**Nama : Venti Weliana**

**Prodi : TKKA**

**NIM : E32200069**

**Rangkuman**

1. **Pengetian PHP**

**PHP Adalah** bahasa scripting server-side, Bahasa pemrograman yang digunakan untuk mengembangkan situs web statis atau situs web dinamis atau aplikasi Web. PHP singkatan dari Hypertext Pre-processor.

**-PHP OOP**

Pemograman prosedural adalah tentang menulis prosedur atau fungsi yang melakukan operasi pada data, sementara pemograman yang objectivitas adalah tentang menciptakan objek yang berisi data dan fungsi.

1. **PHP Classes dan object**

Sebuah kelas menggambarkan isi benda benda yang miliknya itu menggambarkan bidang data dan mendefinisikan operasinya sedangkan objek yaitu elemen atau contoh dari kelas dan objek memiliki perilaku dari kelas.

Contoh syntaxnya :

<?php  
class Fruit {  
  // code goes here...  
}  
?>

1. **PHP Constructor**

Konstruktor dalam PHP adalah jenis fungsi khusus dari suatu kelas yang secara otomatis dieksekusi sebagai objek apa pun dari kelas itu yang dibuat atau dibuat instance-nya. Pembuat juga disebut fungsi ajaib karena dalam PHP, metode sulap biasanya dimulai dengan dua karakter garis bawah.

Contoh:

<?php  
class Fruit {  
  public $name;  
  public $color;  
  
  function \_\_construct($name) {  
    $this->name = $name;  
  }  
  function get\_name() {  
    return $this->name;  
  }  
}  
  
$apple = new Fruit("Apple");  
echo $apple->get\_name();  
?>

1. **PHP Descruptor**

Destruktor dipanggil ketika objek dihancurkan atau skrip dihentikan atau keluar.Jika Anda membuat fungsi \_\_destruct (), PHP secara otomatis akan memanggil fungsi ini di akhir skrip.

Contoh :

<?php  
class Fruit {  
  public $name;  
  public $color;  
  
  function \_\_construct($name) {  
    $this->name = $name;  
  }  
  function \_\_destruct() {  
    echo "The fruit is {$this->name}.";  
  }  
}  
  
$apple = new Fruit("Apple

1. **PHP Access Modifier**

Properti dan metode dapat memiliki pengubah akses yang mengontrol di mana mereka dapat diakses.ada tiga pengubah akses:

publik - properti atau metode dapat diakses dari mana saja. Ini default

protected - properti atau metode dapat diakses di dalam kelas dan oleh kelas yang diturunkan dari kelas itu

private - properti atau metode HANYA dapat diakses di dalam kelas

contoh : set\_color() or the set\_weight() function it will result in a fatal error

**6. Php inheritance**

PHP - Apa itu Warisan?

Pewarisan dalam OOP = Ketika kelas diturunkan dari kelas lain. Kelas anak akan mewarisi semua properti dan metode publik dan dilindungi dari kelas induk. Selain itu, ia dapat memiliki properti dan metodenya sendiri. Kelas yang diwariskan ditentukan dengan menggunakan kata kunci extends.

Mari kita lihat contohnya:

<?php  
class Fruit {  
  public $name;  
  public $color;  
  public function \_\_construct($name, $color) {  
    $this->name = $name;  
    $this->color = $color;  
  }  
  public function intro() {  
    echo "The fruit is {$this->name} and the color is {$this->color}.";  
  }  
}  
  
// Strawberry is inherited from Fruit  
class Strawberry extends Fruit {  
  public function message() {  
    echo "Am I a fruit or a berry? ";  
  }  
}  
$strawberry = new Strawberry("Strawberry", "red");  
$strawberry->message();  
$strawberry->intro();  
?>

Contoh Dijelaskan

Kelas Strawberry diwarisi dari kelas Buah. Ini berarti bahwa kelas Strawberry dapat menggunakan properti $ name dan $ color serta metode public \_\_construct () dan intro () dari kelas Fruit karena pewarisan. Kelas Strawberry juga memiliki metodenya sendiri: message (). PHP - Warisan dan Pengubah Akses Terproteksi Pada bab sebelumnya kita telah mempelajari bahwa properti atau metode yang dilindungi dapat diakses di dalam kelas dan oleh kelas yang diturunkan dari kelas itu. Apa artinya?

Mari kita lihat contohnya:

<?php  
class Fruit {  
  public $name;  
  public $color;  
  public function \_\_construct($name, $color) {  
    $this->name = $name;  
    $this->color = $color;  
  }  
  protected function intro() {  
    echo "The fruit is {$this->name} and the color is {$this->color}.";  
  }  
}  
  
class Strawberry extends Fruit {  
  public function message() {  
    echo "Am I a fruit or a berry? ";  
  }  
}  
  
// Try to call all three methods from outside class  
$strawberry = new Strawberry("Strawberry", "red");  // OK. \_\_construct() is public  
$strawberry->message(); // OK. message() is public  
$strawberry->intro(); // ERROR. intro() is protected  
?>

Dalam contoh di atas kita melihat bahwa jika kita mencoba memanggil metode yang dilindungi (intro ()) dari luar kelas, kita akan menerima kesalahan. metode publik akan bekerja dengan baik!

Mari kita lihat contoh lain:

<?php  
class Fruit {  
  public $name;  
  public $color;  
  public function \_\_construct($name, $color) {  
    $this->name = $name;  
    $this->color = $color;  
  }  
  protected function intro() {  
    echo "The fruit is {$this->name} and the color is {$this->color}.";  
  }  
}  
  
class Strawberry extends Fruit {  
  public function message() {  
    echo "Am I a fruit or a berry? ";  
    // Call protected method from within derived class - OK  
    $this -> intro();  
  }  
}  
  
$strawberry = new Strawberry("Strawberry", "red"); // OK. \_\_construct() is public  
$strawberry->message(); // OK. message() is public and it calls intro() (which is protected) from within the derived class  
?>

PHP - Kata Kunci terakhir

Kata kunci terakhir dapat digunakan untuk mencegah pewarisan kelas atau untuk mencegah penimpaan metode.

Contoh berikut menunjukkan cara mencegah pewarisan kelas:

<?php  
final class Fruit {  
  // some code  
}  
  
// will result in error  
class Strawberry extends Fruit {  
  // some code  
}  
?>

Contoh berikut menunjukkan cara mencegah penimpaan metode:

<?php  
class Fruit {  
  final public function intro() {  
    // some code  
  }  
}  
  
class Strawberry extends Fruit {  
  // will result in error  
  public function intro() {  
    // some code  
  }  
}  
?>

**7. PHP OOP - Class Constants**

Konstanta tidak dapat diubah setelah dideklarasikan. Konstanta kelas dapat berguna jika Anda perlu mendefinisikan beberapa data konstan di dalam kelas. Sebuah konstanta kelas dideklarasikan di dalam kelas dengan kata kunci const. Konstanta kelas peka huruf besar kecil. Namun, disarankan untuk memberi nama konstanta dalam huruf besar semua. Kita dapat mengakses sebuah konstanta dari luar kelas dengan menggunakan nama kelas diikuti oleh operator resolusi cakupan (: :) diikuti dengan nama konstanta, seperti di sini:

Contoh:

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
}  
  
echo Goodbye::LEAVING\_MESSAGE;  
?>

Atau, kita dapat mengakses sebuah konstanta dari dalam kelas dengan menggunakan kata kunci self diikuti oleh operator resolusi cakupan (: :) diikuti dengan nama konstanta, seperti di sini:

### Example

<?php  
class Goodbye {  
  const LEAVING\_MESSAGE = "Thank you for visiting W3Schools.com!";  
  public function byebye() {  
    echo self::LEAVING\_MESSAGE;  
  }  
}  
  
$goodbye = new Goodbye();  
$goodbye->byebye();  
?>

# 8. PHP OOP - Abstract Classes

PHP - What are Abstract Classes and Methods?

Kelas dan metode abstrak adalah ketika kelas induk memiliki metode bernama, tetapi membutuhkan kelas anaknya untuk mengisi tugas.

Kelas abstrak adalah kelas yang berisi setidaknya satu metode abstrak. Metode abstrak adalah metode yang dideklarasikan, tetapi tidak diimplementasikan dalam kode. kelas atau metode abstrak ditentukan dengan kata kunci abstrak:

### Syntax

<?php  
abstract class ParentClass {  
  abstract public function someMethod1();  
  abstract public function someMethod2($name, $color);  
  abstract public function someMethod3() : string;  
}  
?>

Saat mewarisi dari kelas abstrak, metode kelas anak harus didefinisikan dengan nama yang sama, dan pengubah akses yang sama atau kurang terbatas. Jadi, jika metode abstrak didefinisikan sebagai dilindungi, metode kelas anak harus didefinisikan sebagai dilindungi atau publik, tetapi tidak pribadi. Selain itu, jenis dan jumlah argumen yang diperlukan harus sama. Namun, kelas anak mungkin memiliki argumen opsional sebagai tambahan.

Jadi, ketika kelas anak diwarisi dari kelas abstrak, kita memiliki aturan berikut:

Metode kelas anak harus didefinisikan dengan nama yang sama dan mendeklarasikan ulang metode abstrak induk Metode kelas anak harus ditentukan dengan pengubah akses yang sama atau kurang dibatasi Jumlah argumen yang diperlukan harus sama. Namun, kelas anak mungkin memiliki argumen opsional sebagai tambahan

Mari kita lihat contohnya:

<?php  
// Parent class  
abstract class Car {  
  public $name;  
  public function \_\_construct($name) {  
    $this->name = $name;  
  }  
  abstract public function intro() : string;  
}  
  
// Child classes  
class Audi extends Car {  
  public function intro() : string {  
    return "Choose German quality! I'm an $this->name!";  
  }  
}  
  
class Volvo extends Car {  
  public function intro() : string {  
    return "Proud to be Swedish! I'm a $this->name!";  
  }  
}  
  
class Citroen extends Car {  
  public function intro() : string {  
    return "French extravagance! I'm a $this->name!";  
  }  
}  
  
// Create objects from the child classes  
$audi = new audi("Audi");  
echo $audi->intro();  
echo "<br>";  
  
$volvo = new volvo("Volvo");  
echo $volvo->intro();  
echo "<br>";  
  
$citroen = new citroen("Citroen");  
echo $citroen->intro();  
?>

YContoh Dijelaskan

Kelas Audi, Volvo, dan Citroen diwarisi dari kelas Mobil. Ini berarti bahwa kelas Audi,Volvo, dan Citroen dapat menggunakan properti $ name publik serta metode public \_\_construct () dari kelas Mobil karena faktor pewarisan.

Tapi, intro () adalah metode abstrak yang harus didefinisikan di semua kelas anak dan mereka harus mengembalikan sebuah string.

PHP - Contoh Kelas Lebih Abstrak

Mari kita lihat contoh lain di mana metode abstrak memiliki argumen:

Contoh

<?php  
abstract class ParentClass {  
  // Abstract method with an argument  
  abstract protected function prefixName($name);  
}  
  
class ChildClass extends ParentClass {  
  public function prefixName($name) {  
    if ($name == "John Doe") {  
      $prefix = "Mr.";  
    } elseif ($name == "Jane Doe") {  
      $prefix = "Mrs.";  
    } else {  
      $prefix = "";  
    }  
    return "{$prefix} {$name}";  
  }  
}  
  
$class = new ChildClass;  
echo $class->prefixName("John Doe");  
echo "<br>";  
echo $class->prefixName("Jane Doe");  
?>

Try it

# 9. PHP OOP – Interfaces

PHP - Apa itu Antarmuka?

Antarmuka memungkinkan Anda menentukan metode apa yang harus diterapkan kelas.

Antarmuka memudahkan penggunaan berbagai kelas berbeda dengan cara yang sama. Ketika satu atau lebih kelas menggunakan antarmuka yang sama, itu disebut sebagai "polimorfisme".

Antarmuka dideklarasikan dengan kata kunci antarmuka:

### Syntax

<?php  
interface InterfaceName {  
  public function someMethod1();  
  public function someMethod2($name, $color);  
  public function someMethod3() : string;  
}  
?>

PHP - Antarmuka vs. Kelas Abstrak

Antarmuka mirip dengan kelas abstrak. Perbedaan antara antarmuka dan kelas abstrak adalah: Antarmuka tidak boleh memiliki properti, sedangkan kelas abstrak bisa Semua metode antarmuka harus publik, sedangkan metode kelas abstrak bersifat publik atau dilindungi Semua metode dalam sebuah antarmuka bersifat abstrak, sehingga tidak dapat diterapkan dalam kode dan kata kunci abstrak tidak diperlukanK elas dapat mengimplementasikan antarmuka sambil mewarisi dari kelas lain pada waktu yang sama PHP - Menggunakan Antarmuka Untuk mengimplementasikan sebuah antarmuka, sebuah kelas harus menggunakan kata kunci implement. Kelas yang mengimplementasikan antarmuka harus mengimplementasikan semua metode antarmuka.

### Example

<?php  
interface Animal {  
  public function makeSound();  
}  
  
class Cat implements Animal {  
  public function makeSound() {  
    echo "Meow";  
  }  
}  
  
$animal = new Cat();  
$animal->makeSound();  
?>

Dari contoh di atas, misalkan kita ingin membuat perangkat lunak yang mengelola sekelompok hewan. Ada tindakan yang dapat dilakukan semua hewan, tetapi setiap hewan melakukannya dengan caranya sendiri.

Dengan menggunakan antarmuka, kita dapat menulis beberapa kode yang dapat bekerja untuk semua hewan meskipun setiap hewan berperilaku berbeda:

### Example

<?php  
// Interface definition  
interface Animal {  
  public function makeSound();  
}  
  
// Class definitions  
class Cat implements Animal {  
  public function makeSound() {  
    echo " Meow ";  
  }  
}  
  
class Dog implements Animal {  
  public function makeSound() {  
    echo " Bark ";  
  }  
}  
  
class Mouse implements Animal {  
  public function makeSound() {  
    echo " Squeak ";  
  }  
}  
  
// Create a list of animals  
$cat = new Cat();  
$dog = new Dog();  
$mouse = new Mouse();  
$animals = array($cat, $dog, $mouse);  
  
// Tell the animals to make a sound  
foreach($animals as $animal) {  
  $animal->makeSound();  
}  
?>

# 10. PHP OOP – Traits

PHP - Apa itu Sifat?

PHP hanya mendukung pewarisan tunggal: kelas anak hanya dapat mewarisi dari satu orang tua. Jadi, bagaimana jika sebuah kelas perlu mewarisi beberapa perilaku? Ciri OOP memecahkan masalah ini. Sifat digunakan untuk mendeklarasikan metode yang dapat digunakan di banyak kelas. Sifat dapat memiliki metode dan metode abstrak yang dapat digunakan di banyak kelas, dan metode dapat memiliki pengubah akses apa pun (publik, pribadi, atau dilindungi). Sifat dideklarasikan dengan kata kunci sifat:

### Syntax

<?php  
trait TraitName {  
  // some code...  
}  
?>

Contoh Dijelaskan

Di sini, kami mendeklarasikan satu ciri: message1. Kemudian, kami membuat kelas: Selamat Datang. Kelas menggunakan sifat tersebut, dan semua metode dalam sifat tersebut akan tersedia di kelas. Jika kelas lain perlu menggunakan fungsi msg1 (), cukup gunakan sifat message1 di kelas tersebut. Ini mengurangi duplikasi kode, karena tidak perlu mendeklarasikan ulang metode yang sama berulang kali.PHP - Menggunakan Beberapa Sifat Mari kita lihat contoh lain:

### Example

<?php  
trait message1 {  
  public function msg1() {  
    echo "OOP is fun! ";  
  }  
}  
  
trait message2 {  
  public function msg2() {  
    echo "OOP reduces code duplication!";  
  }  
}  
  
class Welcome {  
  use message1;  
}  
  
class Welcome2 {  
  use message1, message2;  
}  
  
$obj = new Welcome();  
$obj->msg1();  
echo "<br>";  
  
$obj2 = new Welcome2();  
$obj2->msg1();  
$obj2->msg2();  
?>

## 11. PHP - Static Methods

Metode statis dapat dipanggil secara langsung - tanpa membuat instance kelas terlebih dahulu. Metode statis dideklarasikan dengan kata kunci statis:

### Syntax

<?php  
class ClassName {  
  public static function staticMethod() {  
    echo "Hello World!";  
  }  
}  
?>

To access a static method use the class name, double colon (::), and the method name:

### Syntax

ClassName::staticMethod();

Let's look at an example:

### Example

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
}  
  
// Call static method  
greeting::welcome();  
?>

### Example Explained

Here, we declare a static method: welcome(). Then, we call the static method by using the class name, double colon (::), and the method name (without creating an instance of the class first).

## PHP - More on Static Methods

A class can have both static and non-static methods. A static method can be accessed from a method in the same class using the self keyword and double colon (::):

### Example

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
  
  public function \_\_construct() {  
    self::welcome();  
  }  
}  
  
new greeting();  
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_static_method2)

Static methods can also be called from methods in other classes. To do this, the static method should be public:

### Example

<?php  
class greeting {  
  public static function welcome() {  
    echo "Hello World!";  
  }  
}  
  
class SomeOtherClass {  
  public function message() {  
    greeting::welcome();  
  }  
}  
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_static_method3)

To call a static method from a child class, use the parent keyword inside the child class. Here, the static method can be public or protected.

### Example

<?php  
class domain {  
  protected static function getWebsiteName() {  
    return "W3Schools.com";  
  }  
}  
  
class domainW3 extends domain {  
  public $websiteName;  
  public function \_\_construct() {  
    $this->websiteName = parent::getWebsiteName();  
  }  
}  
  
$domainW3 = new domainW3;  
echo $domainW3 -> websiteName;  
?>

# 12. PHP OOP - Static Properties

PHP - Properti Statis

Properti statis dapat dipanggil secara langsung - tanpa membuat instance kelas.Properti statis dideklarasikan dengan kata kunci statis:

### Syntax

<?php  
class ClassName {  
  public static $staticProp = "W3Schools";  
}  
?>

Untuk mengakses properti statis gunakan nama kelas, titik dua ganda (: :), dan nama properti:

### Syntax

ClassName::staticProp;

Let's look at an example:

### Example

<?php  
class pi {  
  public static $value = 3.14159;  
}  
  
// Get static property  
echo pi::$value;  
?>

# 13. PHP Namespaces

Namespaces adalah kualifikasi yang menyelesaikan dua masalah berbeda:Mereka memungkinkan pengorganisasian yang lebih baik dengan mengelompokkan kelas yang bekerja sama untuk melakukan tugas

Mereka memungkinkan nama yang sama digunakan untuk lebih dari satu kelasM isalnya, Anda mungkin memiliki sekumpulan kelas yang mendeskripsikan tabel HTML, seperti Tabel, Baris, dan Sel sementara juga memiliki sekumpulan kelas lain untuk mendeskripsikan furnitur, seperti Meja, Kursi, dan Tempat Tidur. Namespaces dapat digunakan untuk mengatur kelas menjadi dua kelompok berbeda sementara juga mencegah dua kelas Tabel dan Tabel agar tidak tercampur.

Mendeklarasikan Namespace

Namespaces dideklarasikan di awal file menggunakan kata kunci namespace:

### Syntax

Declare a namespace called Html:

namespace Html;

Catatan: Deklarasi namespace harus menjadi hal pertama dalam file PHP. Kode berikut tidak valid:

<?php  
echo "Hello World!";  
namespace Html;  
...  
?>

Constants, classes and functions declared in this file will belong to the **Html** namespace:

### Example

Create a Table class in the Html namespace:

<?php  
namespace Html;  
class Table {  
  public $title = "";  
  public $numRows = 0;  
  public function message() {  
    echo "<p>Table '{$this->title}' has {$this->numRows} rows.</p>";  
  }  
}  
$table = new Table();  
$table->title = "My table";  
$table->numRows = 5;  
?>  
  
<!DOCTYPE html>  
<html>  
<body>  
  
<?php  
$table->message();  
?>  
  
</body>  
</html>

Menggunakan Namespaces

Kode apa pun yang mengikuti deklarasi namespace beroperasi di dalam namespace, sehingga kelas yang termasuk dalam namespace dapat dibuat instance-nya tanpa qualifier apa pun. Untuk mengakses kelas dari luar namespace, kelas harus memiliki namespace yang terpasang padanya.

### Example

Use classes from the Html namespace:

$table = new Html\Table()  
$row = new Html\Row();

# 14. PHP Iterables

PHP - Apa itu Iterable?

Sebuah iterable adalah nilai apa pun yang dapat diulangi dengan perulangan foreach ().Pseudo-type iterable diperkenalkan di PHP 7.1, dan dapat digunakan sebagai tipe data untuk argumen fungsi dan nilai kembalian fungsi.

PHP - Menggunakan Iterables

Kata kunci iterable dapat digunakan sebagai tipe data dari argumen fungsi atau sebagai tipe kembalian dari sebuah fungsi:

### Example

Use an iterable function argument:

<?php  
function printIterable(iterable $myIterable) {  
  foreach($myIterable as $item) {  
    echo $item;  
  }  
}  
  
$arr = ["a", "b", "c"];  
printIterable($arr);  
?>

HP - Membuat Iterables

Array

Semua array bersifat iterable, jadi array apa pun dapat digunakan sebagai argumen dari suatu fungsi yang memerlukan iterable.

Iterator

Objek apa pun yang mengimplementasikan antarmuka Iterator dapat digunakan sebagai argumen dari fungsi yang membutuhkan iterable.

Sebuah iterator berisi daftar item dan menyediakan metode untuk mengulanginya. Itu membuat penunjuk ke salah satu elemen dalam daftar. Setiap item dalam daftar harus memiliki kunci yang dapat digunakan untuk menemukan item tersebut.

Sebuah iterator harus memiliki metode ini:

1. current () - Mengembalikan elemen yang ditunjuk penunjuk saat ini. Ini bisa berupa tipe data apa pun
2. key () Mengembalikan kunci yang terkait dengan elemen saat ini dalam daftar. Ini hanya dapat berupa integer, float, boolean, atau string
3. next () Memindahkan penunjuk ke elemen berikutnya dalam daftar
4. rewind () Memindahkan penunjuk ke elemen pertama dalam daftar
5. valid () Jika penunjuk internal tidak menunjuk ke elemen apa pun (misalnya, jika next () dipanggil di akhir daftar), ini harus mengembalikan false. Ini mengembalikan true dalam kasus lain

### Example

Implement the Iterator interface and use it as an iterable:

<?php  
// Create an Iterator  
class MyIterator implements Iterator {  
  private $items = [];  
  private $pointer = 0;  
  
  public function \_\_construct($items) {  
    // array\_values() makes sure that the keys are numbers  
    $this->items = array\_values($items);  
  }  
  
  public function current() {  
    return $this->items[$this->pointer];  
  }  
  
  public function key() {  
    return $this->pointer;  
  }  
  
  public function next() {  
    $this->pointer++;  
  }  
  
  public function rewind() {  
    $this->pointer = 0;  
  }  
  
  public function valid() {  
    // count() indicates how many items are in the list  
    return $this->pointer < count($this->items);  
  }  
}  
  
// A function that uses iterables  
function printIterable(iterable $myIterable) {  
  foreach($myIterable as $item) {  
    echo $item;  
  }  
}  
  
// Use the iterator as an iterable  
$iterator = new MyIterator(["a", "b", "c"]);  
printIterable($iterator);  
?>

[Try it Yourself »](https://www.w3schools.com/php/phptryit.asp?filename=tryphp_inheritance6)