Nama : Intan Budiarty

Npm : G1F0222048

Responsi PBO

Soal.

1.Silahkan lakukan git clone repositori dari https://github.com/alzahfariski/bahan-ajar-pbo

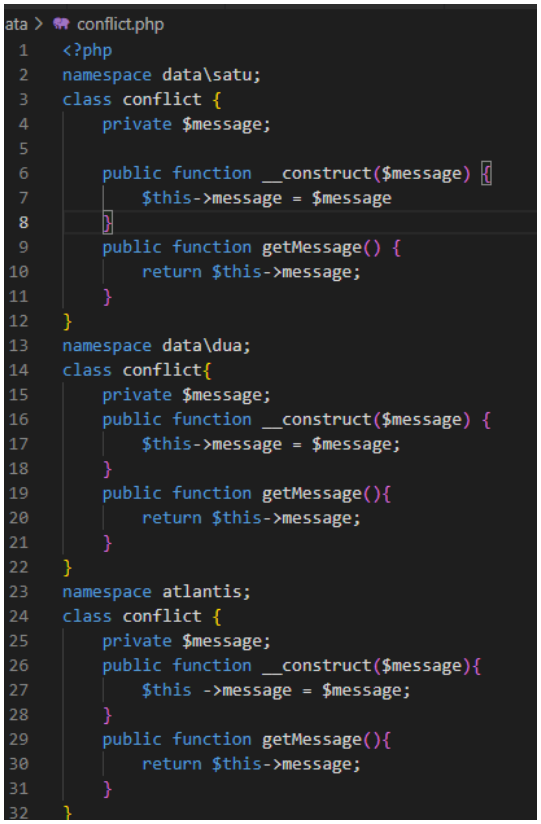
2.lengkapi code php yang belum lengkap sehingga setiap file dapat di run dan tidak memunculkan eror.

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) penjelelasan kalian akan mempengaruhi penilaian.

Jawaban:

1. **Conflict.php**



Penjelasan:

Namespace data\satu diatas memiliki sebuah class yang bernama conflict yang memiliki properti private $message dan sebuah constructor untuk menginisialisasi nilai

properti tersebut juga Terdapat method getMessage yang mengembalikan nilai dari properti $message. Lalu terdapat namespace kedua dengan class conflict yang mirip dengan yang pertama.Juga memiliki properti private $message, constructor untuk menginisialisasi nilai properti, dan method getMessage untuk mendapatkan nilai properti.pada baris ke-24 terdapat class ketiga yaitu atlantis yang memiliki property dan constructor yang sama seperti kelas sebelumnya.setiap namespace memiliki class conflict yang memiliki property message dan method getmessage untuk mengembalikan nilai property tersebut. Dengan demikian penggunaan namesapace dalam php untuk mengelompokkan class dengan nama yang sama ke dalam namespace yang berbeda.

2. **HELPER.PHP**

data > helper.php

```php
<?php

namespace Helper;

function helpMe()
{
    echo "HELP ME" . PHP_EOL;
}

const APPLICATION = "Belajar PHP OOP";
```

Penjelasan:

Namespace Helper berfungsi untuk menggelompokan dan mengorganisasi kode agar tidak bertebrakan dangan kode yang mungkin memiliki nama yang sama di namespace lain.lalu terdapat source code function yang digunakan untuk mendefinisilan sebuah function bernama helpme. Function ini mencetak teks help me yang berdapat didalam echo dan diikuti dengan newline php_eol ke output.kemudian terdapat constant yang bernama application dan nilai berlajar php oop. Constant adalah variable yang nilainya todak dapat diuubah setelah didefinisikan. Source code namespace ini dihunakan untuk mendefinisikan namespace dalam php, namespace membantu menghindari konflik nama antar class, function, dan constant. Function helpme didalam namespace helper dapat diakses dengan menginport namespace. Lalu tanda \ digunakan untuk memisahkan namesapace.

3. **Manager.php**

manager.php

```php
<?php

// buat kelas manager dengan properti nama dan function sayHello
class Manager
{
    var string $nama;

    function sayHello(string $nama): void
    {
        echo "Hi $nama, my name is $this->nama" . PHP_EOL;
    }
}

// buat kelas VicePresident dengan extends manager
class VicePresident extends Manager
{

}
```

Penjelasan:

Kelas Manager dideklarasikan dengan memiliki properti $nama yang merupakan tipe data string.Terdapat fungsi sayHello yang menerima parameter $nama dengan menggunakan tipe data string dan mencetak pesan menggunakan fungsi echo. Pesan tersebut mengandung informasi dari parameter yang diterima dan nilai dari properti $this->nama. Selanjutnya kelas VicePresident yang dideklarasi dengan menggunakan kata kunci extends yang berguna untuk mewarisi sifat atau perilaku dari kelas manager. Dengan kata lain, vicePresident akan memiliki semua property dan metode yang dimiliki oleh manager.

4. Person.php

```php
<?php
class Person{
    var string $nama;
    var ?string $alamat = null;
    var string $negara = "indonesia";
    function sayHello(string $nama){
        echo "Hello $nama" . PHP_EOL;
    }
    function sayHelloNull(?string $nama)
    {
        if (is_null($nama)) {
            echo "Hi, my nama is $this->nama" . PHP_EOL;
        } else {
            echo "Hi $nama, my nama is $this->nama" . PHP_EOL;
        }
    }
    const AUTHOR = "Proyek PBO B 23";
    function info()
    {
        echo "Author : " . self::AUTHOR . PHP_EOL;
    }
    function __construct(string $nama, ?string $alamat)
    {
        $this->nama = $nama;
        $this->alamat = $alamat;
    }
    function __destruct()
    {
        echo "Object person $this->nama is destroyed" . PHP_EOL;
    }
}
```

Penjelasan:

Source code di atas adalah implementasi dasar dalam bahasa pemrograman PHP untuk membuat kelas Person dengan beberapa properti, metode, dan konstanta.$nama adalah

Sebuah properti bertipe string yang menyimpan nama dari objek Person.$alamat adalh Sebuah properti bertipe nullable string (?string) yang dapat berisi alamat dari objek Person. Properti ini diinisialisasi dengan nilai default null $negara, Sebuah properti bertipe string yang menyimpan nilai default "indonesia". sayHello(string $nama) Metode ini menerima parameter nama dan mencetak pesan sapaan. sayHelloNull(?string $nama): Metode ini menerima parameter nama yang bisa null. Jika null, mencetak pesan dengan menggunakan nama objek Person, jika tidak null, mencetak pesan dengan menggunakan nama yang diberikan.info() Metode ini mencetak informasi penulis (author) menggunakan konstanta AUTHOR. AUTHOR adalh Sebuah konstanta yang menyimpan string "Proyek PBO B 23

5. **Produk.php**

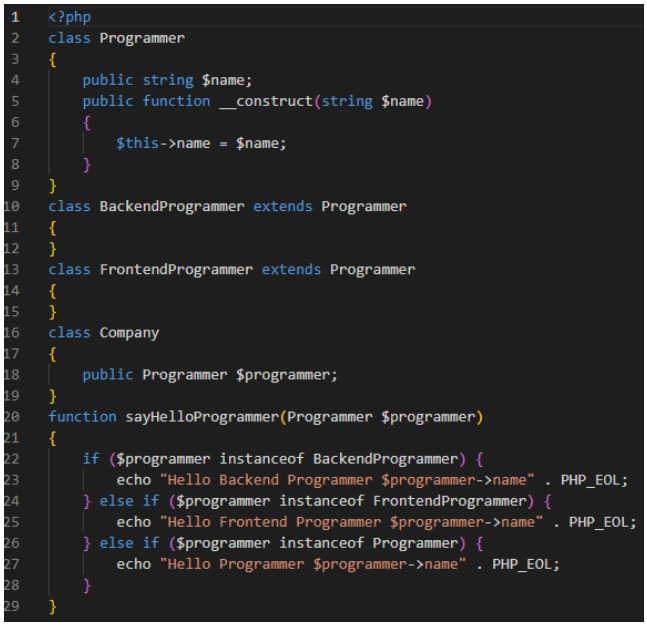
```php
<?php

class Product
{
    protected string $name;
    protected int $price;

    public function __construct(string $name, int $price)
    {
        $this->name = $name;
        $this->price = $price;
    }

    public function getName(): string
    {
        return $this->name;
    }

    public function getPrice(): int
    {
        return $this->price;
    }
}

class ProductDummy extends Product
{

    public function info()
    {
        echo "Name $this->name" . PHP_EOL;
        echo "Price $this->price" . PHP_EOL;
    }
```

Penjelasan:

Kelas Product memiliki dua properti proteksi (protected), yaitu $name bertipe string dan $price bertipe int.Konstruktor __construct digunakan untuk menginisialisasi nilai dari properti $name dan $price saat objek Product dibuat.Terdapat dua metode getter (getName dan getPrice) yang digunakan untuk mendapatkan nilai dari properti $name dan $price. 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 property. Dengan menggunakan konsep pewarisan, kelas ProductDummy dapat menggunakan properti dan metode dari kelas Product tanpa mendefinisikan ulang. Hal ini memungkinkan untuk mengelompokkan dan mewarisi perilaku-perilaku umum dalam kelas induk.

6. **progammer.php**



```php
<?php
class Programmer
{
    public string $name;
    public function __construct(string $name)
    {
        $this->name = $name;
    }
}
class BackendProgrammer extends Programmer
{
}
class FrontendProgrammer extends Programmer
{
}
class Company
{
    public Programmer $programmer;
}
function sayHelloProgrammer(Programmer $programmer)
{
    if ($programmer instanceof BackendProgrammer) {
        echo "Hello Backend Programmer $programmer->name" . PHP_EOL;
    } else if ($programmer instanceof FrontendProgrammer) {
        echo "Hello Frontend Programmer $programmer->name" . PHP_EOL;
    } else if ($programmer instanceof Programmer) {
        echo "Hello Programmer $programmer->name" . PHP_EOL;
    }
}
```

Penjelasan:

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. Dua kelas yang meng-extend kelas Programmer. Ini menunjukkan pewarisan di mana BackendProgrammer dan FrontendProgrammer memiliki semua properti dan metode yang dimiliki oleh Programmer. Fungsi dari sayHelloProgrammer ini menerima objek dari kelas Programmer sebagai parameter dan menyapa programmer sesuai dengan jenisnya.Menggunakan operator instanceof untuk memeriksa tipe objek, sehingga bisa menyapa BackendProgrammer, FrontendProgrammer, atau Programmer biasa.

7. **shape.php**

data > shape.php

```php
<?php
namespace Data;
class Shape
{
    public function getCorner()
    {
        return -1;
    }
}
class Rectangle extends Shape
{
    public function getCorner()
    {
        return 4;
    }
    public function getParentCorner()
    {
        return parent::getCorner();
    }

}
```

Penjelasan:

Source code di atas adalah implementasi sederhana dalam bahasa pemrograman PHP yang menggunakan namespace untuk mendefinisikan kelas Shape dan Rectangle, serta mendemonstrasikan penggunaan pewarisan (inheritance) dan pemanggilan metode dari kelas induk. Namespace digunakan untuk mengorganisir kelas-kelas di dalam PHP dan mencegah konflik nama kelas yang mungkin muncul. Dalam hal ini, kelas Shape dan Rectangle berada dalam namespace Data. Kelas Shape memiliki metode getCorner yang mengembalikan nilai -1. Kelas ini merupakan kelas dasar. Kelas Rectangle meng-extend (mewarisi) kelas Shape. Ini berarti Rectangle akan memiliki metode getCorner yang sama dengan kelas Shape, tetapi dapat menggantinya dengan implementasi yang berbeda.Kelas Rectangle memiliki metode getCorner yang mengembalikan nilai 4. Ini menunjukkan bahwa bentuk persegi memiliki 4 sudut.Terdapat metode getParentCorner yang menggunakan kata kunci parent untuk memanggil metode getCorner dari kelas induk (Shape). Ini membuktikan pemanggilan metode dari kelas induk yang digunakan oleh kelas anak.

8. **Constant.php**

constant.php

```php
<?php

// import data/person.php
require_once "data/Person.php";

// buat define
define("APPLICATION", "Belajar PHP OOP");

// buat const app version
const APP_VERSION = "1.0.0";

// tampilkan hasil
echo APPLICATION . PHP_EOL;
echo APP_VERSION . PHP_EOL;
echo Person::AUTHOR . PHP_EOL;
```

Penjelasan:

Source code di atas merupakan contoh penggunaan beberapa konsep, termasuk penggunaan require_once untuk mengimpor file, penggunaan define untuk membuat konstanta, dan penggunaan const untuk membuat konstanta class. Dengan menggunakan require_once, file Person.php diimport ke dalam file saat ini. Ini memungkinkan penggunaan kelas dan konstanta yang didefinisikan dalam file tersebut. define digunakan untuk membuat konstanta. Dalam hal ini, sebuah konstanta bernama APPLICATION didefinisikan dengan nilai "Belajar PHP OOP". Konstanta ini dapat digunakan di seluruh skrip PHP. const digunakan untuk membuat konstanta. Dalam hal ini, konstanta APP_VERSION didefinisikan dengan nilai "1.0.0". Konstanta ini dapat digunakan di seluruh skrip PHP.

9. **Constractor.php**

constractor.php

```php
<?php

// import data/person.php
require_once "data/Person.php";
// buat object new person dengan 2 parameter
$intan = new Person("Intan", "Bengkulu");
// vardump object
var_dump($intan);

```

Penjelasan:

Pada kode PHP ini, pertama-tama kita mengimpor file Person.php yang berisi definisi kelas Person dari direktori data. Kemudian kita membuat objek baru dari kelas Person dengan memberikan dua parameter yaitu “Intan” sebagai nama dan “Bengkulu” sebagai alamat. Dengan menggunakan sintaks $Intan = new Person("Intan", "Bengkulu");, kita membuat instance objek baru dari kelas Person dan menyimpannya dalam variabel $Intan. Kemudian kita menggunakan var_dump($Intan); untuk menampilkan informasi lengkap tentang objek, termasuk tipe data, nilai properti, dan informasi lainnya

10. **Destructor.php**

```php
<?php

// import data/person.php
require_once "data/Person.php";

// buat 2 object new peson dengan parameter yang berbeda
$intan = new Person("Intan", "Bengkulu");
$budiarty = new Person("Budiarty", "jambi");

// tambahkan echo "Program Selesai" . PHP_EOL;
echo "proyek PBO" . PHP_EOL;

```

Penjelasan:

Dua objek baru, $intan dan $budiarty, dibuat dari kelas Person. Kedua objek ini diinisialisasi dengan nilai yang berbeda pada parameter konstruktor ($nama dan $alamat). Menampilkan string "proyek PBO" ke layar menggunakan fungsi echo dan menambahkan baris baru (PHP_EOL) setelahnya.

11. Funcition.php

function.php

```php
<?php
// import data/person.php
require_once "data/person.php";
// buat object baru dari kelas person
$person1 = new Person("intan","Bengkulu");
// panggil function
$person1->sayHello("intan");

```

Penjelasan:

adapun penjelasan kode diatas yaitu :Import data/person.php: Dalam baris pertama, kita menggunakan require_once untuk mengimport file data/person.php. Dalam file ini, terdapat kelas Person yang akan kita gunakan.Buat objek baru dari kelas person: Baris kedua dan ketiga membuat objek baru bernama $person1. Objek ini dibuat dari kelas Person dan diberi parameter "Intan" dan "Bengkulu". Panggil function: Baris keempat memanggil function sayHello yang ada di dalam kelas Person. Function ini akan mencetak "Hello, Intan!" pada layar.

12. Import.php

import.php

```php
<?php
require_once "data/Conflict.php";
require_once "data/Helper.php";
use Data\One\Conflict;
use function Helper\helpMe;
use const Helper\APPLICATION;
$conflict = new Conflict("belajarrr");
echo $conflict->getMessage();

helpMe();
echo APPLICATION . PHP_EOL;
```

Penjelasan:

Dalam source code di atas, beberapa konsep PHP digunakan, termasuk use untuk mengimpor namespace, fungsi, dan konstanta dari file yang berbedayaitu Menggunakan require_once untuk mengimpor file Conflict.php dan Helper.php, yang berisi definisi kelas Conflict dan fungsi helpMe. Menggunakan use untuk mengimpor kelas Conflict dari namespace Data\One, fungsi helpMe dari namespace Helper, dan konstanta APPLICATION dari namespace Helper.

Membuat objek $conflict dari kelas Conflict dengan memberikan nilai "belajarrr" sebagai argumen konstruktor. Memanggil metode getMessage dari objek $conflict dan menampilkannya menggunakan perintah echo. Memanggil fungsi helpMe yang diimpor menggunakan use. Fungsi ini mungkin memiliki implementasi apa pun yang didefinisikan di dalam file Helper.php. Menampilkan nilai konstanta APPLICATION yang diimpor menggunakan use. Konstanta ini mungkin memiliki nilai "Belajar PHP OOP" atau nilai yang sesuai yang didefinisikan di dalam file Helper.php.

13. imortAlias.php

```php
<?php
require_once "data/Conflict.php";
require_once "data/Helper.php";
use Data\One\Conflict as Conflict1;
use Data\Two\Conflict as Conflict2;
use function Helper\helpMe as help;
use const Helper\APPLICATION as APP;
$conflict1 = new Conflict1("pesan untuk conflict1");
$conflict2 = new Conflict2("pesan untuk conflict2");

help();

echo APP . PHP_EOL;
```

Penjelasan:

Dalam kode yang disediakan, ada dua namespace Data\satudan Data\duakeduanya memiliki kelas bernama Conflict. Untuk mengatasi konflik ini, use pernyataan tersebut digunakan. Pernyataan ini usedigunakan untuk mengimpor kelas dari satu namespace ke namespace lainnya, yang secara efektif memberinya nama baru di namespace saat ini. 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 Helpernamespace diimpor dan diberi alias help. Konstanta APPLICATIONdari Helpernamespace 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

```php
<?php
require_once "data/Manager.php";
$manager = new Manager();
$manager->nama = "Intan";
$manager->sayHello("heyy gays");
$vp = new VicePresident();
$vp->nama = "budiarty";
$vp->sayHello("heyy gays");

```

Penjelasan:

Dalam source code di atas, terdapat implementasi beberapa kelas dan penggunaan pewarisan (inheritance). Adapun fungsi dari source code datas adalah Menggunakan require_once untuk mengimpor file Manager.php, yang berisi definisi kelas Manager dan VicePresident. Membuat objek $manager dari kelas Manager. Mengatur nilai properti $nama pada objek $manager menjadi "Intan".Memanggil metode sayHello pada objek $manager dengan memberikan parameter "heyy gays". Metode ini akan mencetak pesan sapaan menggunakan nama yang diberikan dan nilai dari properti $nama pada objek Manager. Membuat objek $vp dari kelas VicePresident. Karena VicePresident meng-extend Manager, objek ini akan memiliki properti dan metode yang sama seperti Manager. Mengatur nilai properti $nama pada objek $vp menjadi "budiarty".Memanggil metode sayHello pada objek $vp dengan memberikan parameter "heyy gays". Metode ini juga akan mencetak pesan sapaan menggunakan nama yang diberikan dan nilai dari properti $nama pada objek VicePresident.

15. .**nameSpace.php**

nameSpace.php

```php
<?php
// Buat namespace
namespace atlantis\conflict;
// Import data dari conflict
require_once "data/conflict.php";
// Buat object dari namespace yang di buat
$conflictObject = new conflict\ConflictClass();
// Import data helper
require_once "data/helper.php";
// Tampilkan helper menggunakan echo
// Masukkan Helper\helpMe();
use function Helper\helpMe as help;
echo help();
?>
```

Penjelasan:

Dalam source code di atas, beberapa konsep PHP digunakan, termasuk penggunaan namespace, import file dari namespace lain, pembuatan objek dari kelas dalam namespace tersebut, dan penggunaan alias pada fungsi. Dengan menggunakan namespace, sebuah namespace baru dengan nama atlantis\conflict dibuat. Menggunakan require_once untuk mengimpor file conflict.php, yang berisi definisi kelas ConflictClass dalam namespace conflict. Membuat objek $conflictObject dari kelas ConflictClass yang berada dalam

namespace conflict. Menggunakan require_once untuk mengimpor file helper.php, yang berisi definisi fungsi helpMe dalam namespace Helper.Menggunakan use function untuk mengimpor fungsi helpMe dari namespace Helper dan memberikan alias help untuk penggunaan lebih lanjut. Menampilkan hasil dari pemanggilan fungsi helpMe dengan alias help menggunakan perintah echo.

16. Object.php

object.php

```php
<?php
// import data/person.php
require_once "data/person.php";
// buat object baru dari kelas person
$person = new Person("Intan","Bengkulu");
// manipulasi properti nama, alamat, negara
$person->nama = "Intan";
$person->alamat = "Bengkulu";
$person->negara = "Indonesia";
// menampilkan hasil
echo "nama = {$person->nama}" . PHP_EOL;
echo "alamat = {$person->alamat}" . PHP_EOL;
echo "negara = {$person->negara}" . PHP_EOL;

```

Penjelasan:

Membuat objek baru dari kelas Person dengan memberikan nilai "Intan" dan "Bengkulu" sebagai parameter konstruktor. Memanipulasi nilai properti objek $person. Dalam hal ini, nilai properti nama diubah menjadi "Intan", alamat menjadi "Bengkulu", dan negara menjadi "Indonesia"Lalu menampilkan hasil manipulasi properti dengan menggunakan perintah echo. Nilai-nilai properti nama, alamat, dan negara objek $person akan ditampilkan di layar.

17. Parent.php

parent.php

```php
<?php
require_once "data/Shape.php";
use Data\{Shape, Rectangle};
$shape = new Shape();
echo $shape->getCorner() . PHP_EOL;
$rectangle = new Rectangle();
echo $rectangle->getCorner() . PHP_EOL;
echo $rectangle->getParentCorner() . PHP_EOL;
```

Penjelasan:

kode ini 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

```php
<?php
require_once "data/Programmer.php";
$company = new Company();
$company->programmer = new Programmer("Intan");
var_dump($company);
$company->programmer = new BackendProgrammer("Intan");
var_dump($company);
$company->programmer = new FrontendProgrammer("Intan");
var_dump($company);
sayHelloProgrammer(new Programmer("Intan"));
sayHelloProgrammer(new BackendProgrammer("Intan"));
sayHelloProgrammer(new FrontendProgrammer("Intan"));
```

Penjelasan:

Adapun fungsi dari source code diatas adalah, Menggunakan require_once untuk mengimpor file Programmer.php, yang berisi definisi kelas-kelas terkait dengan pemrograman (Programmer, BackendProgrammer, FrontendProgrammer) dan fungsi sayHelloProgrammer. Membuat objek $company dari kelas Company.Memberikan nilai objek $programmer pada objek $company dengan membuat objek baru dari kelas Programmer dengan nama "Intan".Menggunakan var_dump untuk menampilkan informasi struktur dari objek $company. Ini memberikan tampilan detail tentang properti dan nilai-nilai dalam objek tersebut. Mengganti nilai objek $programmer pada objek $company dengan membuat objek baru dari kelas BackendProgrammer dengan nama "Intan".Menggunakan var_dump untuk menampilkan informasi struktur dari objek $company setelah perubahan. Sekarang, properti programmer pada $company adalah objek dari kelas BackendProgrammer. Mengganti nilai objek $programmer pada objek $company dengan membuat objek baru dari kelas FrontendProgrammer dengan nama "Intan".Menggunakan var_dump untuk menampilkan informasi struktur dari objek $company setelah perubahan. Sekarang, properti programmer pada $company adalah objek dari kelas FrontendProgrammer. Memanggil fungsi sayHelloProgrammer dengan parameter berbagai jenis objek programmer (berbagai sub-kelas dari Programmer). Fungsi ini akan menampilkan pesan sapaan sesuai dengan jenis programmer yang diberikan

19. Properti.php

properti.php

```php
<?php
// import data/person.php
require_once "data/person.php";
// buat object baru dari kelas person
$person1 = new Person("Intan","bengkulu");
// manipulasi properti nama person
$person1->nama = "Intan";
// menampilkan hasil
echo "nama = {$person1->nama}" . PHP_EOL;
echo "alamat = {$person1->alamat}" . PHP_EOL;
echo "negara = {$person1->negara}" . PHP_EOL;
```

Penjelasan:

Menggunakan require_once untuk mengimpor file person.php, yang berisi definisi kelas Person. Membuat objek baru dari kelas Person dengan memberikan nilai "Intan" dan "bengkulu" sebagai parameter konstruktor. Memanipulasi nilai properti nama pada objek $person1, menggantinya menjadi "Intan".Menampilkan hasil manipulasi properti menggunakan perintah echo. Nilai-nilai properti nama, alamat, dan negara dari objek $person1 akan ditampilkan di layar.

20. selfKeyword.php

selfKeyword.php

```php
<?php
// import data/person.php
require_once "data/person.php";
// buat object baru dari kelas person
$person1 = new Person("Intan","bengkulu");
// panggil function
$person1->sayHello("Intan");
// panggil self keyword
$person1->info();
```

Penjelasan:

Membuat objek baru dari kelas Person dengan memberikan nilai "Intan" dan "bengkulu" sebagai parameter konstruktor.Lalu Memanggil metode sayHello pada objek $person1 dengan memberikan parameter "Intan". Metode ini akan mencetak pesan sapaan menggunakan nilai yang diberikan. Memanggil metode info pada objek $person1 menggunakan kata kunci self. Dalam konteks ini, self merujuk pada kelas yang saat ini

sedang dieksekusi, yaitu kelas Person. Metode ini akan mencetak informasi yang didefinisikan di dalam metode info.

21. thisKeyword.php

```php
<?php
// import data/person.php
require_once "data/person.php";
// buat object dari kelas person
$intan = new Person("Intan", "bengkulu");
// tambahkan value nama di object
$intan->nama = "Intan";
// panggil function sayHelloNull dengan parameter
$intan->sayHelloNull("kack");
// buat object dari kelas person
$budiarty = new Person("budiarty", "bengkulu");
// tambahkan value nama di object
$budiarty->nama = "bengkulu";
// panggil function sayHelloNull dengan parameter null
$budiarty->sayHelloNull(null);

```

Penjelasan:

Membuat objek baru dari kelas Person dengan memberikan nilai "Intan" dan "bengkulu" sebagai parameter konstruktor. Menambahkan atau mengubah nilai properti nama pada objek $intan menjadi "Intan".Memanggil metode sayHelloNull pada objek $intan dengan memberikan parameter "kack". Metode ini akan mencetak pesan sapaan menggunakan nama yang diberikan. Membuat objek baru dari kelas Person dengan memberikan nilai "budiarty" dan "bengkulu" sebagai parameter konstruktor. Menambahkan atau mengubah nilai properti nama pada objek $budiarty menjadi "budiarty".Memanggil metode sayHelloNull pada objek $budiarty dengan memberikan parameter null. Metode ini akan mencetak pesan sapaan tanpa menyertakan nama, karena parameter yang diberikan adalah null.

22. Visibility.php

```php
<?php
require_once "data/Product.php";

$product = new Product("Apple", 20000);

// tampilkan product get name
// tampilkan product get price

$dummy = new ProductDummy("Dummy", 1000);
$dummy->info();
```

Penjelasan:

Membuat objek $product dari kelas Product dengan memberikan nilai "Apple" dan 20000 sebagai parameter konstruktor. Menggunakan metode getName dan getPrice pada objek $product untuk mendapatkan dan menampilkan nama dan harga produk. Membuat objek $dummy dari kelas ProductDummy dengan memberikan nilai "Dummy" dan 1000 sebagai parameter konstruktor.Memanggil metode info pada objek $dummy untuk menampilkan informasi produk menggunakan metode yang didefinisikan di dalam kelas ProductDummy.