

Nama : Noor Rizki Ramadhan  
NIM : 2243912

## 1. Constructor

```
1 class Person
2 {
3     /* 'const AUTHOR =' Programmer Zaman Now '; mendefinisikan properti konstan bernama 'penulis' dengan
4     Nilai "programmer zaman sekarang" di dalam kelas 'person'. Konstanta ini dapat diakses dari
5     Di mana saja di dalam kelas menggunakan 'self :: Author' dan nilainya tidak dapat diubah selama runtime.*/
6     const AUTHOR = "Programmer Zaman Now";
7
8     /* Baris dibawah ini mendefinisikan properties pada kelas 'Person'.*/
9     var string $name;
10    var ?string $address = null;
11    var string $country = "Indonesia";
12
13    /**
14     * Ini adalah fungsi konstruktor dalam PHP yang mengambil nama string dan alamat string opsional
15     * sebagai parameter dan menugaskannya ke properti yang sesuai dari objek.
16     *
17     * @param string nama parameter string yang diperlukan mewakili nama objek atau entitas.
18     * @param address parameter '' adalah tipe string yang dapat dikosongkan, artinya itu bisa
19     * string atau nol. Itu adalah parameter opsional yang dapat diteruskan ke konstruktor saat membuat
20     * Objek kelas. Jika nilai diteruskan, itu akan ditugaskan ke properti ''
21     */
22    function __construct(string $name, ?string $address)
23    {
24        $this->name = $name;
25        $this->address = $address;
26    }
27
28    /**
29     * Ini adalah fungsi PHP yang menyapa seseorang dengan nama atau memperkenalkan dirinya jika tidak ada nama yang disediakan.
30     *
31     * @param name adalah parameter fungsi sayHello (). Itu adalah tipe string yang dapat dikosongkan, yang
32     * berarti itu bisa berupa string atau nol. Jika nol, fungsi akan menghasilkan default
33     * Pesan dengan properti nama objek. Jika bukan nol, fungsi akan menghasilkan sebuah pesan
34     */
35    function sayHello(?string $name)
36    {
37        if (is.null($name)) {
38            echo "Hi, my name is $this->name" . PHP_EOL;
39        } else {
40            echo "Hi $name, my name is $this->name" . PHP_EOL;
41        }
42    }
43 }
```

```
1 /* Mengambil fungsi yang ada pada file 'Person.php' yang terletak di direktori 'data', yang berisi kelas 'person'
2 definisi. Ini memungkinkan skrip PHP saat ini untuk mengakses dan menggunakan kelas 'person'. Itu
3 Pernyataan 'require_once' memastikan bahwa file hanya disertakan sekali, bahkan jika pernyataannya
4 disebut beberapa kali.*/
5 require_once "data/Person.php";
6
7
8 /* Kode ini membuat instance baru dari kelas 'person' dengan nama "Eko" dan alamat "Subang",
9 dan menugaskannya ke variabel '$person'. Kemudian, menetapkan properti 'country' dari '$person'
10 dengan "Indonesia".*/
11 $person = new Person("Eko", "Subang");
12 $person->country = "Indonesia";
13
14 /* Kode memanggil fungsi 'sayHello()' dari objek '$person' dua kali. Panggilan pertama melewati
15 String "Budi" sebagai argumen, sementara panggilan kedua melewati 'null'. Fungsi 'sayHello()' akan
16 mengeluarkan output pesan ucapan yang menyertakan nama yang disahkan sebagai argumen, atau pesan default jika tidak
17 ada Nama disediakan.*/
18 $person->sayHello("Budi");
19 $person->sayHello(null);
```

```
Hi Budi, my name is Eko
Hi, my name is Eko
```

## 2. Destructor

```
1  /**
2   * Ini adalah fungsi destruktur dalam PHP yang akan mengeprint pesan saat objek kelas orang
3   * dihancurkan.
4   */
5  function __destruct()
6  {
7      echo "Object person $this->name is destroyed" . PHP_EOL;
8  }
```

## 3. Inheritance

```
1  class Manager
2  {
3      /**
4       * `var string;` mendeklarasikan properti yang akan diberikan di __construct dengan tipe data string di
5       * Kelas "Manager".*/
6       var string $name;
7
8       var string $title;
9
10      /**
11       * Ini adalah fungsi konstruktor dalam PHP yang menginisialisasi nama dan sifat judul dari
12       * objek.
13       *
14       * @param string name Parameter nama adalah string yang mewakili nama objek yang dideklarasikan.
15       * Ini memiliki nilai default dari string kosong, yang berarti bahwa jika tidak ada nilai yang disediakan
16       * Untuk parameter nama, itu akan diinisialisasi sebagai string kosong.
17       * @param string title parameter "title" adalah string yang mewakili title pekerjaan dari
18       * orang sedang dibangun .Nilai defaultnya adalah "Manager".
19       */
20      public function __construct(string $name = "", string $title = "Manager")
21      {
22          $this->name = $name;
23          $this->title = $title;
24      }
25
26      /**
27       * Ini adalah fungsi PHP yang mengambil parameter string dan menampilkan pesan ucapan dengan name
28       * Parameter dan nama manajer.
29       *
30       * @param name string parameter "name" adalah parameter tipe string yang mewakili nama
31       * Orang yang akan disapanya.
32       */
33      function sayHello(string $name): void
34      {
35          echo "Hi $name, my name is Manager $this->name" . PHP_EOL;
36      }
37  }
38
39  /* Kelas VicePresident memperluas kelas manajer dan memiliki metode sayHello() yang menyapa seseorang
40  dengan nama VP.*/
41  class VicePresident extends Manager
42  {
43      public function __construct(string $name = "")
44      {
45          // tidak wajib, tapi direkomendasikan
46          parent::__construct($name, "VP");
47      }
48
49      function sayHello(string $name): void
50      {
51          echo "Hi $name, my name is VP $this->name" . PHP_EOL;
52      }
53  }
54
```

```

1  require_once "data/Manager.php";
2
3  /* Kode ini membuat instance baru dari kelas 'Manager', mengatur properti 'name' dari
4  objek dengan "Budi", dan memanggil fungsi 'sayHello()' dengan objek sebagai argumen dan "joko" sebagai
5  parameter. Ini akan menghasilkan pesan ucapan yang mencakup nama "Joko" dan nama
6  Manager, yaitu "Budi".*/
7  $manager = new Manager();
8  $manager->name = "Budi";
9  $manager->sayHello("Joko");
10
11 /* Kode ini membuat instance baru dari kelas 'VicePresident' dan menugaskannya ke variabel
12 '$vp'. Kemudian, ia menetapkan properti 'name' dari objek '$vp' menjadi "Eko". Akhirnya, memanggil fungsi
13 'sayHello()' dengan metode objek '$vp' dengan argumen "joko".*/
14 $vp = new VicePresident();
15 $vp->name = "Eko";
16 $vp->sayHello("Joko");

```

```

Oriented Programming>                                php .\Inheritance.php
Hi Joko, my name is Manager Budi
Hi Joko, my name is VP Eko

```

#### 4. Namespace

```

1  namespace Data\One {
2      class Conflict
3      {
4
5      }
6
7      class Sample
8      {
9
10     }
11
12     class Dummy
13     {
14
15     }
16 }
17
18 namespace Data\Two {
19     class Conflict
20     {
21
22     }
23 }

```

```

1  require_once "data/Conflict.php";
2
3      $conflict1 = new Data\One\Conflict();
4      $conflict2 = new Data\Two\Conflict();

```



```
1 namespace Helper;
2
3 function helpMe()
4 {
5     echo "HELP ME" . PHP_EOL;
6 }
7
8 const APPLICATION = "Belajar PHP OOP";
```



```
1 namespace {
2     echo "Hello Global Namespace" . PHP_EOL;
3 }
4
```

## 5. Import



```
1 require_once "data/Conflict.php";
2 require_once "data/Helper.php";
3
4 use Data\One\Conflict;
5 use function Helper\helpMe;
6 use const Helper\APPLICATION;
7
8 $conflict1 = new Conflict();
9 $conflict2 = new Data\Two\Conflict();
10
11 helpMe();
12
13 echo APPLICATION . PHP_EOL;
14
```



```
1  <?php
2
3  require_once "data/Conflict.php";
4  require_once "data/Helper.php";
5
6  use Data\One\Conflict as Conflict1;
7  use Data\Two\Conflict as Conflict2;
8  use function Helper\helpMe as help;
9  use const Helper\APPLICATION as APP;
10
11 $conflict1 = new Conflict1();
12 $conflict2 = new Conflict2();
13
14 help();
15
16 echo APP . PHP_EOL;
17
```



```
1  <?php
2
3  require_once "data/Conflict.php";
4  require_once "data/Helper.php";
5
6  use Data\One\{Conflict as Conflict1, Dummy, Sample};
7  use function Helper\{helpMe};
8
9  $conflict = new Conflict1();
10 $dummy = new Dummy();
11 $sample = new Sample();
```

## 6. Visibility

```
1  <?php
2
3  class Product
4  {
5      protected string $name;
6      protected int $price;
7
8      public function __construct(string $name, int $price)
9      {
10         $this->name = $name;
11         $this->price = $price;
12     }
13
14     public function getName(): string
15     {
16         return $this->name;
17     }
18
19     public function getPrice(): int
20     {
21         return $this->price;
22     }
23 }
24
25 class ProductDummy extends Product
26 {
27
28     public function info()
29     {
30         echo "Name $this->name" . PHP_EOL;
31         echo "Price $this->price" . PHP_EOL;
32     }
33 }
34 }
```

```
1  require_once "data/Product.php";
2
3  $product = new Product("Apple", 20000);
4
5  echo $product->getName() . PHP_EOL;
6  echo $product->getPrice() . PHP_EOL;
7
8  $dummy = new ProductDummy("Dummy", 1000);
9  $dummy->info();
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Visibility.php
Apple
20000
Name Dummy
Price 1000
```

## 7. Function Overriding

```
1 class Manager
2 {
3     /* 'var string;' mendeklarasikan properti yang akan diberikan di __construct dengan tipe data string di
4     Kelas 'Manager'.*/
5     var string $name;
6
7     var string $title;
8
9     /**
10    * Ini adalah fungsi konstruktor dalam PHP yang menginisialisasi nama dan sifat judul dari
11    * objek.
12    *
13    * @param string name Parameter nama adalah string yang mewakili nama objek yang dideklarasikan.
14    * Ini memiliki nilai default dari string kosong, yang berarti bahwa jika tidak ada nilai yang disediakan
15    * Untuk parameter nama, itu akan diinisialisasi sebagai string kosong.
16    * @param string title parameter "title" adalah string yang mewakili title pekerjaan dari
17    * orang sedang dibangun .Nilai defaultnya adalah "Manager".
18    */
19    public function __construct(string $name = "", string $title = "Manager")
20    {
21        $this->name = $name;
22        $this->title = $title;
23    }
24
25    /**
26    * Ini adalah fungsi PHP yang mengambil parameter string dan menampilkan pesan ucapan dengan name
27    * Parameter dan nama manajer.
28    *
29    * @param name string parameter "name" adalah parameter tipe string yang mewakili nama
30    * Orang yang akan disapanya.
31    */
32    function sayHello(string $name): void
33    {
34        echo "Hi $name, my name is Manager $this->name" . PHP_EOL;
35    }
36 }
37
38 /* Kelas VicePresident memperluas kelas manajer dan memiliki metode sayHello() yang menyapa seseorang
39 dengan nama VP.*/
40 class VicePresident extends Manager
41 {
42
43     public function __construct(string $name = "")
44     {
45         // tidak wajib, tapi direkomendasikan
46         parent::__construct($name, "VP");
47     }
48
49     function sayHello(string $name): void
50     {
51         echo "Hi $name, my name is VP $this->name" . PHP_EOL;
52     }
53 }
54
```

```
1 /* Kode ini membuat instance baru dari kelas 'Manager', mendeklarasikan properti 'name' dari objek dengan "Budi", dan memanggil fungsi 'sayHello()' dengan argumen "Joko".*/
2 $manager = new Manager();
3 $manager->name = "Budi";
4 $manager->sayHello("Joko");
5
6 $vp = new VicePresident();
7 $vp->name = "Eko";
8 $vp->sayHello("Joko");
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman Ke 5 & 6\PHP Object Oriented Programming> php ".\FunctionOverriding.php"
Hi Joko, my name is Manager Budi
Hi Joko, my name is VP Eko
```

## 8. Parent Keyword

```

1  namespace Data;
2
3  class Shape
4  {
5
6      public function getCorner()
7      {
8          return -1;
9      }
10 }
11
12 class Rectangle extends Shape
13 {
14
15     public function getCorner()
16     {
17         return 4;
18     }
19
20     public function getParentCorner()
21     {
22         return parent::getCorner();
23     }
24 }
25
26 }
27

```

```

1  require_once "data/Shape.php";
2
3  use Data\{Shape, Rectangle};
4
5  $shape = new Shape();
6  echo $shape->getCorner() . PHP_EOL;
7
8  $rectangle = new Rectangle();
9  echo $rectangle->getCorner() . PHP_EOL;
10 echo $rectangle->getParentCorner() . PHP_EOL;
11

```

```

uan Ke 5 & 6\PHP Object Oriented Programming> php .\Parent.php
-1
4
-1

```



## 9. Constructor Overriding

```
1 class Manager
2 {
3     /* 'var string;' mendeklarasikan properti yang akan diberikan di __construct dengan tipe data string di
4     kelas 'Manager'.*/
5     var string $name;
6
7     var string $title;
8
9     /**
10     * Ini adalah fungsi konstruktor dalam PHP yang menginisialisasi nama dan sifat judul dari
11     * objek.
12     *
13     * @param string name Parameter nama adalah string yang mewakili nama objek yang dideklarasikan.
14     * Ini memiliki nilai default dari string kosong, yang berarti bahwa jika tidak ada nilai yang disediakan
15     * Untuk parameter nama, itu akan diinisialisasi sebagai string kosong.
16     * @param string title parameter "title" adalah string yang mewakili title pekerjaan dari
17     * orang sedang dibangun .Nilai defaultnya adalah "Manager".
18     */
19     public function __construct(string $name = "", string $title = "Manager")
20     {
21         $this->name = $name;
22         $this->title = $title;
23     }
24 }
```

```
1 class VicePresident extends Manager
2 {
3
4     public function __construct(string $name = "")
5     {
6         // tidak wajib, tapi direkomendasikan
7         parent::__construct($name, "VP");
8     }
9 }
```

## 10. Polymorphism

```
1 class Programmer
2 {
3
4     public string $name;
5
6     public function __construct(string $name)
7     {
8         $this->name = $name;
9     }
10 }
11
12
13 class BackendProgrammer extends Programmer
14 {
15 }
16
17 class FrontendProgrammer extends Programmer
18 {
19 }
20
21 class Company
22 {
23     public Programmer $programmer;
24 }
25
26
27 function sayHelloProgrammer(Programmer $programmer)
28 {
29     if ($programmer instanceof BackendProgrammer) {
30         echo "Hello Backend Programmer $programmer->name" . PHP_EOL;
31     } else if ($programmer instanceof FrontendProgrammer) {
32         echo "Hello Frontend Programmer $programmer->name" . PHP_EOL;
33     } else if ($programmer instanceof Programmer) {
34         echo "Hello Programmer $programmer->name" . PHP_EOL;
35     }
36 }
```

```
1 require_once "data/Programmer.php";
2
3 $company = new Company();
4 $company->programmer = new Programmer("Eko");
5 var_dump($company);
6
7 $company->programmer = new BackendProgrammer("Eko");
8 var_dump($company);
9
10 $company->programmer = new FrontendProgrammer("Eko");
11 var_dump($company);
12
13 sayHelloProgrammer(new Programmer("Eko"));
14 sayHelloProgrammer(new BackendProgrammer("Eko"));
15 sayHelloProgrammer(new FrontendProgrammer("Eko"));
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Polymorphism.php
object(Company)#1 (1) {
    ["programmer"]=>
    object(Programmer)#2 (1) {
        ["name"]=>
        string(3) "Eko"
    }
}
Hello Programmer Eko
Hello Backend Programmer Eko
Hello Frontend Programmer Eko
```

## 11. Type Check & Casts

```
1 function sayHelloProgrammer(Programmer $programmer)
2 {
3     if ($programmer instanceof BackendProgrammer) {
4         echo "Hello Backend Programmer $programmer->name" . PHP_EOL;
5     } else if ($programmer instanceof FrontendProgrammer) {
6         echo "Hello Frontend Programmer $programmer->name" . PHP_EOL;
7     } else if ($programmer instanceof Programmer) {
8         echo "Hello Programmer $programmer->name" . PHP_EOL;
9     }
10 }
```

## 12. Abstract Class

```
1 namespace Data;
2
3 abstract class Location
4 {
5
6     public string $name;
7
8 }
9
10 class City extends Location
11 {
12 }
13
14 class Province extends Location
15 {
16 }
17
18 class Country extends Location
19 {
20 }
```

```
1 require_once "data/Location.php";
2
3 use Data\{Location, City, Province, Country};
4
5 $location = new Location();
6 $city = new City();
7 $province = new Province();
8 $country = new Country();
```

### 13. Abstract Function

```
1 namespace Data;
2
3 require_once "Food.php";
4
5 abstract class Animal
6 {
7     public string $name;
8
9     abstract public function run(): void;
10
11     abstract public function eat(AnimalFood $animalFood): void;
12 }
13
14 class Cat extends Animal
15 {
16     public function run(): void
17     {
18         echo "Cat $this->name is running" . PHP_EOL;
19     }
20
21     public function eat(AnimalFood $animalFood): void
22     {
23         echo "Cat is eating" . PHP_EOL;
24     }
25 }
```

```
1 require_once "data/Animal.php";
2
3 use Data\{Animal, Cat, Dog};
4
5 $cat = new Cat();
6 $cat->name = "Luna";
7 $cat->run();
8
9 $dog = new Dog();
10 $dog->name = "Doggy";
11 $dog->run();
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\AbstractFunction.php
Cat Luna is running
Dog Doggy is running
```

## 14. Getter dan Setter


```
1 require_once "data/Category.php";
2
3 $category = new Category();
4 $category->setName("Handphone");
5 $category->setExpensive(true);
6
7 $category->setName(" ");
8 echo "Name : {$category->getName()}" . PHP_EOL;
9 echo "Expensive : {$category->isExpensive()}" . PHP_EOL;
```

```
1 class Category
2 {
3     private string $name;
4     private bool $expensive;
5
6     public function getName(): string
7     {
8         return $this->name;
9     }
10
11    public function setName(string $name): void
12    {
13        if(trim($name) != ""){
14            $this->name = $name;
15        }
16    }
17
18    public function isExpensive(): bool
19    {
20        return $this->expensive;
21    }
22
23    public function setExpensive(bool $expensive): void
24    {
25        $this->expensive = $expensive;
26    }
27 }
```


```
1 public function setName(string $name): void
2 {
3     if(trim($name) != ""){
4         $this->name = $name;
5     }
6 }
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\GetterAndSetter.php
Name : Handphone
Expensive : 1
```

## 15. Interface



```
1 interface Car extends HasBrand
2 {
3     function drive(): void;
4
5     function getTire(): int;
6 }
```



```
1 class Avanza implements Car, IsMaintenance
2 {
3
4     public function drive(): void
5     {
6         echo "Drive Avanza" . PHP_EOL;
7     }
8
9     public function getTire(): int
10    {
11        return 4;
12    }
13
14    public function getBrand(): string
15    {
16        return "Toyota";
17    }
18
19    public function isMaintenance(): bool
20    {
21        return false;
22    }
23
24 }
```

## 16. Interface Inheritance

```
1 namespace Data;
2
3 interface HasBrand
4 {
5     function getBrand(): string;
6 }
7
8 interface IsMaintenance
9 {
10     function isMaintenance(): bool;
11 }
12
13 interface Car extends HasBrand
14 {
15     function drive(): void;
16     function getTire(): int;
17 }
18
19 class Avanza implements Car, IsMaintenance
20 {
21     {
22
23         public function drive(): void
24         {
25             echo "Drive Avanza" . PHP_EOL;
26         }
27
28         public function getTire(): int
29         {
30             return 4;
31         }
32
33         public function getBrand(): string
34         {
35             return "Toyota";
36         }
37
38         public function isMaintenance(): bool
39         {
40             return false;
41         }
42     }
43 }
```

```
1 class Avanza implements Car, IsMaintenance
2 {
3
4     public function drive(): void
5     {
6         echo "Drive Avanza" . PHP_EOL;
7     }
8
9     public function getTire(): int
10    {
11        return 4;
12    }
13
14    public function getBrand(): string
15    {
16        return "Toyota";
17    }
18
19    public function isMaintenance(): bool
20    {
21        return false;
22    }
23
24 }
```

## 17. Trait

```
1  /* Fungsi 'HasName' mendefinisikan properti publik '$name' dari tipe string. Sifat ini dapat digunakan oleh
2  kelas yang perlu memiliki properti nama.*/
3  trait HasName
4  {
5      public string $name;
6  }
7  class Person
8  {
9      use SayGoodBye, SayHello;
10 }
```

```
1  class Person
2  {
3      use SayGoodBye, SayHello;
4  }
```

```
1  /* Fungsi 'SayGoodBye' mendefinisikan metode 'goodBye' yang mengambil parameter string opsional '$name'.
2  Jika '$name' nol, itu mengeprint "Good bye" ke konsol. Kalau tidak, itu mengeprint "Good bye" diikuti oleh
3  nilai '$name'. Sifat ini dapat digunakan oleh kelas yang perlu menerapkan metode 'Good bye' .*/
4
5  trait SayGoodBye
6  {
7      public function goodBye(?string $name): void
8      {
9          if (is_null($name)) {
10             echo "Good bye" . PHP_EOL;
11          } else {
12             echo "Good bye $name" . PHP_EOL;
13          }
14      }
15 }
```

```
1  require_once "data/SayGoodBye.php";
2
3  use Data\Traits\{Person, SayHello, SayGoodBye};
4
5  $person = new Person();
6  $person->name = "Eko";
7  echo $person->name . PHP_EOL;
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Trait.php
Eko
```



## 18. Trait Overriding

```
1 class Parent
2 {
3
4     use SayGoodBye, SayHello;
5
6     public function goodBye(?string $name): void
7     {
8         echo "Good bye in Person" . PHP_EOL;
9     }
10
11     public function hello(?string $name): void
12     {
13         echo "Hello in Person" . PHP_EOL;
14     }
15 }
```

```
1 trait CanRun
2 {
3     public abstract function run(): void;
4 }
5 class Person
6 {
7     use SayGoodBye, SayHello, HasName, CanRun;
8
9     public function run(): void
10    {
11        echo "Person $this->name is running" . PHP_EOL;
12    }
13 }
```

```
1 class Person
2 {
3     use SayGoodBye, SayHello, HasName, CanRun {
4         hello as private;
5         goodBye as private;
6     }
7 }
```

## 19. Trait Conflict

```
1  /* Sifat 'A' mendefinisikan dua metode 'doA()' dan 'doB()' yang akan digunakan di kelas yang menggunakan
2  sifat ini. Metode 'doA()' akan mengeluarkan string "a" diikuti dengan baris baru, dan metode 'doB()'
3  akan mengeluarkan string "B" diikuti oleh baris baru.*/
4  trait A
5  {
6      function doA(): void
7      {
8          echo "a" . PHP_EOL;
9      }
10
11     function doB(): void
12     {
13         echo "b" . PHP_EOL;
14     }
15 }
```

```
1  /* Fungsi 'B' mendefinisikan dua metode 'doA()' dan 'doB()' yang akan digunakan di kelas yang menggunakan
2  sifat ini. Metode 'doA()' akan mengeluarkan string "a" diikuti oleh newline, dan metode 'doB()'
3  akan mengeluarkan string "B" diikuti oleh garis baru.*/
4  trait B
5  {
6      function doA(): void
7      {
8          echo "A" . PHP_EOL;
9      }
10
11     function doB(): void
12     {
13         echo "B" . PHP_EOL;
14     }
15 }
```

```
1  /* TraitConflict Class menggunakan fungsi A dan B, menyelesaikan konflik antara metode mereka dengan kata kunci insteadof.*/
2  class TraitConflict
3  {
4      use A, B {
5          A::doA insteadof B;
6          B::doB insteadof A;
7      }
8  }
9
10 $sample = new TraitConflict();
11 $sample->doA();
12 $sample->doB();
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\TraitConflict.php
a
B
```

## 20. Trait Inheritance

```
1 trait All
2 {
3     use SayGoodBye, SayHello, HasName, CanRun;
4 }
5
6
7 class Person extends ParentPerson
8 {
9     use All;
10
11     public function run(): void
12     {
13         echo "Person $this->name is running" . PHP_EOL;
14     }
15 }
```

## 21. Final Class

```
1 class SocialMedia
2 {
3     public string $name;
4 }
5
6 class Facebook extends SocialMedia
7 {
8
9 }
10
11 // error
12 class FakeFacebook extends Facebook
13 {
14
15 }
```

## 22. Final Function

```
1 class Facebook extends SocialMedia
2 {
3     final public function login(string $username, string $password): bool
4     {
5
6     }
7 }
8
9 // error
10 class FakeFacebook extends Facebook
11 {
12     // error
13     public function login(string $username, string $password): bool
14     {
15
16     }
17 }
```

## 23. Anonymous Class

```
1 interface HelloWorld
2 {
3     function sayHello(): void;
4 }
5
6 $helloWorld = new class("Eko") implements HelloWorld
7 {
8     public function sayHello(): void
9     {
10         echo "Hello Anonymous Class" . PHP_EOL;
11     }
12 };
13
14 $helloWorld->sayHello();
15
```

```
uan Ke 5 & 6\PHP Object Oriented Programming> php .\AnonymousClass.php
Hello Anonymous Class
```

```

1 interface HelloWorld
2 {
3     function sayHello(): void;
4 }
5
6 $helloWorld = new class("Eko") implements HelloWorld
7 {
8
9     public string $name;
10
11     public function __construct(string $name)
12     {
13         $this->name = $name;
14     }
15
16     public function sayHello(): void
17     {
18         echo "Hello $this->name" . PHP_EOL;
19     }
20 };
21
22 $helloWorld->sayHello();

```

uan Ke 5 & 6\PHP Object Oriented Programming> **php** .\AnonymousClass.php  
Hello Eko

## 24. Static Keyword

```

1 class MathHelper
2 {
3     static public string $name = "MathHelper";
4 }
5
6 echo MathHelper::$name . PHP_EOL;

```

uan Ke 5 & 6\PHP Object Oriented Programming> **php** .\helper\MathHelper.php  
MathHelper

```

1 class MathHelper
2 {
3     static public string $name = "MathHelper";
4
5     static public function sum(int ...$numbers): int
6     {
7         $total = 0;
8         foreach ($numbers as $number) {
9             $total += $number;
10        }
11        return $total;
12    }
13 }
14
15 $total = MathHelper::sum(1, 2, 3, 4, 5);
16 echo "Total $total" . PHP_EOL;

```

uan Ke 5 & 6\PHP Object Oriented Programming> **php** .\helper\MathHelper.php  
Total 15

## 25. stdClass

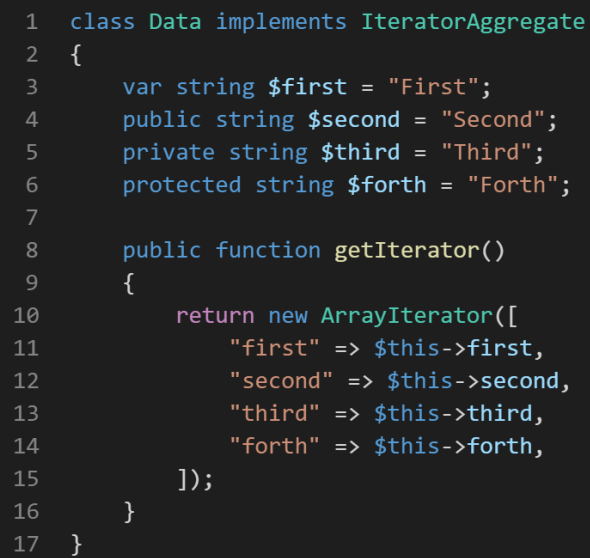
```
1 $array = [  
2     "firstName" => "Eko",  
3     "middleName" => "Kurniawan",  
4     "lastName" => "Khannedy"  
5 ];  
6  
7 $object = (object)$array;  
8  
9 var_dump($object);  
10  
11 echo "First Name $object->firstName" . PHP_EOL;  
12 echo "Middle Name $object->middleName" . PHP_EOL;  
13 echo "Last Name $object->lastName" . PHP_EOL;
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman Ke 5 & 6\PHP Object Oriented Programming> php .\StdClass.php  
object(stdClass)#1 (3) {  
    ["firstName"]=>  
    string(3) "Eko"  
    ["middleName"]=>  
    string(9) "Kurniawan"  
    ["lastName"]=>  
    string(8) "Khannedy"  
}  
First Name Eko  
Middle Name Kurniawan  
Last Name Khannedy
```

## 26. Object Iteration

```
1 class Data  
2 {  
3     var string $first = "First";  
4     public string $second = "Second";  
5     private string $third = "Third";  
6     protected string $forth = "Forth";  
7 }  
8  
9 $data = new Data();  
10  
11 foreach ($data as $property => $value) {  
12     echo "$property : $value" . PHP_EOL;  
13 }
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman Ke 5 & 6\PHP Object Oriented Programming> php .\ObjectIteration.php  
first : First  
second : Second  
third : Third  
forth : Forth
```



```
1 class Data implements IteratorAggregate
2 {
3     var string $first = "First";
4     public string $second = "Second";
5     private string $third = "Third";
6     protected string $forth = "Forth";
7
8     public function getIterator()
9     {
10         return new ArrayIterator([
11             "first" => $this->first,
12             "second" => $this->second,
13             "third" => $this->third,
14             "forth" => $this->forth,
15         ]);
16     }
17 }
```

## 27. Generator



```
1  function getGanjil(int $max): Iterator
2  {
3      for ($i = 1; $i <= $max; $i++) {
4          if ($i % 2 == 1) {
5              yield $i;
6          }
7      }
8  }
9
10 foreach (getGanjil(100) as $value) {
11     echo "Ganjil : $value" . PHP_EOL;
12 }
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-W
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Generator.php
Ganjil : 1
Ganjil : 3
Ganjil : 5
Ganjil : 7
Ganjil : 9
Ganjil : 11
Ganjil : 13
Ganjil : 15
Ganjil : 17
Ganjil : 33
Ganjil : 35
Ganjil : 37
Ganjil : 39
Ganjil : 41
Ganjil : 43
Ganjil : 45
Ganjil : 47
Ganjil : 49
Ganjil : 51
Ganjil : 53
```



## 28. Object Cloning

```
1  $student1 = new Student();
2  $student1->id = "1";
3  $student1->name = "Eko";
4  $student1->value = 100;
5
6
7
8  $student2 = clone $student1;
9  var_dump($student1);
10 var_dump($student2);
```

```
1  class Student
2  {
3      public string $id;
4      public string $name;
5      public int $value;
6
7
8      public function __clone()
9      {
10         unset($this->value);
11     }
12 }
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pem
uan Ke 5 & 6\PHP Object Oriented Programming> php .\ObjectCloning.php
object(Student)#1 (3) {
    ["id"]=>
    string(1) "1"
    ["name"]=>
    string(3) "Eko"
    ["value"]=>
    int(100)
    ["value"]=>
    uninitialized(int)
    ["sample":"Student":private]=>
    uninitialized(string)
}
```

## 29. Comparing Object

```
1  require_once "data/Student.php";
2
3  $student1 = new Student();
4  $student1->id = "1";
5  $student1->name = "Eko";
6  $student1->value = 100;
7
8  $student2 = new Student();
9  $student2->id = "1";
10 $student2->name = "Eko";
11 $student2->value = 100;
12
13 var_dump($student1 == $student2);
14 var_dump($student1 === $student2);
15 var_dump($student1 === $student1);
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrog  
uan Ke 5 & 6\PHP Object Oriented Programming> php .\ComparingObject.php  
bool(true)  
bool(false)  
bool(true)
```

## 30. Magic Function

```
1 class Student
2 {
3     public string $id;
4     public string $name;
5     public int $value;
6     private string $sample;
7
8     public function setSample(string $sample): void
9     {
10         $this->sample = $sample;
11     }
12
13     public function __clone()
14     {
15         unset($this->sample);
16     }
17
18     public function __toString(): string
19     {
20         return "Student id:$this->id, name:$this->name, value:$this->value";
21     }
22
23     public function __invoke(...$arguments): void
24     {
25         $join = join(", ", $arguments);
26         echo "Invoke student with arguments $join" . PHP_EOL;
27     }
28
29     public function __debugInfo()
30     {
31         return [
32             "id" => $this->id,
33             "name" => $this->name,
34             "value" => $this->value,
35             "sample" => $this->sample,
36             "author" => "Eko",
37             "version" => "1.0.0"
38         ];
39     }
40 }
41
```

## 31. Overloading

```
1 class Zero
2 {
3     private array $properties = [];
4
5     public function __get($name)
6     {
7         return $this->properties[$name];
8     }
9
10    public function __set($name, $value)
11    {
12        $this->properties[$name] = $value;
13    }
14
15    public function __isset($name): bool
16    {
17        return isset($this->properties[$name]);
18    }
19
20    public function __unset($name)
21    {
22        unset($this->properties[$name]);
23    }
24
25    public function __call($name, $arguments)
26    {
27        $join = join(", ", $arguments);
28        echo "Call function $name with arguments $join". PHP_EOL;
29    }
30
31    public static function __callStatic($name, $arguments)
32    {
33        $join = join(", ", $arguments);
34        echo "Call static function $name with arguments $join". PHP_EOL;
35    }
36
37 }
```

PS D:\Kampus\STMIK\Wateri\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman Web\uan Ke 5 & 6\PHP Object Oriented Programming> php .\PropertiesOverloading.php  
First Name : Eko  
Middle Name : Kurniawan  
Last Name : Khannedy  
Call function sayHello with arguments Eko,Khannedy  
Call static function sayHello with arguments Eko,Khannedy

## 32. Covariance dan Contravariance

```
1 namespace Data;
2
3 require_once "Food.php";
4
5 abstract class Animal
6 {
7     public string $name;
8
9     abstract public function run(): void;
10
11     abstract public function eat(AnimalFood $animalFood): void;
12 }
13
14 class Cat extends Animal
15 {
16     public function run(): void
17     {
18         echo "Cat $this->name is running" . PHP_EOL;
19     }
20
21     public function eat(AnimalFood $animalFood): void
22     {
23         echo "Cat is eating" . PHP_EOL;
24     }
25 }
26
27 class Dog extends Animal
28 {
29     public function run(): void
30     {
31         echo "Dog $this->name is running" . PHP_EOL;
32     }
33
34     public function eat(Food $animalFood): void
35     {
36         echo "Dog is eating" . PHP_EOL;
37     }
38 }
```

```
1 namespace Data;
2
3 require_once "Animal.php";
4
5 interface AnimalShelter
6 {
7     function adopt(string $name): Animal;
8 }
9
10 class CatShelter implements AnimalShelter
11 {
12     public function adopt(string $name): Cat
13     {
14         $cat = new Cat();
15         $cat->name = $name;
16         return $cat;
17     }
18 }
19
20 class DogShelter implements AnimalShelter
21 {
22     public function adopt(string $name): Dog
23     {
24         $dog = new Dog();
25         $dog->name = $name;
26         return $dog;
27     }
28 }
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemro
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Covariance.php
Cat is eating
Dog is eating
```

### 33. DateTime

```
1 $dateTime = new DateTime();
2 $dateTime->setDate(1990, 1, 20);
3 $dateTime->setTime(10, 10, 10, 0);
4
5 $dateTime->add(new DateInterval("P1Y"));
6
7 $minusOneMonth = new DateInterval("P1M");
8 $minusOneMonth->invert = true;
9 $dateTime->add($minusOneMonth);
10
11 var_dump($dateTime);
12
13 $now = new DateTime();
14 var_dump($now);
15 $now->setTimezone(new DateTimeZone("America/Toronto"));
16 var_dump($now);
17
18 $string = $now->format("Y-m-d H:i:s");
19 echo "Waktu Saat Ini : $string" . PHP_EOL;
20
21 $date = DateTime::createFromFormat("Y-m-d H:i:s", "2020-10-10 10:10:10", new DateTimeZone("Asia/Jakarta"));
22 if ($date) {
23     var_dump($date);
24 } else {
25     echo "Format Salah" . PHP_EOL;
26 }
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-Web\Tugas 5 & 6\PHP Object Oriented Programming> php .\DateTime.php
object(DateTime)#1 (3) {
    ["date"]=>
    string(26) "2023-06-07 01:25:22.493830"
    ["timezone_type"]=>
    int(3)
    ["timezone"]=>
    string(15) "America/Toronto"
}
Waktu Saat Ini : 2023-06-07 01:25:22
object(DateTime)#5 (3) {
    ["date"]=>
    string(26) "2020-10-10 10:10:10.000000"
    ["timezone_type"]=>
    int(3)
    ["timezone"]=>
    string(12) "Asia/Jakarta"
}
```

## 34. Exception

```
1 class ValidationException extends Exception {
2
3 }
```

```
1 function validateLoginRequest(LoginRequest $request)
2 {
3     if (!isset($request->username)) {
4         throw new ValidationException("Username is null");
5     } else if (!isset($request->password)) {
6         throw new ValidationException("Password is null");
7     } else if (trim($request->username) == "") {
8         throw new Exception("Username is empty");
9     } else if (trim($request->password) == "") {
10        throw new Exception("Password is empty");
11    }
12 }
```

```
1 require_once "exception/ValidationException.php";
2 require_once "data/LoginRequest.php";
3 require_once "helper/Validation.php";
4
5 $loginRequest = new LoginRequest();
6 $loginRequest->username = " ";
7 $loginRequest->password = " ";
8
9 try {
10     validateLoginRequest($loginRequest);
11     echo "VALID" . PHP_EOL;
12 } catch (ValidationException | Exception $exception) {
13     echo "Error : {$exception->getMessage()}" . PHP_EOL;
14
15     var_dump($exception->getTrace());
16
17     echo $exception->getTraceAsString() . PHP_EOL;
18 } finally {
19     echo "ERROR ATAU ENGGAK, AKAN DIEKSEKUSI" . PHP_EOL;
20 }
21
```

```
P5 D:\Kampus\STMIK\Wateri\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-Web\Praktikum\Pertem
uan Ke 5 & 6\PHP Object Oriented Programming> php .\Exception.php
Error : Username is empty
array(1) {
  [0]=>
  array(4) {
    ["file"]=>
    string(155) "D:\Kampus\STMIK\Wateri\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-Web\Pr
aktikum\Pertemuan Ke 5 & 6\PHP Object Oriented Programming\Exception.php"
    ["line"]=>
    int(12)
    ["function"]=>
    string(20) "validateLoginRequest"
    ["args"]=>
    array(1) {
      [0]=>
      object(LoginRequest)#1 (2) {
        ["username"]=>
        string(2) " "
        ["password"]=>
        string(2) " "
      }
    }
  }
}
#0 D:\Kampus\STMIK\Wateri\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-Web\Praktikum\Pertem
uan Ke 5 & 6\PHP Object Oriented Programming\Exception.php(12): validateLoginRequest(Object(LoginReq
uest))
#1 {main}
ERROR ATAU ENGGAK, AKAN DIEKSEKUSI
```

## 35. Regular Expression

```
1 $matches = [];  
2 $result = preg_match_all("/eko|awan|edy/i", "Eko Kurniawan Khannedy", $matches);  
3  
4 var_dump($result);  
5 var_dump($matches);  
6  
7 $result = preg_replace("/anjing|bangsat/i", "****", "dasar lu ANJING dan BANGSAT!");  
8  
9 var_dump($result);  
10  
11 $result = preg_split("/[\\s,-]/", "Eko Kurniawan Khannedy,Programmer,Zaman-Now");  
12  
13 var_dump($result);
```

```
PS D:\Kampus\STMIK\Materi\Genap 2022-2023\Pemrograman Web\Praktikum\Pemrograman-Web\Praktikum\Pertem  
uan Ke 5 & 6\PHP Object Oriented Programming> php .\RegularExpression.php  
string(9) "Kurniawan"  
[2]=>  
string(8) "Khannedy"  
[3]=>  
string(10) "Programmer"  
[4]=>  
string(5) "Zaman"  
[5]=>  
string(3) "Now"  
}
```

## 36. Reflection

```
1 class ValidationUtil  
2 {  
3     static function validate(LoginRequest $request)  
4     {  
5         if (!isset($request->username)) {  
6             throw new ValidationException("username is not set");  
7         } else if (!isset($request->password)) {  
8             throw new ValidationException("password is not set");  
9         } else if (is_null($request->username)) {  
10            throw new ValidationException("username is null");  
11        } else if (is_null($request->password)) {  
12            throw new ValidationException("password is null");  
13        }  
14    }  
15  
16    static function validateReflection($request)  
17    {  
18        $reflection = new ReflectionClass($request);  
19        $properties = $reflection->getProperties(ReflectionProperty::IS_PUBLIC);  
20        foreach ($properties as $property) {  
21            if (!$property->isInitialized($request)) {  
22                throw new ValidationException("$property->name is not set");  
23            } else if (is_null($property->getValue($request))) {  
24                throw new ValidationException("$property->name is null");  
25            }  
26        }  
27    }  
28 }
```