

LAPORAN
WORKSHOP PEMROGRAMAN WEB
MINGGU 6
ACARA 11 dan 12



Oleh :
FULAY FILLAH
E32231532
GOLONGAN C

PROGRAM STUDI TEKNIK KOMPUTER
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI JEMBER
2024

ACARA 11

Materi Pembelajaran : Interface

pada PHP Acara Praktikum/Praktik: 6/1

Tempat : Daring/Luring Politeknik Negeri Jember

Alokasi Waktu : 2 x 100 menit

a. Capaian Pembelajaran Mata Kuliah (CPMK)

- Mahasiswa mampu memahami dan mengimplementasikan PHP dalam pengembangan WEB
- Mahasiswa mampu memahami dan mengimplementasikan Interface pada PHP

b. Penilaian Bertahap BNSP (*Skill Passport*)

Nama Skema Sertifikasi: Pengembang Web/Web Developer (SKM-496-028)

No	Kode Unit	Nama Unit Kompetensi	Elemen Kompetensi
1	J.620100.017.02	Mengimplementasikan Pemrograman Terstruktur	<ul style="list-style-type: none">• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Membuat program untuk akses file• Membuat program untuk akses file
2	J.620100.018.02	Mengimplementasikan Pemrograman Berorientasi Objek	<ul style="list-style-type: none">• Membuat program berorientasi objek dengan memanfaatkan class• Menggunakan tipe data dan control program pada metode atau operasi dari suatu kelas• Membuat program dengan konsep berbasis objek• Membuat program object oriented dengan interface dan paket• Mengkompilasi Program

c. Indikator Penilaian

- Ketepatan memahami dan mengimplementasikan PHP dalam pengembangan WEB
- Ketepatan memahami dan mengimplementasikan Interface pada PHP

d. Dasar Teori

Secara sederhana, Object Interface adalah sebuah ‘kontrak’ atau perjanjian implementasi method. Bagi class yang menggunakan object interface, class tersebut harus mengimplementasikan ulang seluruh method yang ada di dalam interface. Dalam pemrograman objek, penyebutan object interface sering disingkat dengan ‘Interface’ saja. Jika anda telah mempelajari abstract class, maka interface bisa dikatakan sebagai bentuk lain dari abstract class. Walaupun secara konsep teoritis dan tujuan penggunaannya berbeda. Sama seperti abstract class, interface juga hanya berisi signature dari method, yakni hanya nama method dan parameternya saja (jika ada). Isi dari method akan dibuat ulang di dalam class yang menggunakan interface. Jika kita menganggap abstract class sebagai ‘kerangka’ atau ‘blue print’ dari class-class lain, maka interface adalah implementasi method yang harus ‘tersedia’ dalam sebuah objek. Interface tidak bisa disebut sebagai ‘kerangka’ class. Bentuk umum dari penulisan interface sebagai berikut:

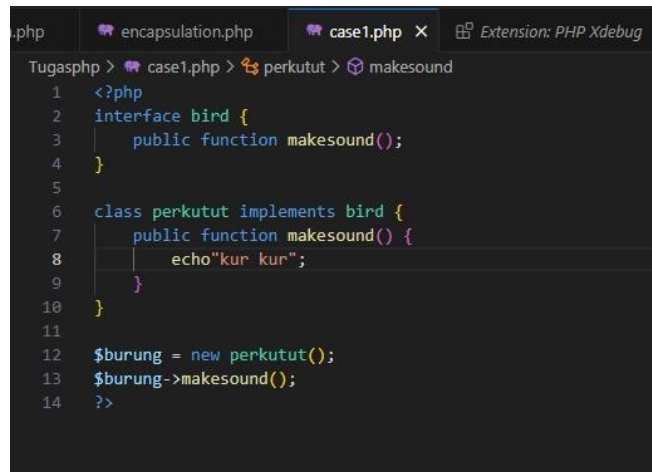
e. Alat dan Bahan

- Client Server: xampp
- Text Editor: Visual Code/Sublime Text 3
- Management Storage: Git dan Github
- Tools Dependency Manager Multiplatform: Composer
- Kertas A4 / Folio Bergaris
- Pulpen

f. Prosedur Kerja

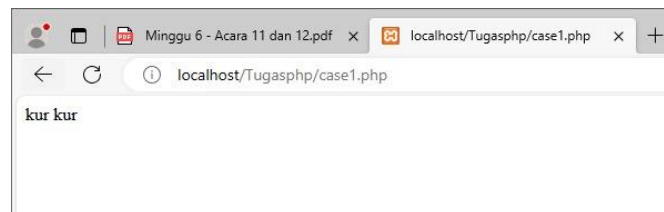
Cobalah kode program dibawah ini secara individu, kemudian buatlah laporan dari kode program yang telah dicoba. Hal yang dilaporkan mengenai fungsi setiap baris dari kode program tersebut dan Analisa hasil output dari kode program tersebut

Case 1 :



```
1 <?php
2 interface bird {
3     public function makesound();
4 }
5
6 class perkutut implements bird {
7     public function makesound() {
8         echo "kur kur";
9     }
10 }
11
12 $burung = new perkutut();
13 $burung->makesound();
14 ?>
```

Hasil Case 1 :



kur kur

Mari kita bahas setiap baris kode dan hasil outputnya:

1. `<?php`: Ini adalah tag pembuka untuk menandai awal dari kode PHP.
2. `interface Bird { public function makeSound(); }`: Ini mendefinisikan sebuah antarmuka bernama `Bird` yang memiliki satu metode abstrak yaitu `makeSound()`. Antarmuka ini menetapkan kontrak bahwa setiap kelas yang mengimplementasikan `Bird` harus menyediakan implementasi untuk `makeSound()`.
3. `class lakberd implements Bird {`: Ini mendefinisikan sebuah kelas bernama `Perkutut` yang mengimplementasikan antarmuka `Bird`.
4. `public function makeSound() { echo "Kur Kur"; }`: Ini mendefinisikan metode `makeSound()` dalam kelas `lakberd`. Ketika metode ini dipanggil, itu akan mencetak string "Kur Kur".

5. ``$burung = new lakberd();``: Ini membuat sebuah instance baru dari kelas ``Perkutut`` dan menetakannya ke variabel ``$burung``.
6. ``$burung->makeSound();``: Ini memanggil metode ``makeSound()`` dari objek ``$burung``, yang akan mencetak "Kur Kur" ke output.

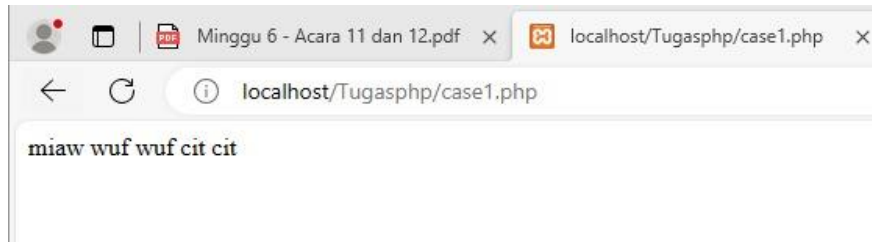
Analisis:

- Kode program ini mendefinisikan sebuah antarmuka ``Bird`` dengan satu metode ``makeSound()``, yang harus diimplementasikan oleh kelas-kelas yang mengimplementasikan antarmuka tersebut.
- Kelas ``lakberd`` diimplementasikan dari antarmuka ``Bird`` dan menyediakan implementasi untuk metode ``makeSound()``, yang hanya mencetak "Kur Kur".
- Ketika objek ``$burung`` dibuat dan metode ``makeSound()`` dipanggil, hasilnya adalah output "Kur Kur", sesuai dengan implementasi metode ``makeSound()`` dalam kelas ``lakberd``.

Case 2 :

```
1 <?php
2 interface animal {
3     public function makesound() ;
4 }
5 class cat implements animal {
6     public function makesound () {
7         echo "miaw ";
8     }
9 }
10 class dog implements animal {
11     public function makesound () {
12         echo "wuf wuf ";
13     }
14 }
15 class mouse implements animal {
16     public function makesound () {
17         echo "cit cit";
18     }
19 }
20 $cat = new cat ();
21 $dog = new dog ();
22 $mouse = new mouse ();
23 $animals = array ($cat,$dog,$mouse);
24
25 foreach ($animals as $animal) {
26     $animal->makesound();
27 }
28 ?>
```

Hasil Case 2 :



Mari kita analisis baris per baris dari kode program PHP yang Anda berikan beserta hasil outputnya:

1. `<?php`: Ini adalah tag pembuka untuk menandai awal dari kode PHP.
2. `// Interface definition`: Ini adalah komentar untuk memberikan penjelasan bahwa definisi antarmuka akan dimulai.
3. `interface Animal { public function makeSound (); }`: Ini mendefinisikan sebuah antarmuka bernama `Animal` dengan satu metode abstrak `makeSound()`. Antarmuka ini menetapkan kontrak bahwa setiap kelas yang mengimplementasikan `Animal` harus menyediakan implementasi untuk `makeSound()`.
4. `// Class definitions`: Ini adalah komentar untuk memberikan penjelasan bahwa definisi kelas akan dimulai.
5. `class Cat implements Animal { public function makeSound() { echo "Meow"; } }`: Ini mendefinisikan kelas `Cat` yang mengimplementasikan antarmuka `Animal`. Kelas `Cat` menyediakan implementasi untuk metode `makeSound()`, yang mencetak string "Meow".
6. `class Dog implements Animal { public function makeSound() { echo "wuf wuf"; } }`: Ini mendefinisikan kelas `Dog` yang mengimplementasikan antarmuka `Animal`. Kelas `Dog` menyediakan implementasi untuk metode `makeSound()`, yang mencetak string "wuf wuf".
7. `class Mouse implements Animal { public function makeSound() { echo "cit`

cit"; } }': Ini mendefinisikan kelas `Mouse` yang mengimplementasikan antarmuka `Animal`. Kelas `Mouse` menyediakan implementasi untuk metode `makeSound()`, yang mencetak string "cit cit".

8. `// Create a list of animals`: Ini adalah komentar untuk memberikan penjelasan bahwa akan dibuat daftar hewan.

9. `\$cat = new Cat(); \$dog = new Dog(); \$mouse = new Mouse(); \$animals = array (\$cat, \$dog, \$mouse);`: Ini membuat objek dari masing-masing kelas (`Cat`, `Dog`, `Mouse`) dan menyimpannya dalam sebuah array `\$animals`.

10. `// Tell the animals to make a sound`: Ini adalah komentar untuk memberikan penjelasan bahwa akan meminta hewan-hewan tersebut untuk mengeluarkan suara.

11. `foreach (\$animals as \$animal) { \$animal->makeSound(); }`: Ini melakukan iterasi melalui array `\$animals` dan memanggil metode `makeSound()` pada setiap objek dalam array. Oleh karena itu, setiap hewan akan mengeluarkan suara sesuai implementasi yang telah ditentukan pada masing-masing kelas.

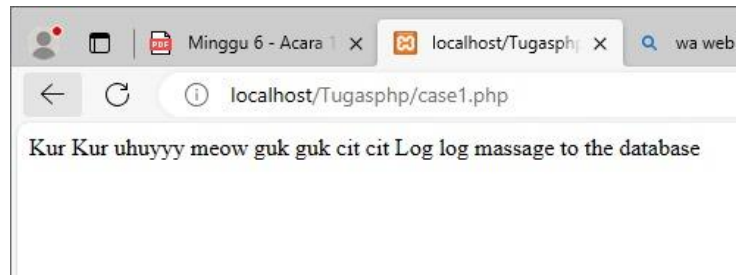
Analisis:

- Kode program ini mendefinisikan sebuah antarmuka `Animal` dengan satu metode `makeSound()`, yang harus diimplementasikan oleh kelas-kelas yang mengimplementasikan antarmuka tersebut.
- Tiga kelas (`Cat`, `Dog`, `Mouse`) diimplementasikan dari antarmuka `Animal`, masing-masing dengan implementasi metode `makeSound()` yang berbeda.
- Objek-objek dari ketiga kelas tersebut dibuat dan disimpan dalam sebuah array.
- Dalam loop `foreach`, setiap objek dalam array dipanggil metode `makeSound()`-nya, sehingga masing-masing objek mengeluarkan suara sesuai dengan kelasnya masing-masing.

Case 3 :

```
1 <?php
2 interface Bird{
3     public function makesound();
4     public function terbang();
5 }
6
7 class Perkutut implements Bird{
8     public function makesound(){
9         echo "Kur Kur\n";
10    }
11    public function terbang(){
12        echo "uhuyyy";
13    }
14 }
15
16 $burung = new Perkutut();
17 $burung->makesound();
18 $burung->terbang();
19 ?>
20
21 <?php
22 interface animal{
23     public function makesound();
24 }
25 class cat implements animal{
26     public function makesound(){
27         echo "meow \n";
28     }
29 }
30 class dog implements animal{
31     public function makesound(){
32         echo "guk guk \n";
33     }
34 }
35 class mouse implements animal{
36     public function makesound(){
37         echo "cit cit \n";
38     }
39 }
40 $cat = new cat;
41 $dog = new dog;
42 $mouse = new mouse;
43 $animals = array ($cat, $dog, $mouse);
44
45 foreach ($animals as $animal) {
46     $animal->makesound();
47 }
48 ?>
49
50 <?php
51 interface logger{
52     public function log($message);
53 }
54 class filelogger implements logger{
55     private $handle;
56     private $logfile;
57
58     public function __construct($filename, $mode = "a"){
59         $this->logfile = $filename;
60         $this->handle = fopen($filename, $mode)
61         or die('Could not open the log file');
62     }
63     public function log($message){
64         $message = date ('F j, Y, g:i a'). ': ' . $message . "\n";
65         fwrite ($this->handle, $message);
66     }
67     public function __destruct(){
68         if ($this->handle){
69             fclose($this->handle);
70         }
71     }
72 }
73 class databaselogger implements logger{
74     public function log($message){
75         echo sprintf("Log %s to the database \n", $message);
76     }
77 }
78
79 $logger = new filelogger('./log.txt', 'w');
80 $logger->log('PHP interface is awesome');
81
82 $loggers=[
83     new filelogger('./loh.txt'),
84     new databaselogger()
85 ];
86 foreach ($loggers as $logger){
87     $logger->log('log message');
88 }
89 ?>
```


Hasil Case 3



Mari kita analisis baris per baris dari kode program PHP yang Anda berikan beserta hasil outputnya:

1. `<?php`: Ini adalah tag pembuka untuk menandai awal dari kode PHP.
2. `interface logger{ public function log($message); }`: Ini mendefinisikan sebuah antarmuka `logger` dengan satu metode `log()`.
3. `class filelogger implements logger{ ... }`: Ini mendefinisikan kelas `filelogger` yang mengimplementasikan antarmuka `logger`. Kelas ini bertanggung jawab untuk mencatat log ke dalam sebuah file.
4. Di dalam kelas `filelogger`:
 - `private $handle;` dan `private $logfile;`: Mendefinisikan properti untuk menangani file dan nama file log.
 - `public function __construct($filename, $mode = "a"){ ... }`: Ini adalah konstruktor kelas `filelogger` yang membuka file log untuk ditulis. Jika file tidak bisa dibuka, program akan menghasilkan pesan kesalahan dan berhenti.
 - `public function log($message){ ... }`: Implementasi metode `log()` dari antarmuka `logger`, yang mencatat pesan log ke dalam file. Pesan log yang dicatat juga dilengkapi dengan tanggal dan waktu.
 - `public function __destruct(){ ... }`: Ini adalah destruktur kelas `filelogger` yang menutup file log jika masih terbuka.
5. `class databaselogger implements logger{ ... }`: Ini mendefinisikan kelas

``databaselogger`` yang juga mengimplementasikan antarmuka ``logger``. Kelas ini bertanggung jawab untuk mencatat log ke dalam database.

6. Di dalam kelas ``databaselogger``:

- ``public function log($message){ ... }``: Implementasi metode ``log()`` dari antarmuka ``logger``, yang mencetak pesan log ke layar dengan format tertentu.

7. ``$logger = new filelogger('./log.txt', 'w');``: Membuat objek dari kelas ``filelogger`` dengan nama file log `"/log.txt"` dan mode penulisan `"w"`.

8. ``$logger->log('PHP interface is awesome');``: Memanggil metode ``log()`` pada objek ``$logger``, sehingga akan dicatat pesan log `"PHP interface is awesome"` ke dalam file log `"/log.txt"`.

9. ``$loggers=[...]``: Membuat sebuah array yang berisi objek-objek logger, termasuk sebuah ``filelogger`` dan sebuah ``databaselogger``.

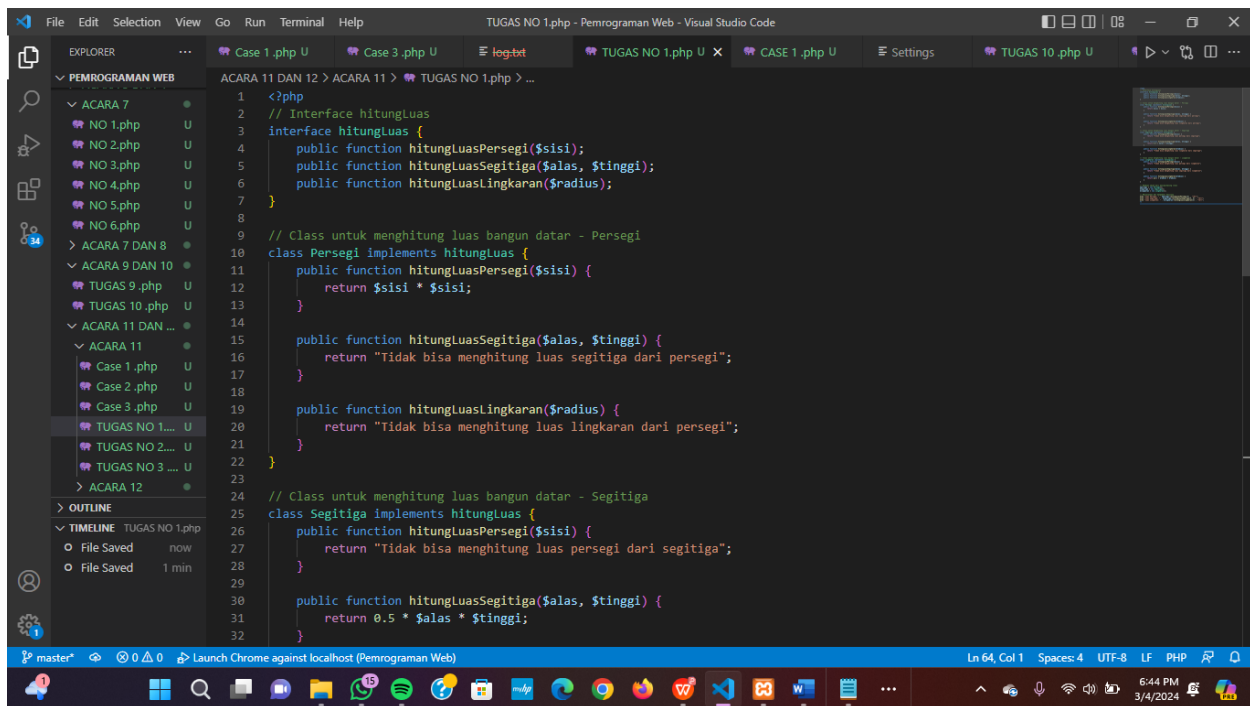
10. ``foreach ($loggers as $logger){ $logger->log('log message'); }``: Melakukan iterasi melalui array ``$loggers`` dan memanggil metode ``log()`` pada setiap objek logger. Pesan log `"log message"` akan dicatat oleh setiap objek logger sesuai dengan implementasinya masing-masing.

Analisis:

- Kode program ini menggunakan konsep antarmuka (``interface``) untuk mengimplementasikan polimorfisme, di mana kelas-kelas yang berbeda dapat menggunakan metode yang sama namun dengan implementasi yang berbeda.
- Dalam contoh ini, terdapat dua kelas yang mengimplementasikan antarmuka ``logger``: ``filelogger`` dan ``databaselogger``.
- Objek ``filelogger`` bertanggung jawab untuk mencatat log ke dalam file, sementara ``databaselogger`` mencatat log ke dalam database.
- Melalui loop ``foreach``, setiap objek logger dipanggil untuk mencatat pesan log `"log message"`, dan output dari setiap pemanggilan metode ``log()`` sesuai dengan implementasi masing-masing kelas logger.

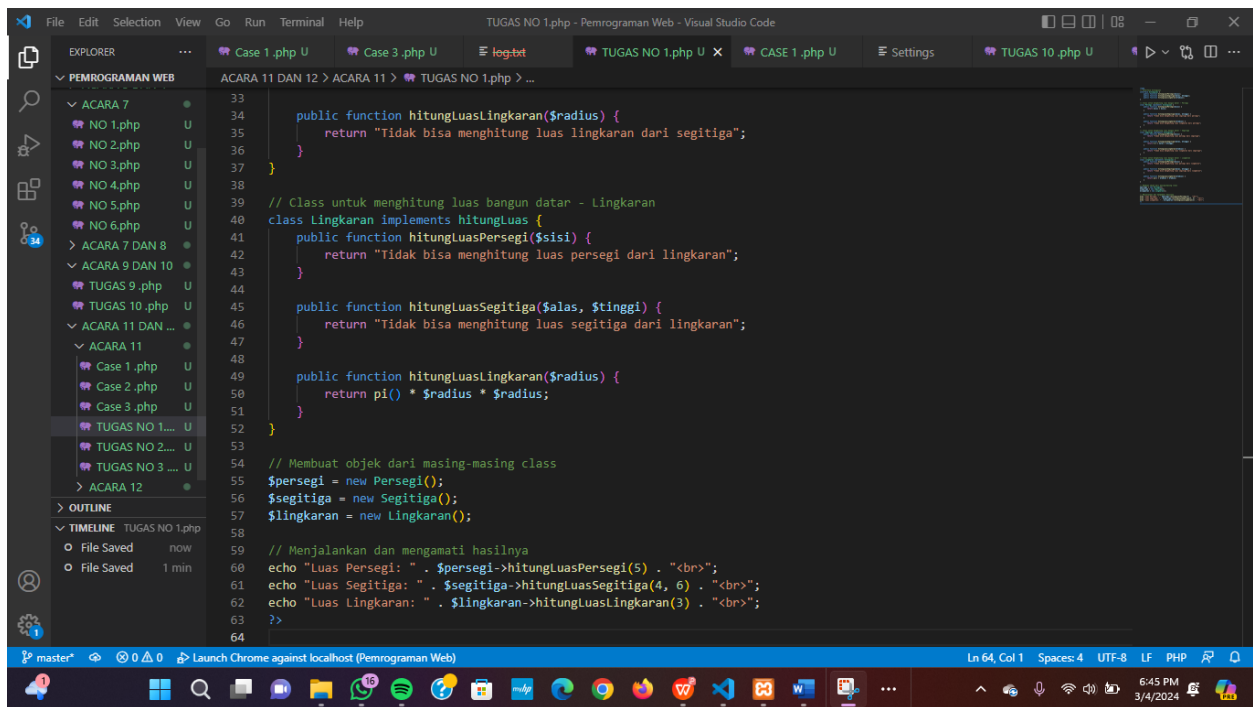
Berdasarkan case diatas kerjakan Latihan dibawah ini secara individu dan buatlah laporan dengan ketentuan sebagai berikut:

1. Buatlah interface yang bernama hitungLuas dengan property bernama 'sisi' dan berisi 3 methods yaitu fungsi hitungLuasPersegi(), hitungLuasSegitiga() dan hitungLuasLingkaran().
2. Buatlah 3 class untuk menghitung luasbangun datar yang mengimplementasikan interface tersebut.
3. Buatlah object dari masing-masing class, kemudian jalankan dan amati hasilnya.

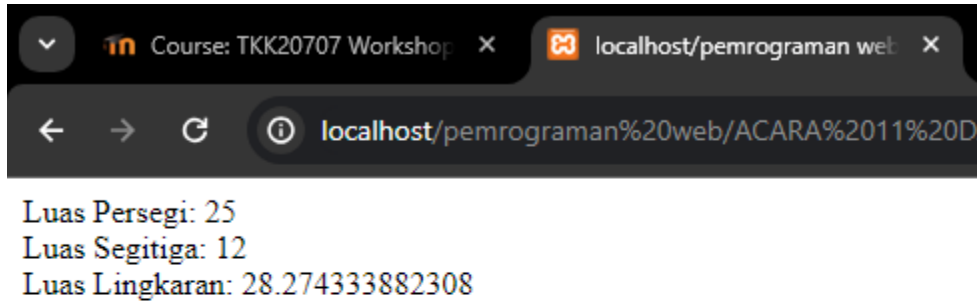


The screenshot shows a Visual Studio Code editor window with the following code:

```
1 <?php
2 // Interface hitungLuas
3 interface hitungLuas {
4     public function hitungLuasPersegi($sisi);
5     public function hitungLuasSegitiga($alas, $tinggi);
6     public function hitungLuasLingkaran($radius);
7 }
8
9 // Class untuk menghitung luas bangun datar - Persegi
10 class Persegi implements hitungLuas {
11     public function hitungLuasPersegi($sisi) {
12         return $sisi * $sisi;
13     }
14
15     public function hitungLuasSegitiga($alas, $tinggi) {
16         return "Tidak bisa menghitung luas segitiga dari persegi";
17     }
18
19     public function hitungLuasLingkaran($radius) {
20         return "Tidak bisa menghitung luas lingkaran dari persegi";
21     }
22 }
23
24 // Class untuk menghitung luas bangun datar - Segitiga
25 class Segitiga implements hitungLuas {
26     public function hitungLuasPersegi($sisi) {
27         return "Tidak bisa menghitung luas persegi dari segitiga";
28     }
29
30     public function hitungLuasSegitiga($alas, $tinggi) {
31         return 0.5 * $alas * $tinggi;
32     }
33 }
```



Hasil



Hasil yang diamati dari kode tersebut adalah:

- Luas Persegi dengan sisi 5 adalah 25.
- Luas Segitiga dengan alas 4 dan tinggi 6 adalah 12.
- Luas Lingkaran dengan jari-jari 3 adalah sekitar 28.274333882308.

Setiap objek dari kelas `Persegi`, `Segitiga`, dan `Lingkaran` mengimplementasikan metode yang sesuai untuk menghitung luas bangun datar. Dengan menggunakan objek dari masing-masing kelas tersebut dan memanggil metode sesuai dengan jenis bangun datarnya, kita dapat menghitung luas dengan benar sesuai dengan formula yang telah ditentukan.

ACARA 12

Materi Pembelajaran : Polymorphism
pada PHP Acara Praktikum/Praktik : 6/2
Tempat : Daring/Luring Politeknik Negeri Jember
Alokasi Waktu : 2 x 100 menit

a. Capaian Pembelajaran Mata Kuliah (CPMK)

- Mahasiswa mampu memahami dan mengimplementasikan PHP dalam pengembangan WEB
- Mahasiswa mampu memahami dan mengimplementasikan Polymorphism pada PHP

b. Penilaian Bertahap BNSP (*Skill Passport*)

Nama Skema Sertifikasi: Pengembang Web/Web Developer (SKM-496-028)

No	Kode Unit	Nama Unit Kompetensi	Elemen Kompetensi
1	J.620100.017.02	Mengimplementasikan Pemrograman Terstruktur	<ul style="list-style-type: none">• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Menggunakan tipe data dan control program• Membuat program untuk akses file• Membuat program untuk akses file
2	J.620100.018.02	Mengimplementasikan Pemrograman Berorientasi Objek	<ul style="list-style-type: none">• Membuat program berorientasi objek dengan memanfaatkan class• Menggunakan tipe data dan control program pada metode atau operasi dari suatu kelas• Membuat program dengan konsep berbasis objek• Membuat program object oriented dengan interface dan paket• Mengkompilasi Program

c. Indikator Penilaian

- Ketepatan memahami dan mengimplementasikan PHP dalam pengembangan WEB
- Ketepatan memahami dan mengimplementasikan Polymorphism pada PHP

d. Dasar Teori

Dari segi bahasa, Polimorfisme (bahasa inggris: Polymorphism) berasal dari dua kata bahasa latin yakni poly dan morph. Poly berarti banyak, dan morph berarti bentuk. Polimorfisme berarti banyak bentuk (wikipedia). Di dalam pemrograman objek, polimorfisme adalah konsep dimana terdapat banyak class yang memiliki signature method yang sama. Implementasi dari method-method tersebut diserahkan kepada tiap class, akan tetapi cara pemanggilan method harus sama. Agar kita dapat ‘memaksakan’ signature method yang sama pada banyak class, class tersebut harus diturunkan dari sebuah abstract class atau object interface.

e. Alat dan Bahan

- Client Server: xampp
- Text Editor: Visual Code/Sublime Text 3
- Management Storage: Git dan Github
- Tools Dependency Manager Multiplatform: Composer
- Kertas A4 / Folio Bergaris
- Pulpen

f. Prosedur Kerja

Cobalah kode program dibawah ini secara individu, kemudian buatlah laporan dari kode program yang telah dicoba. Hal yang dilaporkan mengenai fungsi setiap baris dari kode program tersebut dan Analisa hasil output dari kode program tersebut.

Case 1 :

The screenshot shows the Visual Studio Code editor with the file explorer on the left displaying a project named 'PEMROGRAMAN WEB'. The file 'CASE 1.php' is selected. The main editor window shows the following PHP code:

```
1 <?php
2 // membuat interface Tanah
3 interface Tanah
4 {
5     public function hitungLuas();
6 }
7
8 // membuat abstract class Bibit
9 abstract class Bibit
10 {
11     abstract public function ditanami();
12 }
13
14 // extends Bibit dan implements Tanah
15 class Lingkaran extends Bibit implements Tanah
16 {
17     private $jarIjari;
18     private $pi = 3.14;
19
20     public function __construct($jarIjari) {
21         $this->jarIjari = $jarIjari;
22     }
23
24     // menghitung luas tanah berbentuk lingkaran
25     // implements method dari interface tanah
26     public function hitungLuas() {
27         return $this->jarIjari * $this->jarIjari * $this->pi;
28     }
29
30     // tanah ditanami Kopi
31     // extends method dari abstract class Bibit
32     public function ditanami() {
```

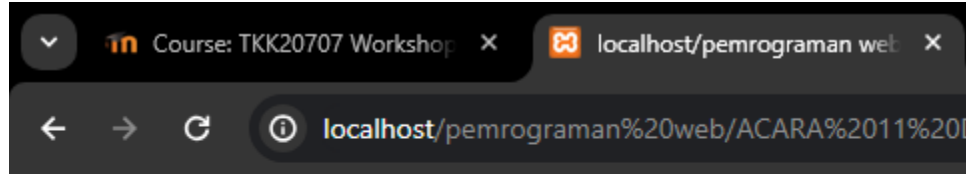
The status bar at the bottom indicates the cursor is at line 68, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the PHP file type.

The screenshot shows the Visual Studio Code editor with the file explorer on the left. The file 'CASE 1.php' is selected. The main editor window shows the following PHP code:

```
36
37 class PersegiPanjang extends Bibit implements Tanah {
38     private $panjang;
39     private $lebar;
40
41     public function __construct($panjang, $lebar) {
42         $this->panjang = $panjang;
43         $this->lebar = $lebar;
44     }
45
46     // menghitung luas tanah berbentuk persegi panjang
47     // implements method dari interface tanah
48     public function hitungLuas() {
49         return $this->panjang * $this->lebar;
50     }
51
52     // tanah ditanama Padi
53     // Extends method dari abstract class Bibit
54     public function ditanami() {
55         return "ditanami Padi";
56     }
57 }
58
59 $tanahAgus = new Lingkaran(3);
60 $tanahChandra = new PersegiPanjang(3,4);
61
62 echo "Tanah Agus seluas " . $tanahAgus->hitungLuas() . " m2 \n";
63 echo $tanahAgus->ditanami() . "\n \n" . "<br>";
64
65 echo "Tanah Chandra seluas " . $tanahChandra->hitungLuas() . " m2 \n";
66 echo $tanahChandra->ditanami()."\n";
67 ?>
```

The status bar at the bottom indicates the cursor is at line 68, column 1, with 4 spaces, UTF-8 encoding, CRLF line endings, and the PHP file type.

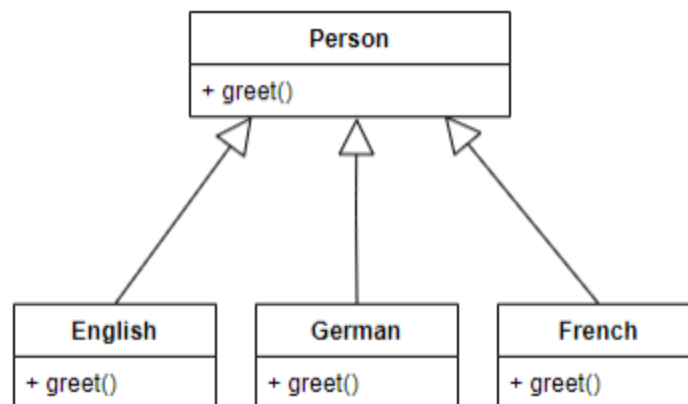
Hasil case :



Tanah Agus seluas 28.26 m2 ditanami Kopi
Tanah Chandra seluas 12 m2 ditanami Padi

TUGAS

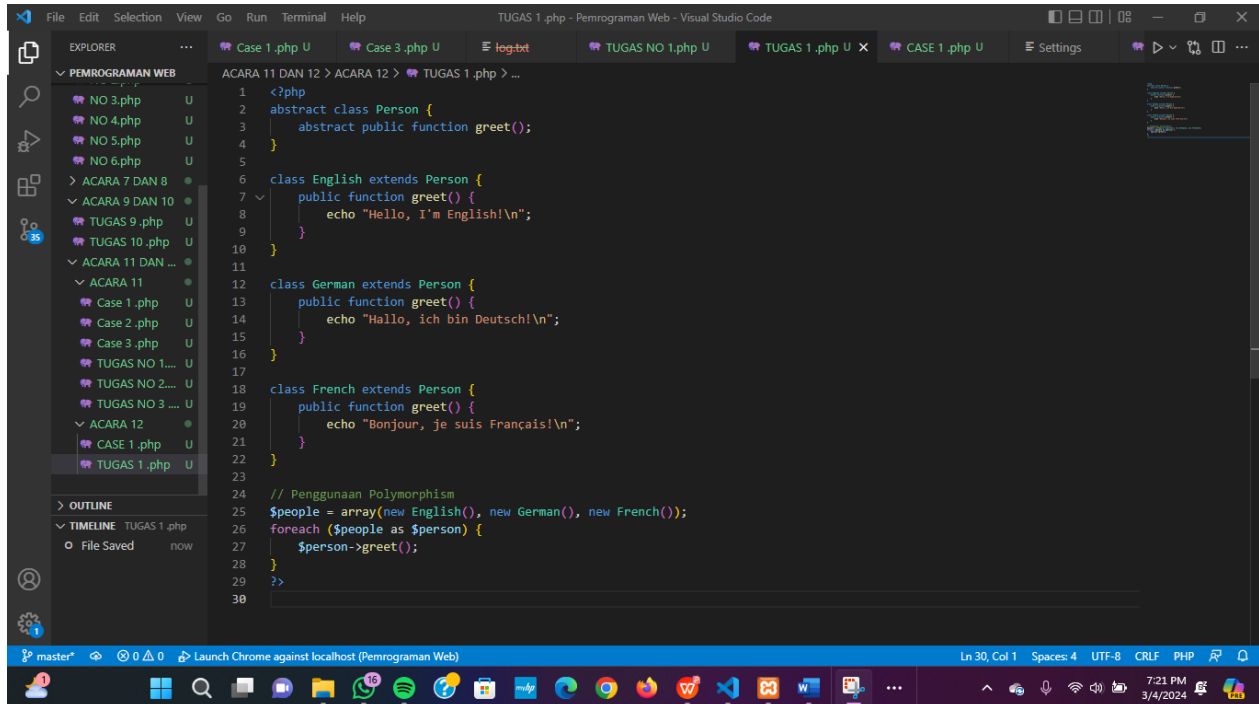
1. Berdasarkan diagram dibawah ini buatlah sebuah contoh polymorphism



2. Dimana Person adalah abstract class yang memiliki abstract fungsi, masing – masing fungsi greet mencetak nilai yang berbeda.
3. Buatlah sebuah contoh lain polymorphism dimana didalamnya terdapat minimal 1 abstract class, 1 interface dan 3 class lain yang merupakan extends dari class abstract atau interface.

Jawab

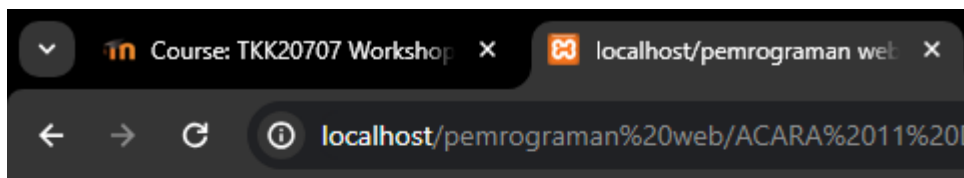
1. Program



The screenshot shows the Visual Studio Code editor with a PHP file named 'TUGAS 1.php'. The code implements an abstract class 'Person' with a 'greet()' method. Three subclasses, 'English', 'German', and 'French', extend 'Person' and override the 'greet()' method to output their respective greetings. The main code creates an array of these objects and iterates through them, calling 'greet()' on each.

```
1 <?php
2 abstract class Person {
3     abstract public function greet();
4 }
5
6 class English extends Person {
7     public function greet() {
8         echo "Hello, I'm English!\n";
9     }
10 }
11
12 class German extends Person {
13     public function greet() {
14         echo "Hallo, ich bin Deutsch!\n";
15     }
16 }
17
18 class French extends Person {
19     public function greet() {
20         echo "Bonjour, je suis Français!\n";
21     }
22 }
23
24 // Penggunaan Polymorphism
25 $people = array(new English(), new German(), new French());
26 foreach ($people as $person) {
27     $person->greet();
28 }
29 ?>
```

Hasil



Hello, I'm English! Hallo, ich bin Deutsch! Bonjour, je suis Français!

2. Dalam contoh program di atas

```
ACARA 11 DAN 12 > ACARA 12 > TUGAS 1.php > ...
1  <?php
2  ∨ abstract class Person {
3      |     abstract public function greet();
4  }
5
6  ∨ class English extends Person {
7  ∨      |     public function greet() {
8      |         |     echo "Hello, I'm English!\n";
9      |     }
10 }
11
12 ∨ class German extends Person {
13 ∨      |     public function greet() {
14      |         |     echo "Hallo, ich bin Deutsch!\n";
15      |     }
16 }
17
18 ∨ class French extends Person {
19 ∨      |     public function greet() {
20      |         |     echo "Bonjour, je suis Français!\n";
21      |     }
22 }
23
24 // Penggunaan Polymorphism
25 $people = array(new English(), new German(), new French());
26 ∨ foreach ($people as $person) {
27     |     $person->greet();
28 }
29 ?>
30
```

- Person adalah kelas abstrak dengan metode 'greet()' yang bersifat abstrak.
- 'English', 'German', dan 'French' adalah kelas turunan yang mengimplementasikan metode 'greet()' dengan pesan yang berbeda dalam bahasa yang berbeda.
- Kemudian, kita membuat array dari objek-objek 'English', 'German', dan 'French', dan menggunakan polimorfisme untuk memanggil metode

'greet()' dari setiap objek.

Output dari contoh di atas adalah:

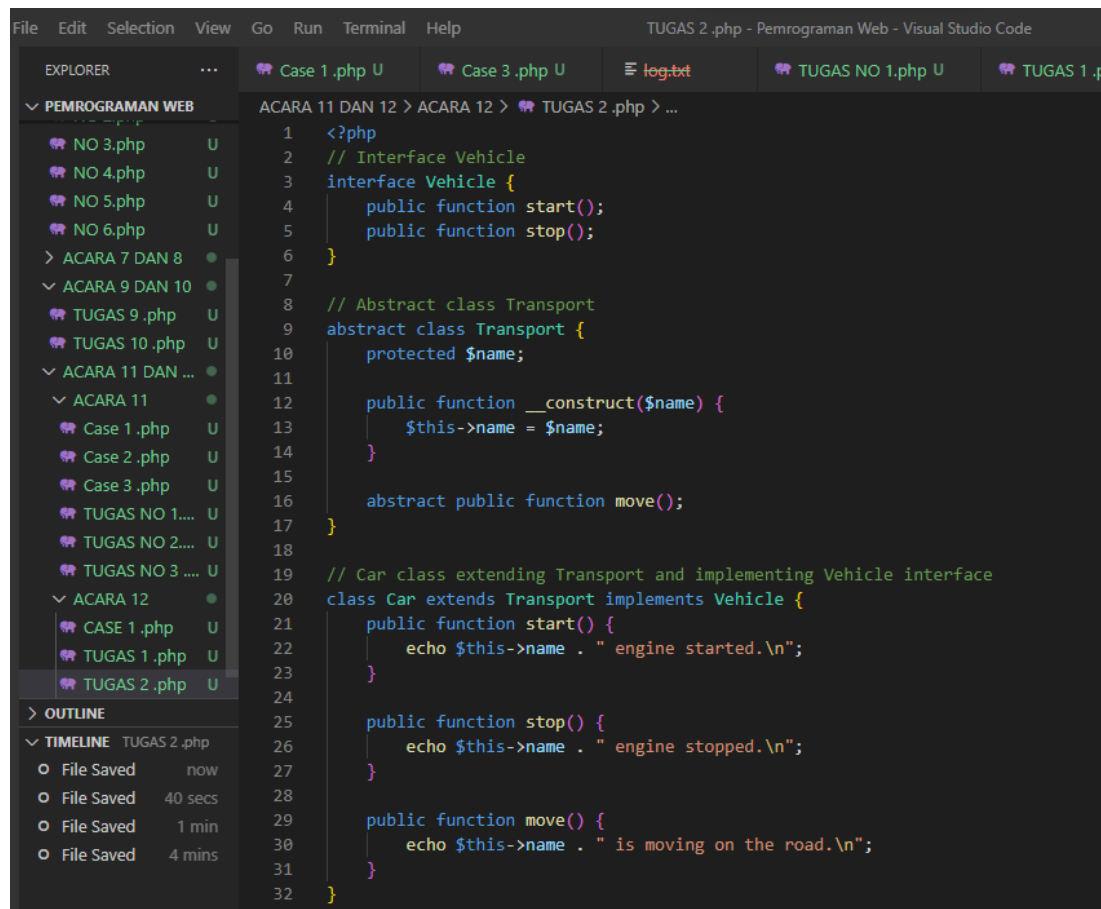
Hello, I'm English!

Hallo, ich bin Deutsch!

Bonjour, je suis Français!

Dengan demikian, kita telah berhasil menerapkan polimorfisme dengan menggunakan kelas abstrak 'Person' yang memiliki metode abstrak 'greet()' yang diimplementasikan dengan cara yang berbeda dalam setiap turunannya.

3. Program



The screenshot shows the Visual Studio Code interface with a PHP file named 'TUGAS 2.php' open. The code defines an interface 'Vehicle' and an abstract class 'Transport' that implements 'Vehicle'. The 'Transport' class has a protected property '\$name' and methods '__construct()', 'move()', and 'stop()'. The 'Car' class extends 'Transport' and implements the 'Vehicle' interface, providing concrete implementations for 'start()', 'stop()', and 'move()'. The 'Car' class's 'start()' method echoes the car's name followed by 'engine started.\n', 'stop()' echoes 'engine stopped.\n', and 'move()' echoes 'is moving on the road.\n'.

```
1 <?php
2 // Interface Vehicle
3 interface Vehicle {
4     public function start();
5     public function stop();
6 }
7
8 // Abstract class Transport
9 abstract class Transport {
10     protected $name;
11
12     public function __construct($name) {
13         $this->name = $name;
14     }
15
16     abstract public function move();
17 }
18
19 // Car class extending Transport and implementing Vehicle interface
20 class Car extends Transport implements Vehicle {
21     public function start() {
22         echo $this->name . " engine started.\n";
23     }
24
25     public function stop() {
26         echo $this->name . " engine stopped.\n";
27     }
28
29     public function move() {
30         echo $this->name . " is moving on the road.\n";
31     }
32 }
```

```
EXPLORER    ...    Case 1 .php U    Case 3 .php U    log.txt    TUGAS NO 1.php U    TUGAS 1 .php U    TUGAS 2 .php U

PEMROGRAMAN WEB
  NO 3.php U
  NO 4.php U
  NO 5.php U
  NO 6.php U
  > ACARA 7 DAN 8
  ACARA 9 DAN 10
    TUGAS 9 .php U
    TUGAS 10 .php U
  ACARA 11 DAN 12
    ACARA 11
      Case 1 .php U
      Case 2 .php U
      Case 3 .php U
      TUGAS NO 1.... U
      TUGAS NO 2.... U
      TUGAS NO 3 .... U
    ACARA 12
      CASE 1 .php U
      TUGAS 1 .php U
      TUGAS 2 .php U

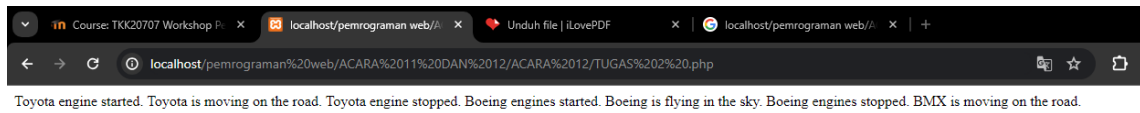
  > OUTLINE
  TIMELINE TUGAS 2 .php
    File Saved now
    File Saved 40 secs
    File Saved 1 min
    File Saved 4 mins

ACARA 11 DAN 12 > ACARA 12 > TUGAS 2 .php > ...
33
34 // Plane class extending Transport and implementing Vehicle interface
35 class Plane extends Transport implements Vehicle {
36     public function start() {
37         echo $this->name . " engines started.\n";
38     }
39
40     public function stop() {
41         echo $this->name . " engines stopped.\n";
42     }
43
44     public function move() {
45         echo $this->name . " is flying in the sky.\n";
46     }
47 }
48
49 // Bicycle class extending Transport
50 class Bicycle extends Transport {
51     public function move() {
52         echo $this->name . " is moving on the road.\n";
53     }
54 }
55
56 // Usage of Polymorphism
57 $vehicles = array(new Car("Toyota"), new Plane("Boeing"), new Bicycle("BMX"));
58 foreach ($vehicles as $vehicle) {
59     if ($vehicle instanceof Vehicle) {
60         $vehicle->start();
61         $vehicle->move();
62         $vehicle->stop();
63         echo "\n";
64     } else {
```

```
ACARA 11 DAN 12 > ACARA 12 > TUGAS 2 .php > ...

64     } else {
65         $vehicle->move();
66         echo "\n";
67     }
68 }
69 ?>
70
```

Hasil nya



Dalam contoh ini:

- 'Vehicle' adalah antarmuka yang mendefinisikan metode 'start()' dan 'stop()'.
- 'Transport' adalah kelas abstrak yang memiliki metode abstrak 'move()'.
- 'Car' dan 'Plane' adalah kelas turunan dari 'Transport' yang mengimplementasikan 'metode 'move()' dengan perilaku yang sesuai untuk kendaraan darat dan udara, serta mengimplementasikan metode dari antarmuka Vehicle untuk memulai dan menghentikan mesin.
- 'Bicycle' adalah kelas turunan dari 'Transport' yang hanya mengimplementasikan metode 'move()' untuk sepeda.
- Dalam penggunaan polimorfisme, kita membuat array dari berbagai objek kendaraan dan menggunakan metode-metode yang sesuai untuk memulai, bergerak, dan menghentikan masing-masing kendaraan. Jika objek adalah instance dari 'Vehicle', maka akan memanggil metode 'start()' dan 'stop()' sebelum dan setelah memanggil 'move()'.