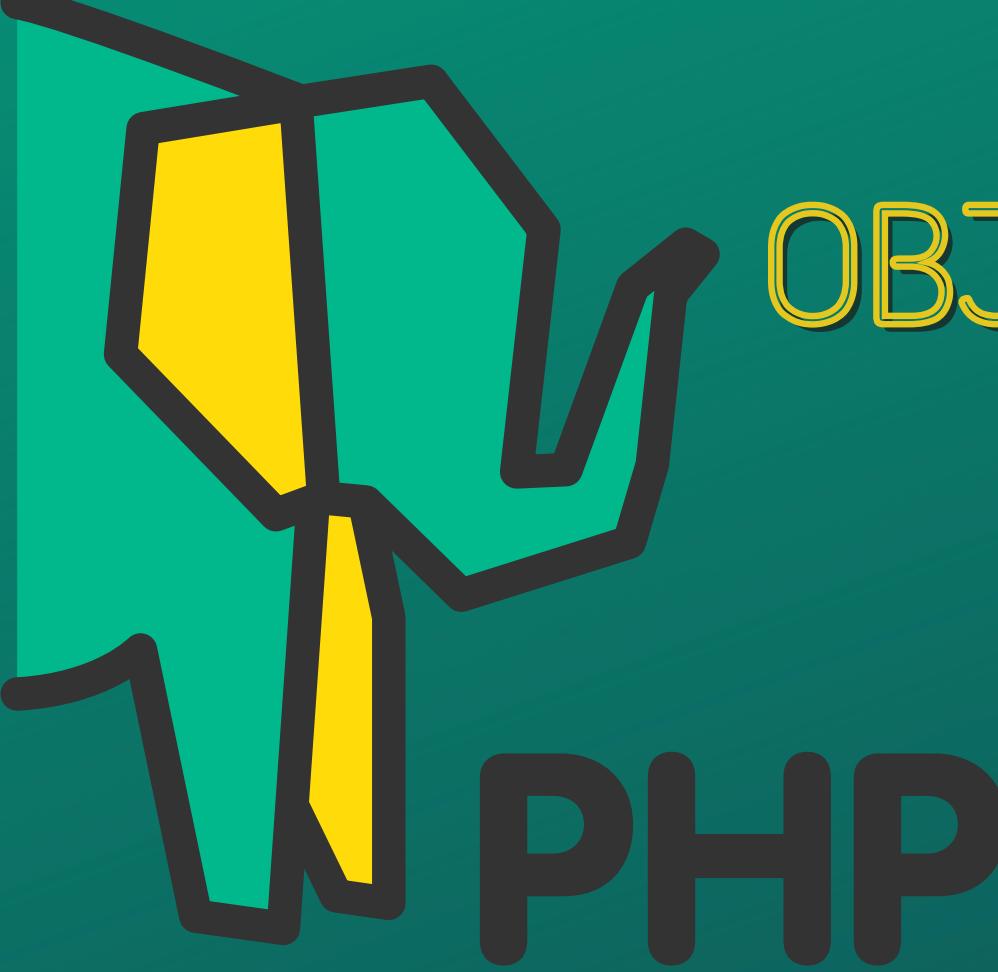
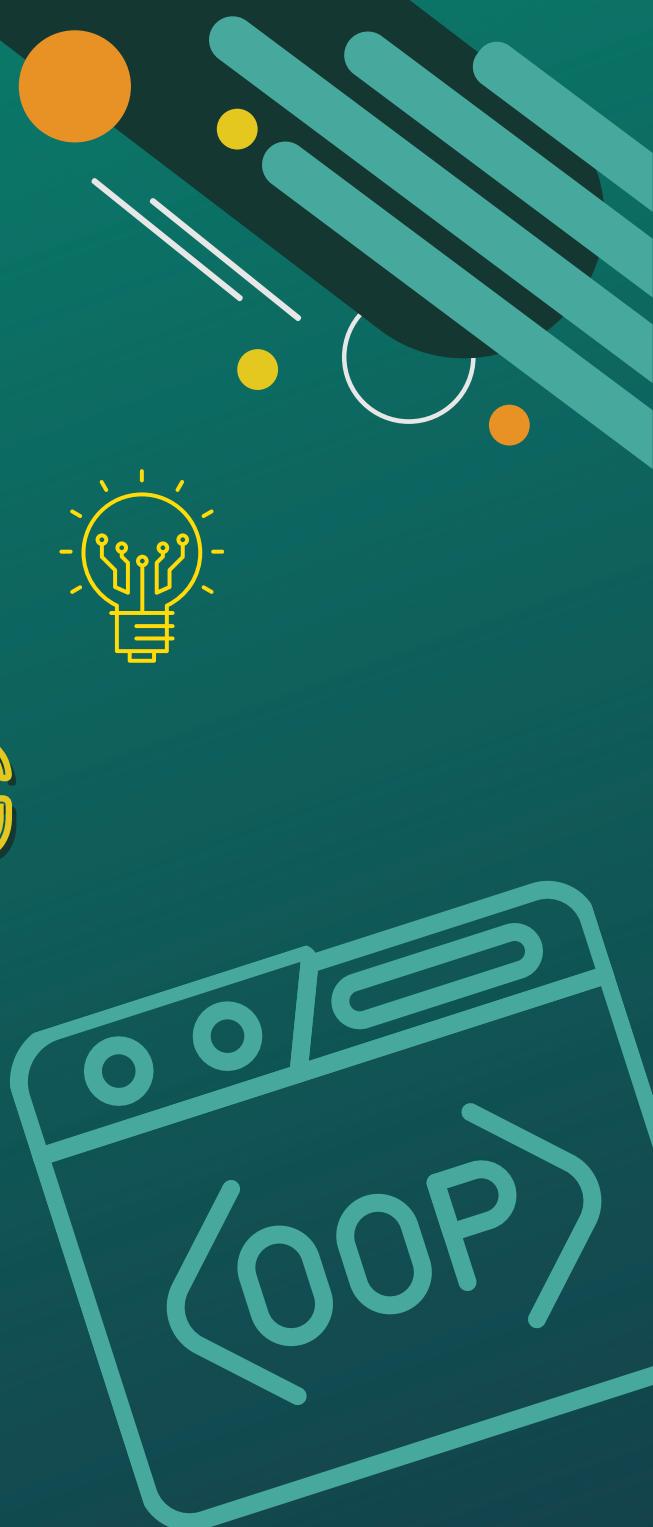


PEMOGRAMAN BERBASIS WEB



# OBJECT ORIENTED PROGRAMMING DENGAN PHP



GUDANG GUNAWAN



## WHAT IS OBJECT ORIENTED PROGRAMMING ?

Merupakan gaya pemrograman sebagai cara kita membuat code program dengan metode yang lebih berorientasi pada objek.

## WHAT IS PROCEDURAL PROGRAMMING ?

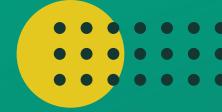
Paradigma pemrograman, yang penamaannya diambil dari pemrograman imperatif berdasarkan konsep pemanggilan prosedur. Prosedur (sejenis rutin atau subrutin) hanya berisi serangkaian langkah komputasi yang akan dilakukan.



# PEMOGRAMAN BERBASIS WEB

## PERBEDAAN SINGKAT

Prosedural Programming	Object Oriented Programming
Intruksi di lakukan langkah demi langkah	Menyusun semua kode program dan data sebagai objek
Memecah program menjadi bagian bagian kecil	Objek adalah unit dasar dari program
Disebut prosedur, subroutine atau function	Objek menyimpan data dan perilaku
Linear / Top to Bottom	Objek bisa saling berinteraksi
Fortan, Algol, Cobol, Pascal, C, PHP, Javascript	Java, Ruby, Python, C++, PHP



# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB

# PERBEDAAN CODE

## PROCEDURAL PROGRAMMING

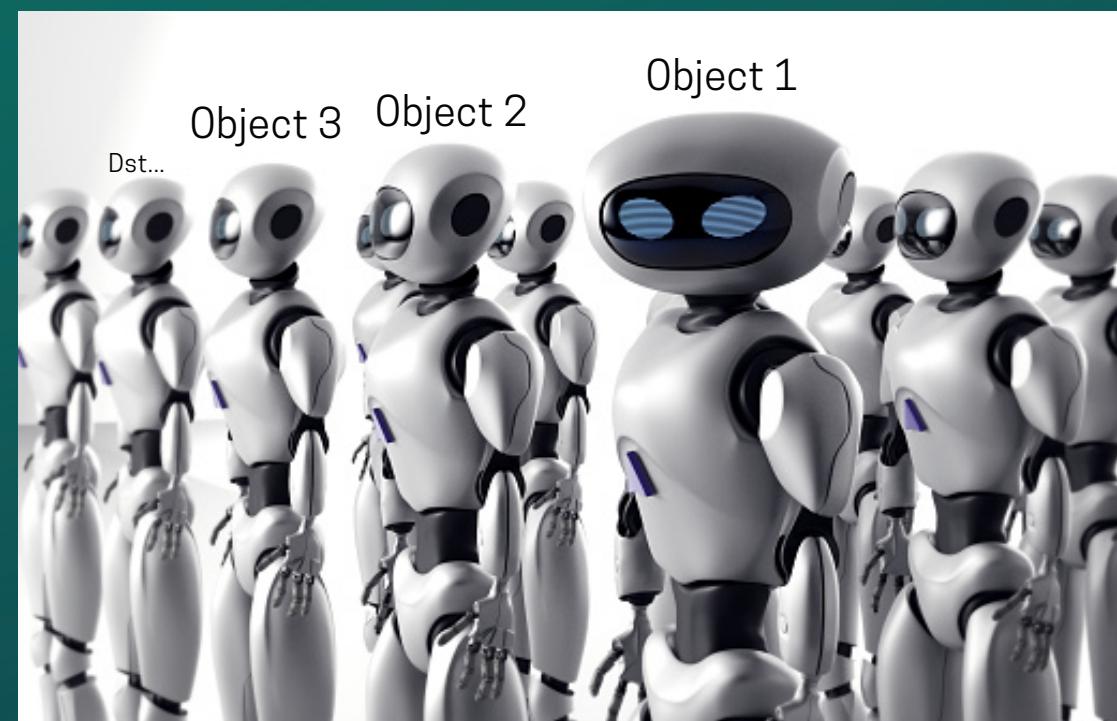
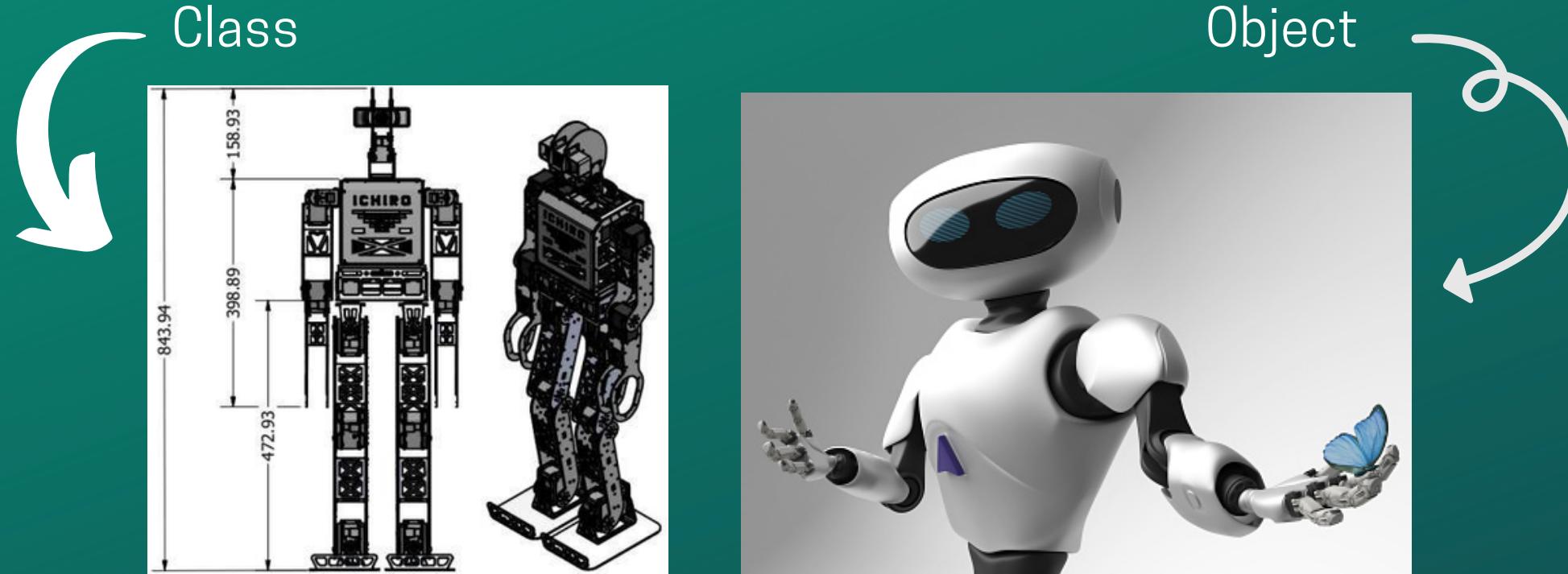
```
1 <?php
2 // Pendekatan Prosedural
3
4 // Fungsi untuk menghitung luas segitiga
5 function hitungLuasSegitiga($alas, $tinggi) {
6     return 0.5 * $alas * $tinggi;
7 }
8
9 // Menggunakan fungsi untuk menghitung luas segitiga
10 $alas = 10;
11 $tinggi = 5;
12 $luas = hitungLuasSegitiga($alas, $tinggi);
13
14 echo "Luas segitiga dengan alas $alas dan tinggi $tinggi adalah $luas \n";
15 ?>
16
```

## OBJECT ORIENTED PROGRAMMING

```
1 <?php
2 // Pendekatan OOP
3
4 class Segitiga {
5     private $alas;
6     private $tinggi;
7
8     // Konstruktor
9     public function __construct($alas, $tinggi) {
10         $this->alas = $alas;
11         $this->tinggi = $tinggi;
12     }
13
14     // Metode untuk menghitung luas segitiga
15     public function hitungLuas() {
16         return 0.5 * $this->alas * $this->tinggi;
17     }
18
19     // Getter dan Setter untuk alas
20     public function setAlas($alas) {
21         $this->alas = $alas;
22     }
23
24     public function getAlas() {
25         return $this->alas;
26     }
27
28     // Getter dan Setter untuk tinggi
29     public function setTinggi($tinggi) {
30         $this->tinggi = $tinggi;
31     }
32
33     public function getTinggi() {
34         return $this->tinggi;
35     }
36 }
37
38 // Membuat objek dari kelas Segitiga
39 $segitiga = new Segitiga(10, 5);
40
41 // Menggunakan metode untuk menghitung luas segitiga
42 $luas = $segitiga->hitungLuas();
43
44 echo "Luas segitiga dengan alas {$segitiga->getAlas()} dan tinggi {$segitiga->getTinggi()} adalah $luas \n";
45 ?>
46
```



# CLASS & OBJECT





## PERBEDAAN SINGKAT

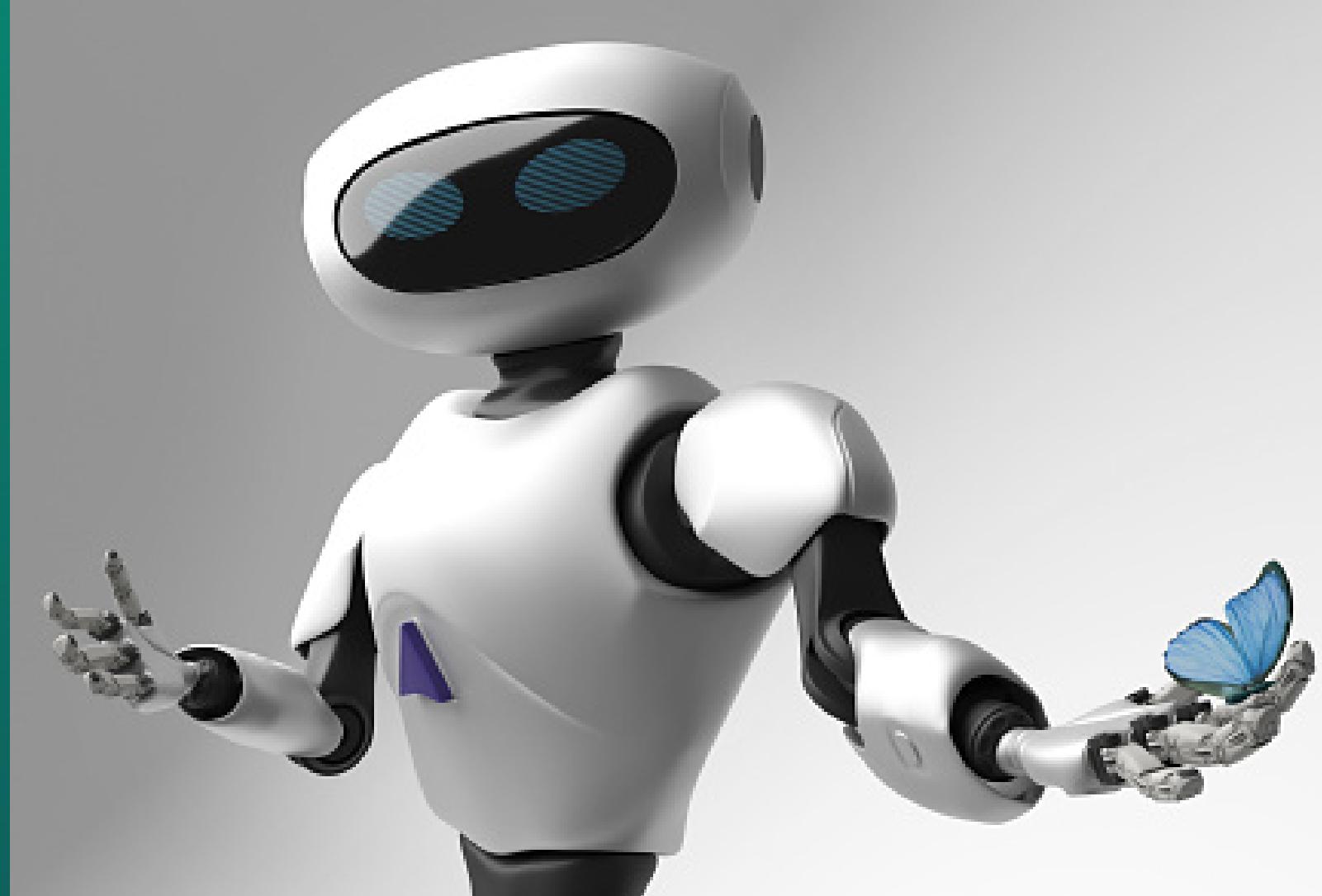
Class	Object
Blueprint / Template untuk membuat instance dari object	Instance yang didefinisikan oleh class
Class mendefinisikan object	Bisa membuat banyak object yang dapat dibuat menggunakan satu class
Menyimpan data dan perilaku yang disebut dengan property dan method	Object dibuat dengan menggunakan keyword <b>new</b>





PEMOGRAMAN BERBASIS WEB

# PROPERTY & METHOD



## PROPERTY

- Suara
- Berat
- Warna

## METHOD

- Bersuara
- Berat Robot
- Bergerak Maju
- Menghancurkan Bumi

GUDANG GUNAWAN





## PERBEDAAN SINGKAT

Property	Method
Merepresentasikan data / keadaan dari sebuah object	Merepresentasikan perilaku dari sebuah object
Variabel yang ada di dalam object (member variable)	Function yang ada di dalam object
Sama seperti variable di dalam PHP, ditambah dengan visibility di depan	Sama seperti function di dalam PHP, ditambah dengan visibility di depannya





# PEMOGRAMAN BERBASIS WEB

## CODE PROGRAM

```
1 <?php
2
3 // Class
4 Class Robot {
5
6     //property
7     public $suara = 'ngik-ngik';
8     public $berat = 30;
9
10    //metode
11    public function bersuara (){
12        echo 'suara robotnya ...' . $this->suara;
13    }
14    public function berat_robot(){
15        echo 'Berat robotnya adalah...';
16        return $this->berat;
17    }
18
19    }
20 // object
21 $robot1 = new robot;
22 $robot1 ->bersuara();
23 echo $robot1->berat_robot();
24 ?>
```





# SETTER & GETTER

Getter dan setter merupakan method untuk melakukan mengisi (set) dan mengambil nilai (get).





# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB

# CODE PROGRAM

```
1 class Robot {  
2  
3     // Property  
4     public $suara;  
5     public $berat;  
6  
7     // Metode set & get  
8     public function set_suara($suara){  
9         $this->suara = $suara;  
10    }  
11  
12    public function get_suara(){  
13        return $this->suara;  
14    }  
15  
16    public function set_berat($berat){  
17        $this->berat = $berat;  
18    }  
19  
20    public function get_berat(){  
21        return $this->berat;  
22    }  
23  
24 }  
25  
// Object  
26 $robot1 = new Robot;  
27 $robot2 = new Robot;  
28 $robot1->set_suara('ngik-ngik');  
29 echo 'bunyinya ... ' . $robot1->get_suara() . '<br>';  
30 $robot1->set_suara('ngok-ngok');  
31 echo 'bunyinya ... ' . $robot1->get_suara() . '<br>';  
32  
33 $robot2->set_berat(30);  
34 echo 'berat robotnya ... ' . $robot2->get_berat() . '<br>';  
35
```





# CONSTRUCTOR

Merupakan sebuah method yang akan dijalankan ketika sebuah object di instansiasi. Untuk membuat constructor di PHP menggunakan keyword **`_construct`**.



# PEMOGRAMAN BERBASIS WEB

## CODE PROGRAM

```
● ● ●  
1 /*Constructor  
2 require_once 'robot.php';  
3 // Object  
4 $robot1 = new Robot;  
5 $robot2 = new Robot;  
6 $robot3 = new Robot;  
7 $robot1->set_suara('ngik-ngik');  
8 echo 'bunyinya ... ' . $robot1->get_suara() . '<br>';  
9 $robot2->set_suara('ngok-ngok');  
10 echo 'bunyinya ... ' . $robot2->get_suara() . '<br>';  
11  
12 $robot3->set_berat(30);  
13 echo 'berat robotnya ... ' . $robot3->get_berat() . '<br>';  
14 */  
15 //Constructor2  
16 require_once 'robot.php';  
17 // Object  
18 $robot1 = new Robot ('ngik-ngik', 75);  
19 $robot2 = new Robot ('ngok-ngok', 80);  
20 $robot3 = new Robot ('dewani puh', 55);  
21  
22 echo 'bunyinya ... ' . $robot1->get_suara() . '<br>';  
23 echo 'beratnya ... ' . $robot1->get_berat() . '<br>';  
24 echo 'bunyinya ... ' . $robot2->get_suara() . '<br>';  
25 echo 'beratnya ... ' . $robot2->get_berat() . '<br>';  
26 echo 'bunyinya ... ' . $robot3->get_suara() . '<br>';  
27 echo 'beratnya ... ' . $robot3->get_berat() . '<br>';  
28  
29 ?>
```

```
● ● ●  
1 <?php  
2 // Class  
3 class Robot {  
4  
5     // Property  
6     public $suara;  
7     public $berat;  
8  
9     /* Constructor  
10    public function __construct(){  
11        echo 'halo robot..';  
12    }*/  
13  
14    //Constructor2  
15    public function __construct($suara, $berat){  
16        $this->suara = $suara;  
17        $this->berat = $berat;  
18    }  
19    /*  
20    // Constructor Default  
21    public function __construct($suara = 'ohh may gat',$berat = 88){  
22        $this->suara = $suara;  
23        $this->berat = $berat;  
24    }*/  
25  
26  
27    // Metode set & get  
28    public function set_suara($suara){  
29        $this->suara = $suara;  
30    }  
31  
32    public function get_suara(){  
33        return $this->suara;  
34    }  
35  
36    public function set_berat($berat){  
37        $this->berat = $berat;  
38    }  
39  
40    public function get_berat(){  
41        return $this->berat;  
42    }  
43  
44 }  
45 ?>  
46
```



# PEMOGRAMAN BERBASIS WEB



## Inheritance

Menciptakan hierarki antar kelas (Parent & Child)

Child Class, mewarisi semua properti dan method dari parent-nya (yang visible)

Child Class, memperluas (extends) fungsionalitas dari parent-nya





# PEMOGRAMAN BERBASIS WEB

## PARENT PROPERTY

- Suara
- Berat
- Warna

## METHOD

- Bersuara
- Berat Robot
- Bergerak Maju

## INHERITANCE



## CHILD PROPERTY

- khusus

## METHOD

- khusus

GUDANG GUNAWAN





## CODE PROGRAM INHERITANCE & OVERRIDING



```
1 //Inheritance
2 $robothewan = new robot_hewan ('pika-pika', 20);
3 echo $robothewan->get_suara(). '<br>';
4 echo 'beratnya ... ' . $robothewan->get_berat() . '<br>';
5 echo $robothewan->get_kekuatan();
6 ?>
```



```
1 <?php
2
3 require_once 'robot.php';
4
5 //Inheritance
6 class robot_hewan extends Robot {
7
8     public function get_kekuatan(){
9         echo 'Saya memiliki kekuatan petir...';
10    }
11
12 // Overriding
13 public function get_suara(){
14     return 'suaranya adalah ....' . parent::get_suara();
15 }
16 }
17
18 ?>
```





# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB



### Visibility (Access Modifier)

Konsep yang digunakan untuk mengatur akses dari property dan method pada sebuah objek

Ada 3 keyword visibility : **public, protected, dan private**





I.N.G.O.U.D.E.

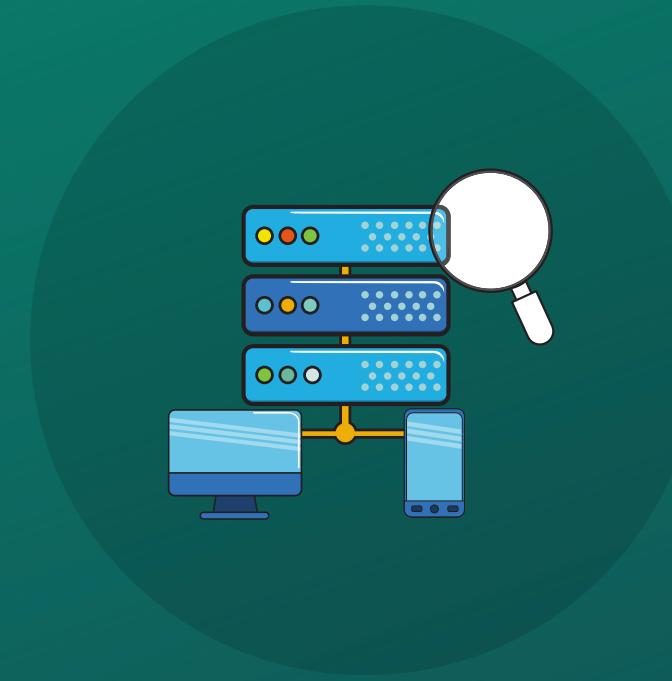
# VISIBILITY (ACCESS MODIFIER)

Public



Dapat digunakan di mana saja, bahkan di luar kelas

Protected



Hanya dapat digunakan di dalam sebuah kelas beserta turunannya

Private



Hanya dapat digunakan di dalam sebuah kelas tertentu saja





# PEMOGRAMAN BERBASIS WEB

## CODE PROGRAM



```
1 $robotPertanian = new RobotPertanian();
2 echo 'bunyinya ... ' . $robotPertanian->get_suara() . '<br>';
3 echo 'beratnya ... ' . $robotPertanian->get_berat() . '<br>';
4 echo 'Jenis tanamannya... ' . $robotPertanian->get_jenisTanaman() . '<br>';
5
```



```
1 class RobotPertanian extends Robot {
2
3     private $jenisTanaman;
4
5     public function __construct($suara = 'Nanem Maju', $berat = 100, $jenisTanaman = 'padi'){
6         parent::__construct($suara, $berat);
7         $this->jenisTanaman = $jenisTanaman;
8     }
9
10    private function set_jenisTanaman($jenisTanaman){
11        self::$jenisTanaman = $jenisTanaman;
12    }
13
14    public function get_jenisTanaman(){
15        return $this->jenisTanaman;
16    }
}
```





# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB



### Static Keyword

Kata static keyword digunakan untuk mendeklarasikan properti dan metode suatu kelas sebagai statis. Properti dan metode statis dapat digunakan tanpa membuat turunan kelas.

Kata static keyword juga digunakan untuk mendeklarasikan variabel dalam suatu fungsi yang mempertahankan nilainya setelah fungsi tersebut berakhir.

Membuat kode menjadi prosedural, biasanya digunakan untuk membuat fungsi bantuan/helper, atau digunakan di class-class utility pada framework





# PEMOGRAMAN BERBASIS WEB

## CODE PROGRAM

```
● ● ●  
1 <?php  
2  
3 class orang {  
4  
5     static $angka = 1;  
6  
7     public function ajakan(){  
8         return 'jumlah ngajak.... ' . self :: $angka++ . ' kali. <br>';  
9     }  
10  
11    public static function bersuara (){  
12        echo 'hallo-hallo dek kapan bisa jalan....';  
13    }  
14 }  
15  
16 orang::bersuara();  
17 echo '<hr>';  
18  
19 $ajakan1 = new orang;  
20 echo $ajakan1->ajakan();  
21 echo $ajakan1->ajakan();  
22 echo '<hr>';  
23  
24 $ajakan2 = new orang;  
25 echo $ajakan2->ajakan();  
26 echo $ajakan2->ajakan();  
27 echo '<hr>';  
28
```





## PROGRAMMING

# PEMOGRAMAN BERBASIS WEB

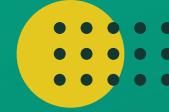


### Constant

Konstanta adalah pengidentifikasi (nama) untuk nilai sederhana. Nilainya tidak dapat diubah selama skrip

Nama konstanta yang valid dimulai dengan huruf atau garis bawah (tidak ada tanda \$ sebelum nama konstanta).





# PROGRAMMING

## BERBASIS WEB

# CODE PROGRAM

```
● ● ●  
1 <?php  
2  
3 // Definisi konstanta global dengan define()  
4 define('UNIVERSITAS', 'Universitas Teknologi');  
5  
6 class Mahasiswa {  
7     // Definisi konstanta dalam kelas dengan const  
8     const PROGRAM_STUDI = 'Teknik Informatika';  
9  
10    public function getUniversitas() {  
11        return UNIVERSITAS;  
12    }  
13  
14    public function getProgramStudi() {  
15        return self::PROGRAM_STUDI;  
16    }  
17  
18 }  
19  
20 // Mengakses konstanta global  
21 echo 'Universitas: ' . UNIVERSITAS . '<br>';  
22  
23 // Membuat objek dari kelas Mahasiswa  
24 $objMahasiswa = new Mahasiswa();  
25  
26 // Mengakses metode untuk mendapatkan konstanta global  
27 echo 'Universitas dari metode: ' . $objMahasiswa->getUniversitas() . '<br>';  
28  
29 // Mengakses konstanta dalam kelas Mahasiswa  
30 echo 'Program Studi: ' . Mahasiswa::PROGRAM_STUDI . '<br>';  
31  
32 // Mengakses metode untuk mendapatkan konstanta dalam kelas Mahasiswa  
33 echo 'Program Studi dari metode: ' . $objMahasiswa->getProgramStudi();  
34  
35 ?>  
36
```



# PEMOGRAMAN BERBASIS WEB

## Magic Constants

	Magic Constants
<code>__LINE__</code>	Berisi sebuah nilai yang menyatakan nomer baris saat itu.
<code>__FILE__</code>	Berisi alamat lengkap (path) dari file PHP.
<code>__DIR__</code>	Hampir sama dengan konstanta <code>__FILE__</code> , konstanta <code>__DIR__</code> akan berisi alamat direktori dari file PHP.
<code>__FUNCTION__</code>	Berisi nama fungsi.
<code>__CLASS__</code>	Berisi nama dari class.
<code>__TRAIT__</code>	Berisi nama trait dan namespace.
<code>__METHOD__</code>	Sama seperti konstanta <code>__FUNCTION__</code> la akan berisi nama method dan class tempat ia digunakan.



# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB

### Abstract Class

Sebuah class yang tidak dapat di-intansiasi secara langsung.

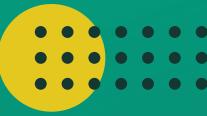
Sebagai class induk memiliki metode bernama, namun memerlukan kelas turunannya untuk mengisi tugas.

Memiliki minimal 1 method abstrak, digunakan dalam inheritance untuk memasakan implementasi method abstrak yang sama untuk semua kelas turunannya.

Semua kelas turunannya, harus mengimplementasikan method abstrak yang ada di kelas abstraknya.

Kelas abstrak boleh memiliki property / method reguler ataupun static method.





PROGRAMMING

PEMOGRAMAN BERBASIS WEB

# CODE PROGRAM

```
1 <?php
2
3 abstract class Robot {
4     protected $name;
5     protected $model;
6
7     // Constructor
8     public function __construct($name, $model) {
9         $this->name = $name;
10        $this->model = $model;
11    }
12
13    // Abstract method for action
14    abstract public function action();
15
16    // Getter method for name
17    public function getName() {
18        return $this->name;
19    }
20
21    // Getter method for model
22    public function getModel() {
23        return $this->model;
24    }
25
26    // Display information
27    public function displayInfo() {
28        echo "Name: {$this->name}\n";
29        echo "Model: {$this->model}\n";
30    }
31 }
32
33 // Contoh penggunaan abstract class Robot
34 class CleaningRobot extends Robot {
35     public function action() {
36         echo "{$this->name} is cleaning.\n";
37     }
38 }
39
40 class CookingRobot extends Robot {
41     public function action() {
42         echo "{$this->name} is cooking.\n";
43     }
44 }
45
46 // Membuat objek dari kelas CleaningRobot
47 $cleaningRobot = new CleaningRobot("Cleaner", "C1");
48 $cleaningRobot->displayInfo();
49 $cleaningRobot->action();
50 echo '<hr>';
51 // Membuat objek dari kelas CookingRobot
52 $cookingRobot = new CookingRobot("Chef", "C2");
53 $cookingRobot->displayInfo();
54 $cookingRobot->action();
55
56 ?>
57
```

GUDANG GUNAWAN





# PEMOGRAMAN BERBASIS WEB

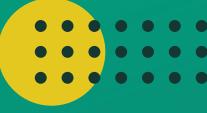
## Interface

Merupakan kelas abstrak yang sama sekali tidak memiliki implementasi

Kelas Murni sebagai template untuk kelas turunnanya

Tidak boleh memiliki property, hanya deklarasi method saja





# PROGRAMMING

## BERBASIS WEB

# CODE PROGRAM

```
1  interface Robot {
2      // Abstract method for action
3      public function action();
4
5      // Getter method for name
6      public function getName();
7
8      // Getter method for model
9      public function getModel();
10
11     // Display information
12     public function displayInfo();
13 }
14
15 class CleaningRobot implements Robot {
16     protected $name;
17     protected $model;
18
19     public function __construct($name, $model) {
20         $this->name = $name;
21         $this->model = $model;
22     }
23
24     public function action() {
25         echo "{$this->name} is cleaning.\n";
26     }
27
28     public function getName() {
29         return $this->name;
30     }
31
32     public function getModel() {
33         return $this->model;
34     }
35
36     public function displayInfo() {
37         echo "Name: {$this->name}\n";
38         echo "Model: {$this->model}\n";
39     }
40 }
41
42 class CookingRobot implements Robot {
43     protected $name;
44     protected $model;
45
46     public function __construct($name, $model) {
47         $this->name = $name;
48         $this->model = $model;
49     }
50
51     public function action() {
52         echo "{$this->name} is cooking.\n";
53     }
54
55     public function getName() {
56         return $this->name;
57     }
58
59     public function getModel() {
60         return $this->model;
61     }
62
63     public function displayInfo() {
64         echo "Name: {$this->name}\n";
65         echo "Model: {$this->model}\n";
66     }
67 }
68
69 // Membuat objek dari kelas CleaningRobot
70 $cleaningRobot = new CleaningRobot("Cleaner", "C1");
71 $cleaningRobot->displayInfo();
72 $cleaningRobot->action();
73 echo 'hr';
74
75 // Membuat objek dari kelas CookingRobot
76 $cookingRobot = new CookingRobot("Chef", "C2");
77 $cookingRobot->displayInfo();
78 $cookingRobot->action();
79 echo 'hr';
```

GUDANG GUNAWAN



# PROGRAMMING

## PEMOGRAMAN BERBASIS WEB



### Autoloading

Memanggil class (file) tanpa harus menggunakan require ataupun includ





PEMOGRAMAN BERBASIS WEB

# CODE PROGRAM



```
1 <?php  
2  
3 spl_autoload_register(function ($class) {  
4     require_once __DIR__ . '\\produk.robot\\' . $class . '.php';  
5 });  
6  
7 ?>
```

GUDANG GUNAWAN



## PEMOGRAMAN BERBASIS WEB



THANK'S FOR  
YOUR ATTENTION

JANGAN LUPA TIDUR;

GUDANG GUNAWAN