakan kode dari impotrAlias. Dalam kode yang disediakan,

Conflictkelas dari Data\satunamespace diimpor dan diberi alias Conflict1. Kelas Conflictdari Data\duanamespace juga diimpor dan diberi alias Conflict2. Hal ini memungkinkan kode untuk membuat instance setiap kelas tanpa konflik. Selain itu, helpMefungsi dari Helpnamespace diimpor dan diberi alias help. Konstanta APPLICATIONdari Helpnamespace juga diimpor dan diberi alias APP. Item yang diimpor ini kemudian dapat digunakan di seluruh kode lainnya. Terakhir, kode tersebut membuat instance kelas Conflict1and Conflict2, memanggil helpfungsi, dan menggemakan nilai konstanta APP. Ini menunjukkan bagaimana kelas dan fungsi yang diimpor dapat digunakan dalam kode.

14. Inheritance.php

```
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 = "Glori";
9  $manager->sayHello("Hallo bestiee");
10
11 // buat object new vicepresident dan tambahkan value nama kemudian panggil function
12 $vp = new VicePresident();
13 $vp->nama = "friska";
14 $vp->sayHello("halo mae???");
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\inheritance.php
Hi Hallo bestiee, my name is Glori
Hi halo mae???, my name is friska
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode inheritance.php. keseluruhan kode tersebut menciptakan objek dari kelas Manager dan VicePresident, memberikan nilai ke properti nama, dan memanggil metode sayHello dengan argumen tertentu pada masing-masing objek.

15. namespace.php

```
RESPONSI GLORI SITURUS_... namespace.php > ...
1  <?php
2
3  // buat namespace
4  // import data dari conflict
5  require "conflict.php";
6  // buat obeject 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();
```

data

- conflict.php M
- helper.php
- manager.php
- person.php M
- product.php
- programmer.php M
- shape.php
- constant.php
- constructor.php M
- destructor.php M
- function.php M
- import.php M
- importAlias.php M
- inheritance.php M
- nameSpace.php M
- object.php M
- parent.php
- polymorphism.p... M

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode namespace.php. 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. 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.

16. Object.php

```
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("Glori","Ketahun");
8
9  // manipulasi properti nama, alamat, negara
10 $person->nama = "Glori";
11 $person->alamat = "Ketahun";
12 $person->negara = "Korea barat";
13
14 // menampilkan hasil
15 echo "nama = {$person->nama}" . PHP_EOL;
16 echo "alamat = {$person->alamat}" . PHP_EOL;
17 echo "negara = {$person->negara}" . PHP_EOL;
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\object.php
nama = Glori
alamat = Ketahun
negara = Korea barat
Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan kode dari object.php. Kode PHP di atas mengimpor file eksternal "person.php" dan membuat objek baru dari kelas Person dengan memberikan nilai "Glori" sebagai nama dan "Ketahun" sebagai alamat melalui konstruktor. Pemanggilan pernyataan echo digunakan untuk menampilkan hasil manipulasi properti, mencetak nilai yang telah diubah untuk properti nama, alamat, dan negara ke layar.

17. Parent.php

```
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;
```

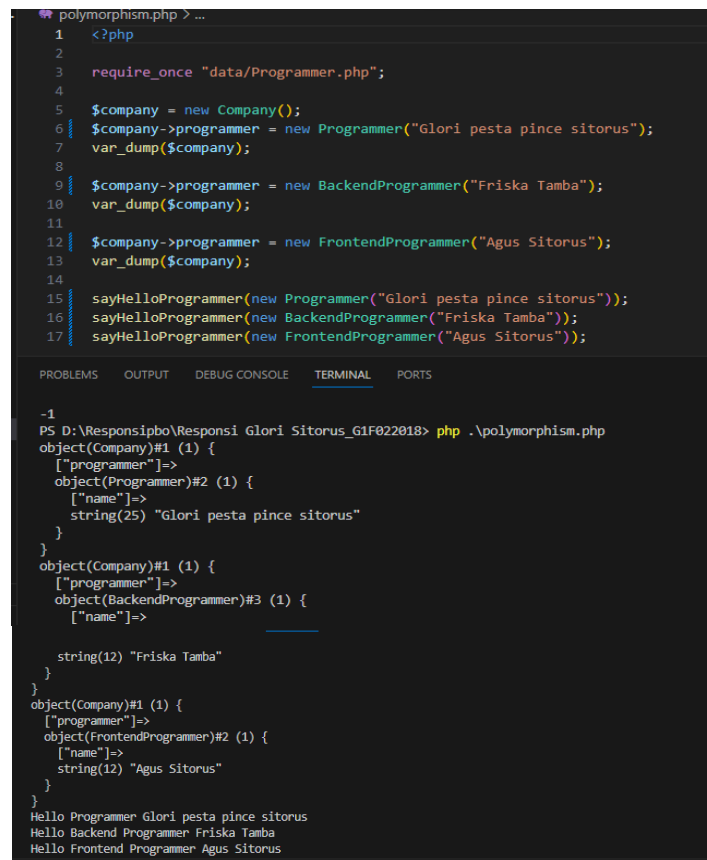
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
negara = Korea barat
Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\parent.php
-1
-1
-1
```

Penjelasan:

Pada gambar di atas menggunakan namespace, aliasing, dan membuat objek dari kelas Shape dan Rectangle. Selanjutnya, ia memanggil metode dari keduanya, menampilkan hasilnya ke dalam output. Membuat objek baru dari kelas Shape dan memanggil metode getCorner untuk menampilkan hasilnya. Membuat objek baru dari kelas Rectangle, memanggil metode getCorner untuk menampilkan hasilnya, dan memanggil metode getParentCorner dari kelas Shape (parentclass) yang diwarisi oleh kelas Rectangle.

18. Polymorphism.php



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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
-1
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\polymorphism.php
object(Company)#1 (1) {
    ["programmer"]=>
    object(Programmer)#2 (1) {
        ["name"]=>
        string(25) "Glori pesta pince sitorus"
    }
}
object(Company)#1 (1) {
    ["programmer"]=>
    object(BackendProgrammer)#3 (1) {
        ["name"]=>
        string(12) "Friska Tamba"
    }
}
object(Company)#1 (1) {
    ["programmer"]=>
    object(FrontendProgrammer)#2 (1) {
        ["name"]=>
        string(12) "Agus Sitorus"
    }
}
Hello Programmer Glori pesta pince sitorus
Hello Backend Programmer Friska Tamba
Hello Frontend Programmer Agus Sitorus
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode Polymorphism.php. Instansiasi kelas "Company" dan memberikan nilai objek dari kelas "Programmer". Lalu, var_dump untuk mengecek isi objek tersebut. Melakukan override objek "programmer" dengan objek dari kelas "BackendProgrammer". Lalu, var_dump untuk mengecek isi objek tersebut. Melakukan override objek "programmer" kembali dengan objek dari kelas "FrontendProgrammer". Lalu, var_dump untuk mengecek isi objek tersebut. Menggunakan konsep polimorfisme untuk menjalankan method "sayHelloProgrammer" pada berbagai jenis objek programmer. Pada setiap kelas "Programmer", "BackendProgrammer", dan "FrontendProgrammer", ada method bernama "sayHello". Saat menjalankan method "sayHelloProgrammer" pada objek, dia akan mencari method "sayHello" pada objek tersebut dan menjalankannya.

19. Property.php

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

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
Hello Frontend Programmer Agus Sitorus
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\property.php
nama = Glori
alamat = Ketahun
negara = Korea barat
Object person Glori is destroyed
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode property.php. kode ini membuat objek dari kelas Person, mengubah nilai dari beberapa propertinya, dan menampilkan nilai-nilai tersebut ke dalam output. Membuat objek baru dari kelas Person dengan memberikan dua parameter, yaitu "Glori" dan "Ketahun" pada saat inisialisasi objek. Mengubah nilai dari properti nama, alamat, dan negara pada objek \$person1. Mencetak nilai properti nama, alamat, dan negara dari objek \$person1 ke dalam output.

20. Visibility.php

```
visibility.php M 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();
9  // tampilkan product get price
10 echo $product->getPrice();
11
12 $dummy = new ProductDummy("Dummy", 1000);
13 $dummy->info();
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS

```
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018> php .\visibility.php
Apple20000Name Dummy
Price 1000
PS D:\Responsipbo\Responsi Glori Sitorus_G1F022018>
```

Penjelasan:

Pada gambar di atas merupakan tampilan dari kode Visibility.php. kode ini membuat objek dari kelas Product dan ProductDummy, menampilkan nilai dari beberapa metode, dan memanggil metode info dari objek ProductDummy. Membuat objek baru dari kelas Product dengan memberikan dua parameter, yaitu "Apple" dan 20000 pada saat inisialisasi objek. Mencetak nilai dari metode getName dan getPrice dari objek \$product ke dalam output. Membuat objek baru dari

kelas ProductDummy dengan memberikan dua parameter, yaitu "Dummy" dan 1000 pada saat inisialisasi objek. Selanjutnya, memanggil metode info dari objek \$dummy.