

Nama : Salsabila Refiani Safitri

NPM : G1F022030

Kelas : B

Matkul : Proyek Pemrograman Berorientasi Objek

## RESPONSI PEMROGRAMAN BERORIENTASI OBJEK

a. Data

1. Conflict

```
data > conflict.php
1  <?php
2
3  // buat namespace data\satu
4  namespace data\satu{
5      // dengan class conflict
6      class conflict
7      {
8      }
9      // class sample
10     class sample
11     {
12     }
13     // class dummy
14     class dummy
15     {
16     }
17 }
18 // buat namespace data\dua
19 namespace data\dua {
20     // dengan class conflict
21     class conflict{
22     }
23 }
```

G1F022030

### Penjelasan :

Pada gambar di atas membuat namespace untuk mengatur dan mengelompokkan kode dalam lingkup tertentu dan membantu membedakan nama antar kelas, fungsi, dan konstanta dalam proyek PHP. Disini untuk membuat namespace data\satu itu terdapat 3 class yang dibuat yaitu class conflict, class sample dan class dummy sedangkan untuk namespace data\dua itu terdapat 1 class yang dibuat yaitu class conflict.

2. Helper

```
data > 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";
```

G1F022030

### Penjelasan :

Pada gambar di atas terdapat namespace Helper yang digunakan untuk mengorganisir kode dalam kelompok tertentu, membantu menghindari konflik nama dan memungkinkan untuk mengelompokkan kelas, fungsi, dan konstanta. Di dalam namespace Helper, terdapat fungsi bernama helpMe() yang berfungsi untuk ketika dipanggil, akan mencetak teks "HELP ME" diikuti dengan baris baru (PHP\_EOL digunakan untuk memastikan pemisahan baris). Lalu const yang mendefinisikan konstanta dengan nama APPLICATION dan memberikan nilainya sebagai string "Belajar PHP OOP". Konstanta adalah nilai yang tidak dapat diubah selama eksekusi program.

### 3. Manager

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

G1F022030

### Penjelasan :

Pada gambar di atas terdapat class Manager yang akan deklarasi kelas dengan nama "Manager". Kelas ini memiliki satu properti yaitu \$name yang merupakan string, dan satu metode yaitu sayHello. Metode sayHello mengambil string \$name dan mencetak pesan sapaan. Lalu class VicePresident extends Manager untuk deklarasi kelas baru dengan nama "VicePresident" yang meng-extends atau mewarisi kelas "Manager". Dengan mewarisi kelas "Manager", kelas "VicePresident" akan memiliki semua properti dan metode yang dimiliki oleh kelas "Manager" dan var string \$address untuk tambahan properti pada kelas "VicePresident". Kelas "VicePresident" memiliki properti \$name yang diwarisi dari kelas "Manager" dan properti \$address yang unik untuk kelas "VicePresident".

#### 4. Person

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

G1F022030

#### Penjelasan :

Pada gambar diatas terdapat deklarasi kelas person yang merupakan awal dari definisi kelas Person, lalu membuat beberapa properti kelas, yaitu \$nama, \$alamat (dengan nullable), dan \$negara, lalu function untuk mencetak pesan sapaan menggunakan parameter \$nama, lalu function sayHelloNull untuk mencetak pesan sapaan berbeda tergantung apakah \$nama null atau tidak, lalu conts mendefinisikan konstanta AUTHOR di dalam kelas, kemudian function info untuk mencetak informasi tentang penulis menggunakan kata kunci self untuk mengakses konstanta di dalam kelas, lalu function construct untuk inialisasi properti \$nama dan \$alamat saat objek dibuat dan fuction destruct untuk mencetak pesan ketika objek dihancurkan (misalnya, saat skrip PHP selesai dijalankan).

## 5. Product

```
data > product.php
1  <?php
2  class Product
3  {
4      protected string $name;
5      protected int $price;
6      public function __construct(string $name, int $price)
7      {
8          $this->name = $name;
9          $this->price = $price;
10     }
11     public function getName(): string
12     {
13         return $this->name;
14     }
15     public function getPrice(): int
16     {
17         return $this->price;
18     }
19 }
20 class ProductDummy extends Product
21 {
22     public function info()
23     {
24         echo "Name $this->name" . PHP_EOL;
25         echo "Price $this->price" . PHP_EOL;
26     }
27 }
```

G1F022030

### Penjelasan :

Pada gambar di atas terdapat pembuatan class product dengan objek nama dengan tipe string dan objek price dengan tipe int, lalu public function yang berarti bersifat public dengan nama dan price, lalu return sebagai method yang mengembalikan nilai secara langsung atau sebuah nilai dari variable dari nama dan nilai dari price dan class ProductDummy untuk memanggil fungsi dari kelas lain product.

## 6. Programmer

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

G1F022030

### Penjelasan :

Pada gambar diatas terdapat class programmer kelas dasar yang memiliki properti publik \$name bertipe string dan memiliki konstruktor (\_\_construct) yang mengambil string dan menetapkan properti nama, lalu Class BackendProgrammer kelas ini memperluas kelas Programmer dan tidak memiliki properti atau metode tambahan apa pun serta mewarisi properti \$name dari kelas Programmer, lalu Class FrontendProgrammer mirip dengan BackendProgrammer, kelas ini juga memperluas kelas Programmer, lalu Class Company yang dimana kelas ini mempunyai properti publik \$programmer bertipe Programmer, kemudian Function sayHelloProgrammer yang dimana fungsi ini mengambil objek Programmer sebagai parameter serta menggunakan operator instanceof untuk memeriksa apakah pemrogram adalah turunan dari BackendProgrammer, FrontendProgrammer, atau hanya Programmer dan meyapa salam berdasarkan jenis programmer.

### 7. Shape

```
data > shape.php
1  <?php
2  namespace Data;
3  class Shape
4  {
5      public function getCorner()
6      {
7          return -1;
8      }
9  }
10 class Rectangle extends Shape
11 {
12     public function getCorner()
13     {
14         return 4;
15     }
16     public function getParentCorner()
17     {
18         return parent::getCorner();
19     }
20 }
```

G1F022030

### Penjelasan :

Pada gambar di atas terdapat namespace data di mana namespace berfungsi untuk mengatur kode dan membuat program lebih mudah dibaca, lalu terdapat public function getCorner sebagai method yang berguna untuk mengembalikan Component di sudut yang ditentukan. Seperti di atas yaitu return -1. Kemudian terdapat clas rectangle yang memanggil dari kelas lain pada shape dengan return 4.

## b. Constant

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

G1F022030

### Penjelasan :

- `require_once "data/Person.php";`

Baris ini digunakan untuk mengimpor (include) file `Person.php` yang terletak dalam direktori `data`. Fungsi `require_once` memastikan bahwa file hanya diimpor sekali, meskipun ada beberapa percobaan untuk mengimpornya.

- `define("TITLE", "Responsi Pemrograman Berorientasi Objek");`

Baris ini membuat konstanta dengan nama `TITLE` dan memberikan nilai string `"Responsi Pemrograman Berorientasi Objek"` padanya. Konstanta dalam PHP bersifat tetap dan tidak dapat diubah setelah didefinisikan.

- `const APP_VERSION = "1.0.0";`

Baris ini membuat konstanta lain menggunakan kata kunci `const` dengan nama `APP_VERSION` dan memberikan nilai string `"1.0.0"` padanya. Sama seperti `define`, konstanta ini juga bersifat tetap.

- `echo TITLE . PHP_EOL;`

Baris ini menampilkan nilai konstanta `TITLE` menggunakan fungsi `echo`. `PHP_EOL` digunakan untuk menambahkan garis baru setelah nilai `TITLE`.

- `echo APP_VERSION . PHP_EOL;`

Baris ini menampilkan nilai konstanta `APP_VERSION` menggunakan fungsi `echo`. `PHP_EOL` juga digunakan di sini untuk menambahkan garis baru setelah nilai `APP_VERSION`.

- `echo Person::AUTHOR . PHP_EOL;`

Baris ini menampilkan nilai properti statis `AUTHOR` dari kelas `Person`. Properti statis dapat diakses langsung dari kelas tanpa membuat objek dari kelas tersebut. `PHP_EOL` digunakan kembali untuk menambahkan garis baru setelah nilai `AUTHOR`.

### c. Constractor

```
constractor.php
1  <?php
2  // import data/person.php
3  require_once "data/Person.php";
4  // buat object new person dengan 2 parameter
5  $salsabila = new Person("Salsabila Refiani Safitri", "Bengkulu");
6  // vardump object
7  var_dump($salsabila);
8
```

G1F022030

#### Penjelasan :

Pada gambar diatas terdapat `require_once` untuk mengimpor atau memasukkan file `Person.php`. Pernyataan `require_once` digunakan untuk memasukkan dan menjalankan file PHP sehingga kelas dan fungsi yang didefinisikan dalam file tersebut dapat digunakan dalam skrip saat ini, lalu membuat objek baru dari kelas `Person` dengan nama `$salsabila`. Objek ini dibuat dengan menggunakan kata kunci `new` diikuti oleh nama kelas `Person`. Kemudian, dua parameter string ("`Salsabila Refiani Safitri`" dan "`Bengkulu`") diberikan sebagai argumen kepada konstruktor kelas `Person`, lalu `var_dump` untuk menampilkan informasi tentang objek yang baru saja dibuat. Ini akan mencakup jenis objek, jumlah properti, dan nilai dari setiap properti. Tujuan dari langkah ini adalah untuk memeriksa atau memverifikasi bahwa objek `$salsabila` telah dibuat dengan benar dan mengandung nilai yang diharapkan.

### d. Destructor

```
desturctor.php
1  <?php
2  // import data/person.php
3  require_once "data/Person.php";
4  // buat 2 object new peson dengan parameter yang berbeda
5  $salsabila = new Person("Salsabila Refiani Safitri", "Bengkulu");
6  $teguh = new Person("Teguh Nata Kusuma", "Bengkulu");
7  // tambahkan echo "Program Selesai" . PHP_EOL;
8  echo "Program Selesai" . PHP_EOL;
9
```

G1F022030

#### Penjelasan :

- `require_once "data/Person.php";`

Memasukkan (import) file "`data/Person.php`" ke dalam kode. Ini digunakan untuk mengakses kelas `Person` yang mungkin didefinisikan di dalam file tersebut. `require_once` memastikan bahwa file hanya dimasukkan sekali, bahkan jika dipanggil lebih dari satu kali.

- `$salsabila = new Person("Salsabila Refiani Safitri", "Bengkulu");`

Membuat objek baru dengan nama `$salsabila` dari kelas `Person`. Konstruktor dari kelas `Person` dipanggil dengan dua parameter: nama ("`Salsabila Refiani Safitri`") dan alamat ("`Bengkulu`").

- `$teguh = new Person("Teguh Nata Kusuma", "Bengkulu");`  
Membuat objek baru dengan nama `$teguh` dari kelas `Person`. Sama seperti sebelumnya, konstruktor kelas `Person` dipanggil dengan dua parameter: nama ("Teguh Nata Kusuma") dan alamat ("Bengkulu").
- `echo "Program Selesai" . PHP_EOL;`  
Menampilkan pesan "Program Selesai" ke layar dengan tambahan baris baru (`PHP_EOL`).

e. Function

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

G1F022030

**Penjelasan:**

Pada gambar diatas terdapat `require_once` untuk mengimpor file "data/person.php". Ini berarti bahwa kode di file tersebut akan dimasukkan ke dalam file saat ini, dan `require_once` memastikan bahwa file hanya akan diimpor sekali, bahkan jika ada percobaan untuk mengimpornya beberapa kali, lalu membuat objek baru dari kelas `Person`. `Person` adalah suatu kelas yang mungkin didefinisikan di dalam file `person.php`. Objek ini disimpan dalam variabel `$person1`. Saat membuat objek, Anda memberikan dua argumen ke konstruktor kelas `Person`: nama ("Salsabila Refiani Safitri") dan kota ("Bengkulu"), kemudian memanggil metode `sayHello` pada objek `$person1`. Metode ini mungkin didefinisikan di dalam kelas `Person` dan menerima satu argumen (dalam hal ini, nama "Salsabila Refiani Safitri"). Metode tersebut kemungkinan mencetak pesan salam atau melakukan tindakan lain yang sesuai dengan logikanya.

f. Import

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

G1F022030



### Penjelasan :

Pada gambar diatas terdapat `require_once` untuk pernyataan yang digunakan untuk memasukkan dan menjalankan file PHP atau kode lainnya. Dalam kasus ini, file "Conflict.php" dan "Helper.php" diimport ke dalam skrip PHP. Perbedaan utama antara `require` dan `require_once` adalah bahwa `require_once` memastikan bahwa file hanya diimpor sekali agar tidak ada duplikat, lalu pernyataan `use` digunakan untuk mengimpor kelas, fungsi, atau konstanta dari namespace tertentu. Dalam contoh ini, kelas `Conflict` dari namespace `Data\One` diimpor. Ini memungkinkan kita menggunakan nama kelas tersebut tanpa harus menyertakan namespace lengkap setiap kali kita menggunakannya, lalu `use function` digunakan untuk mengimpor fungsi dari namespace tertentu. Di sini, fungsi `helpMe` dari namespace `Helper` diimpor, kemudian `use const` untuk mengimpor konstanta dari namespace tertentu. Dalam contoh ini, konstanta `APPLICATION` dari namespace `Helper` diimpor, lalu `$conflict1 = new Conflict();` untuk membuat objek dari kelas `Conflict` yang berada dalam namespace `Data\One`. Objek ini diakses melalui variabel `$conflict1`, lalu `$conflict2 = new data\dua\Conflict();` untuk membuat objek dari kelas `Conflict` yang berada dalam namespace `data\dua`. Objek ini diakses melalui variabel `$conflict2`. Perlu diperhatikan bahwa penggunaan `data\dua\Conflict` menunjukkan namespace yang berbeda dari objek sebelumnya, kemudian Memanggil fungsi `helpMe()` yang telah diimpor dari namespace `Helper` dan `echo APPLICATION . PHP_EOL;` untuk mencetak nilai konstanta `APPLICATION` yang telah diimpor dari namespace `Helper`, diikuti oleh sebuah newline (`PHP_EOL`).

g. `importAlias`

```
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();
9  $conflict2 = new Conflict2();
10 help();
11 echo APP . PHP_EOL;
```

G1F022030

### Penjelasan :

- Dua kelas `Conflict1` dan `Conflict2` diinisialisasi dari namespace yang berbeda (`data\satu` dan `data\dua`).
- Fungsi `helpMe` dari namespace `Helper` dipanggil melalui alias `help`.
- Konstanta `APPLICATION` dari namespace `Helper` diakses melalui alias `APP` dan ditampilkan menggunakan `echo`.

## 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 = "Salsabila";
7  $manager->sayHello("Teguh");
8  // buat object new vicepresident dan tambahkan value nama kemudian panggil function
9  $vicePresident1 = new VicePresident();
10 $vicePresident1->nama = "Teguh";
11 $vicePresident1->alamat = "Bengkulu";
12 $vicePresident1->sayHello("Salsabia");
```

G1F022030

### Penjelasan :

- `require_once "data/Manager.php";`  
Ini adalah pernyataan untuk mengimpor file Manager.php yang berisi definisi kelas Manager dan VicePresident. `require_once` digunakan untuk memastikan bahwa file hanya diimpor sekali dan tidak lebih.
- `$manager = new Manager();`  
Membuat objek baru dari kelas Manager.
- `$manager->nama = "Salsabila";`  
Menetapkan nilai properti nama pada objek \$manager menjadi "Salsabila".
- `$manager->sayHello("Teguh");`  
Memanggil metode sayHello dari objek \$manager dengan parameter "Teguh".
- `$vicePresident1 = new VicePresident();`  
Membuat objek baru dari kelas VicePresident.
- `$vicePresident1->nama = "Teguh";`  
Menetapkan nilai properti nama pada objek \$vicePresident1 menjadi "Teguh".
- `$vicePresident1->alamat = "Bengkulu";`  
Menetapkan nilai properti alamat pada objek \$vicePresident1 menjadi "Bengkulu".
- `$vicePresident1->sayHello("Salsabia");`  
Memanggil metode sayHello dari objek \$vicePresident1 dengan parameter "Salsabila".

## i. namespace

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

G1F022030

### Penjelasan :

Pada gambar diatas terdapat namespace yang digunakan untuk secara unik mengidentifikasi satu atau lebih nama dari nama serupa lainnya dari berbagai objek, grup atau namespace secara umum. Dengan php yang menjalankan proses pemrograman pada saat proses runtime. Dengan mengimport file data/conflict dari objek . kemudian terdapat file data/helper yang kemudian digunakan echo untuk menampilkan echo\application dengan method Helper/helpMe.

### 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  $person1 = new Person("Salsabila","Bengkulu");
6  // manipulasi properti nama, alamat, negara
7  $person1->nama = "Salsabila Refiani Safitri";
8  // menampilkan hasil
9  echo "nama = {$person1->nama}";
```

G1F022030

### Penjelasan :

- `require_once "data/person.php";`

Ini adalah pernyataan yang mengimpor (meng-include) file person.php. `require_once` digunakan untuk memastikan bahwa file hanya diimpor satu kali. File ini berisi definisi kelas Person yang digunakan di program.

- `$person1 = new Person("Salsabila","Bengkulu");`

Baris ini membuat objek baru dengan menggunakan kelas Person. Objek ini disimpan dalam variabel `$person1`. Saat objek dibuat, konstruktor kelas Person dipanggil dengan dua parameter yaitu "Salsabila" untuk nama dan "Bengkulu" untuk alamat.

- `$person1->nama = "Salsabila Refiani Safitri";`

Baris ini memanipulasi properti nama dari objek `$person1`. Properti nama pada awalnya diatur menjadi "Salsabila" melalui konstruktor, tetapi kemudian nilainya diubah menjadi "Salsabila Refiani Safitri".

- `echo "nama = {$person1->nama}";`

Baris ini menampilkan hasil ke layar. Itu menggunakan fungsi `echo` untuk mencetak string ke output. Dalam string tersebut, kita menggunakan sintaks `{ }` untuk menampilkan nilai properti nama dari objek `$person1`.

#### k. Parents

```
parent.php
1  <?php
2  require_once "data/Shape.php";
3  use Data\{Shape, Rectangle};
4  $shape = new Shape();
5  echo $shape->getCorner() . PHP_EOL;
6  $rectangle = new Rectangle();
7  echo $rectangle->getCorner() . PHP_EOL;
8  echo $rectangle->getParentCorner() . PHP_EOL;
```

G1F022030

#### Penjelasan :

- `require_once "data/Shape.php";`  
Mengimpor file Shape.php yang terletak dalam direktori "data".
- `use Data\{Shape, Rectangle};`  
Menggunakan namespace "Data" dan mengimpor kelas Shape dan Rectangle. Ini memungkinkan kita untuk menggunakan kelas-kelas ini tanpa menuliskan namespace mereka setiap kali.
- `$shape = new Shape();`  
Membuat instance dari kelas Shape.
- `echo $shape->getCorner() . PHP_EOL;`  
Memanggil metode `getCorner()` dari objek `$shape` dan mencetak hasilnya ke layar, diikuti dengan `PHP_EOL` (end of line) untuk membuat baris baru.
- `$rectangle = new Rectangle();`  
Membuat instance dari kelas Rectangle.
- `echo $rectangle->getCorner() . PHP_EOL;`  
Memanggil metode `getCorner()` dari objek `$rectangle` dan mencetak hasilnya ke layar, diikuti dengan `PHP_EOL`.
- `echo $rectangle->getParentCorner() . PHP_EOL;`  
Memanggil metode `getParentCorner()` dari objek `$rectangle` dan mencetak hasilnya ke layar, diikuti dengan `PHP_EOL`.

#### l. Polymorphism

```
polymorphism.php
1  <?php
2  require_once "data/Programmer.php";
3  $company = new Company();
4  $company->programmer = new Programmer("Salsabila Refiani Safitri");
5  var_dump($company);
6  $company->programmer = new BackendProgrammer("Salsabila Refiani Safitri");
7  var_dump($company);
8  $company->programmer = new FrontendProgrammer("Salsabila Refiani Safitri");
9  var_dump($company);
10 sayHelloProgrammer(new Programmer("Salsabila Refiani Safitri"));
11 sayHelloProgrammer(new BackendProgrammer("Salsabila Refiani Safitri"));
12 sayHelloProgrammer(new FrontendProgrammer("Salsabila Refiani S
```

G1F022030

### Penjelasan :

Pada gambar diatas terdapat `require_once "data/Programmer.php"` baris ini berisi file PHP yang diperlukan (`Programmer.php`) yang berisi definisi kelas yang digunakan dalam kode, lalu `$company = new Company` untuk sebuah instance dari kelas `company` dibuat dan ditugaskan ke variabel `$company`, kemudian `$company->programmer = new Programmer` untuk instance dari kelas `Programmer` dibuat dengan nama "Salsabila Refiani Safitri" dan ditugaskan ke properti `programmer` dari objek `$company`. Fungsi `var_dump($company)` kemudian digunakan untuk menampilkan informasi tentang objek `$company`, lalu `$company->programmer = new BackendProgrammer` proses yang sama diulangi, tapi kali ini, instance baru dari kelas `BackendProgrammer` ditugaskan ke properti `programmer` dari objek `$company`, kemudian `$company->programmer = new FrontendProgrammer` instance baru dari kelas `FrontendProgrammer` ditugaskan ke properti `programmer` dari objek `$company`, dan informasi tentang objek `$company` dikeluarkan menggunakan `var_dump()` dan Fungsi `sayHelloProgrammer` dipanggil tiga kali, setiap kali dengan tipe objek pemrogram yang berbeda (`Programmer`, `BackendProgrammer`, dan `FrontendProgrammer`). Fungsi ini mungkin mengeluarkan salam atau melakukan beberapa tindakan berdasarkan jenis pemrogram yang meneruskannya.

### m. Properti

```
❏ properti.php
1  <?php
2  // import data/person.php
3  require_once "data/person.php";
4  // buat object baru dari kelas person
5  $person1 = new Person("Salsabila Refiani Safitri","Bengkulu");
6  // manipulasi properti nama person
7  $person1->nama = "Salsabila";
8  // menampilkan hasil
9  echo "Nama = {$person1->nama}" . PHP_EOL;
10 echo "Alamat = {$person1->alamat}" . PHP_EOL;
11 echo "Negara = {$person1->negara}" . PHP_EOL;
```

G1F022030

### Penjelasan :

- Import File Kelas (`person.php`)

Pada langkah ini, file `person.php` diimpor ke dalam file saat ini menggunakan pernyataan `require_once`. File ini mungkin berisi definisi kelas `Person` dan mungkin juga beberapa inisialisasi atau kode lainnya.

- Membuat Objek Baru dari Kelas `Person`

Objek baru dari kelas `Person` dibuat dengan menggunakan operator `new`. Constructor dari kelas `Person` mungkin mengharapkan dua parameter: nama dan alamat.

- Manipulasi Properti Objek

Properti nama dari objek `$person1` diubah nilainya menjadi "Salsabila".

- Menampilkan Hasil

Hasilnya dicetak ke layar menggunakan pernyataan echo. Nilai dari properti nama dan alamat dicetak, dan kemungkinan ada properti lain seperti negara yang dicetak (meskipun properti ini tidak didefinisikan dalam kode yang diberikan).

n. SelfKeyword

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

G1F022030

Penjelasan :

- require\_once "data/person.php";

Ini adalah pernyataan yang mengimpor file person.php. require\_once digunakan untuk memastikan bahwa file hanya diimpor sekali, bahkan jika ada beberapa panggilan.

- \$person1 = new Person("Salsabila Refiani Safitri","Bengkulu");

Baris ini membuat objek baru dari kelas Person. Objek ini disimpan dalam variabel \$person1. Konstruktor Person menerima dua argumen, yaitu nama dan kota, yang diberikan sebagai "Salsabila Refiani Safitri" dan "Bengkulu" dalam contoh ini.

- \$person1->sayHello("Salsabila Refiani Safitri");

Memanggil metode sayHello dari objek \$person1 dengan melewati argumen "Salsabila Refiani Safitri". Metode ini mungkin digunakan untuk menampilkan pesan sapaan atau tindakan serupa.

- \$person1->info();

Memanggil metode info dari objek \$person1. Metode ini mungkin digunakan untuk menampilkan informasi tambahan tentang objek atau melakukan tindakan lainnya.

o. ThisKeyword

```
thisKeyword.php
1  <?php
2  // import data/person.php
3  require_once "data/person.php";
4  // buat object dari kelas person
5  $person1 = new Person("Salsabila Refiani Safitri", "Bengkulu");
6  // tambahkan value nama di object
7  $person1->nama = "Salsabila Refiani Safitri";
8  // panggil function sayHelloNull dengan parameter
9  $person1->sayHelloNull("Teguh Nata Kusuma");
10 // buat object dari kelas person
11 $person2 = new Person("Teguh Nata Kusuma", "Bengkulu");
12 // tambahkan value nama di object
13 $person2->nama = "Teguh";
14 // panggil function sayHelloNull dengan parameter null
15 $person2->sayHelloNull(null);
```

G1F022030

### Penjelasan :

- `require_once "data/person.php";`

Ini adalah pernyataan untuk mengimpor file "person.php" yang berada di dalam folder "data". `require_once` digunakan untuk memastikan bahwa file hanya diimpor sekali, dan jika sudah diimpor sebelumnya, maka tidak akan diimpor lagi.

- `$person1 = new Person("Salsabila Refiani Safitri", "Bengkulu");`

Baris ini membuat objek dari kelas "Person" dengan nama `$person1`. Objek ini diinisialisasi dengan dua parameter: nama "Salsabila Refiani Safitri" dan lokasi "Bengkulu".

- `$person1->nama = "Salsabila Refiani Safitri";`

Baris ini menetapkan nilai "Salsabila Refiani Safitri" ke properti nama dari objek `$person1`.

- `$person1->sayHelloNull("Teguh Nata Kusuma");`

Baris ini memanggil metode `sayHelloNull` pada objek `$person1` dengan parameter "Teguh Nata Kusuma".

- `$person2 = new Person("Teguh Nata Kusuma", "Bengkulu");`

Baris ini membuat objek lain dari kelas "Person" dengan nama `$person2`. Objek ini diinisialisasi dengan dua parameter: nama "Teguh Nata Kusuma" dan lokasi "Bengkulu".

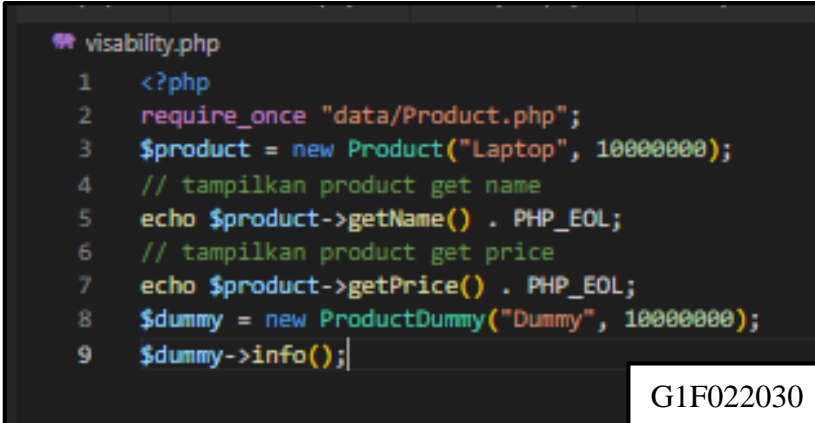
- `$person2->nama = "Teguh";`

Baris ini menetapkan nilai "Teguh" ke properti nama dari objek `$person2`.

- `$person2->sayHelloNull(null);`

Baris ini memanggil metode `sayHelloNull` pada objek `$person2` dengan parameter `null`.

### p. Visability



```
visibility.php
1  <?php
2  require_once "data/Product.php";
3  $product = new Product("Laptop", 10000000);
4  // tampilkan product get name
5  echo $product->getName() . PHP_EOL;
6  // tampilkan product get price
7  echo $product->getPrice() . PHP_EOL;
8  $dummy = new ProductDummy("Dummy", 10000000);
9  $dummy->info();|
```

G1F022030

### Penjelasan :

- `require_once "data/Product.php";`

Mengimpor file Product.php, yang berisi definisi kelas Product.

- `new Product("Laptop", 10000000);`

Membuat objek dari kelas Product dengan nama "Laptop" dan harga 10,000,000.

- `echo $product->getName() . PHP_EOL;`

Menampilkan nama produk menggunakan method `getName()` dari objek `$product`.

- `echo $product->getPrice() . PHP_EOL;`

Menampilkan harga produk menggunakan method `getPrice()` dari objek `$product`.

- `new ProductDummy("Dummy", 10000000);`

Membuat objek dari kelas `ProductDummy` dengan nama "Dummy" dan harga 10,000,000.

- `$dummy->info();`

Memanggil method `info()` dari objek `$dummy`, yang mungkin didefinisikan di dalam kelas `ProductDummy`.