

MATERI MODULE P.B.O-PEMOGRAMAN WEBSITE MENGGUNAKAN BAHASA PHP

Kelas : 12 RPL
Mapel : P.B.O dan Pemograman Website
Disusun oleh : Hasudungan Sitorus S, Kom

PROLOG	•penjelasan singkat, penggunaan PBO dan kelebihan dalam penggunaan
CLASS-OBJECT	•penjelasan singkat, penggunaan tentang class-objek dan disertai dengan base code
ACCESS MODIFIERS	•penjelasan singkat, penggunaan tentang access modifiers dan disertai dengan base code
INHERITANCE / PEWARISAN	•penjelasan singkat, penggunaan tentang pewarisan properti dan disertai dengan base code
CONSTANTS	•penjelasan singkat, penggunaan tentang constant dan disertai dengan base code
ABSTRACT CLASS	•penjelasan singkat, penggunaan tentang abstract class dan disertai dengan base code
INTERFACE	•penjelasan singkat, penggunaan tentang interface dan disertai dengan base code
TRAIT	•penjelasan singkat, penggunaan tentang trait dan disertai dengan base code
STATIC METHOD	•penjelasan singkat, penggunaan tentang static method dan disertai dengan base code
STATIC PROPERTHIES	•penjelasan singkat, penggunaan tentang static properti dan disertai dengan base code

AUTOLOADING

- penjelasan singkat, penggunaan tentang static properti dan disertai dengan base code

DEPEDENCY INJECTION

- penjelasan singkat, penggunaan tentang static properti dan disertai dengan base code

STUDI KASUS PENGGUNAAN KONSEP PBO(CRUD)

- penjelasan singkat, penggunaan tentang static properti dan disertai dengan base code

#PROLOG

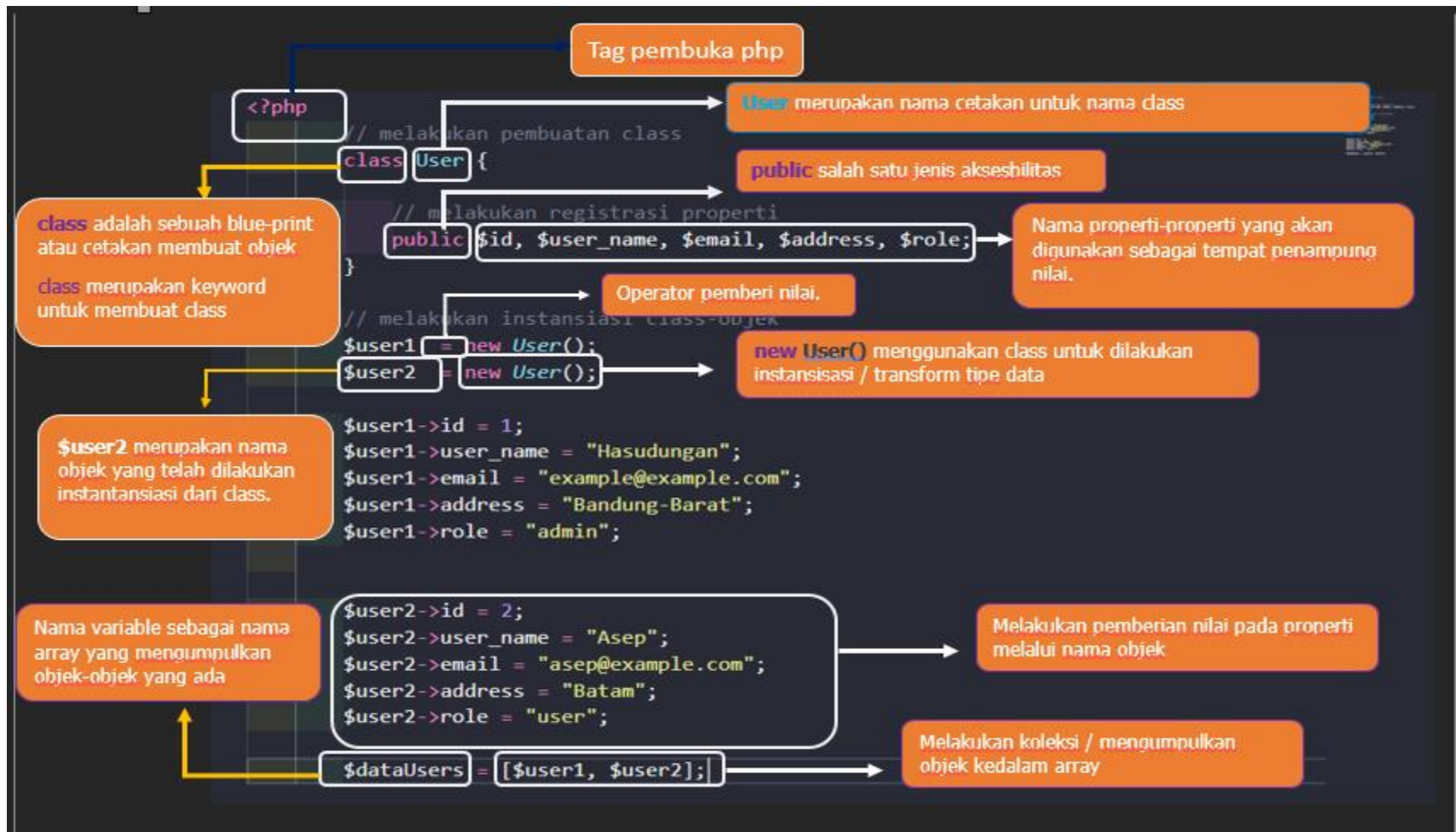
Pemrograman Berorientasi Objek atau Object Oriented Programming (OOP) adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Pemahaman ini mengacu pada teknis dari pemrograman OOP, yakni menyusun, mengolah objek-objek berisi data dan operasi yang dibutuhkan melalui Class dan Method yang dibutuhkan dalam suatu perangkat lunak. Referensi: <https://serupa.id/konsep-pemrograman-berorientasi-objek-pbo-oop/>

Dengan kata lain pemograman berorientasi objek merupakan cara berpikir programmer menyelesaikan / memecahkan masalah berorientasi dasar class,objek-objek serta attribute pendukung lainnya yang digunakan.

#CLASS-OBJEK

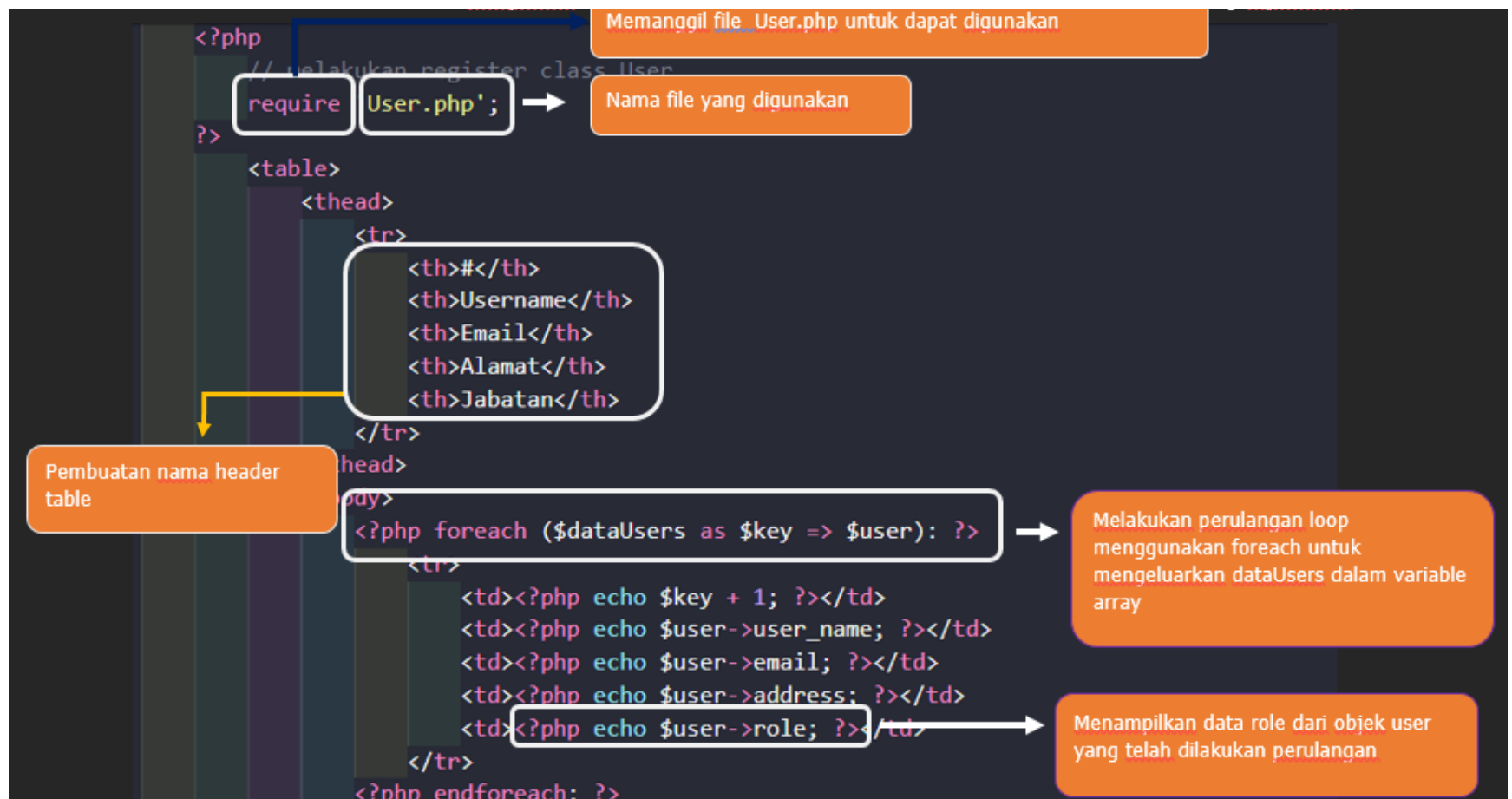
Class adalah template (pola) untuk berbagai obyek dengan fitur serupa. Class mewujudkan semua kumpulan/set fitur tertentu dari objek. Referensi: <https://staffnew.uny.ac.id/upload/132206816/pendidikan/bambangshm-pbo-matericlassobject.pdf#:~:text=Class%20adalah%20template%20%28pola%29%20untuk%20berbagai%20obyek%20dengan,Class%20mewujudkan%20semua%20kumpulan%2Fset%20fitur%20tertentu%20dari%20objek.>

Contoh base-code dan penggunaannya.

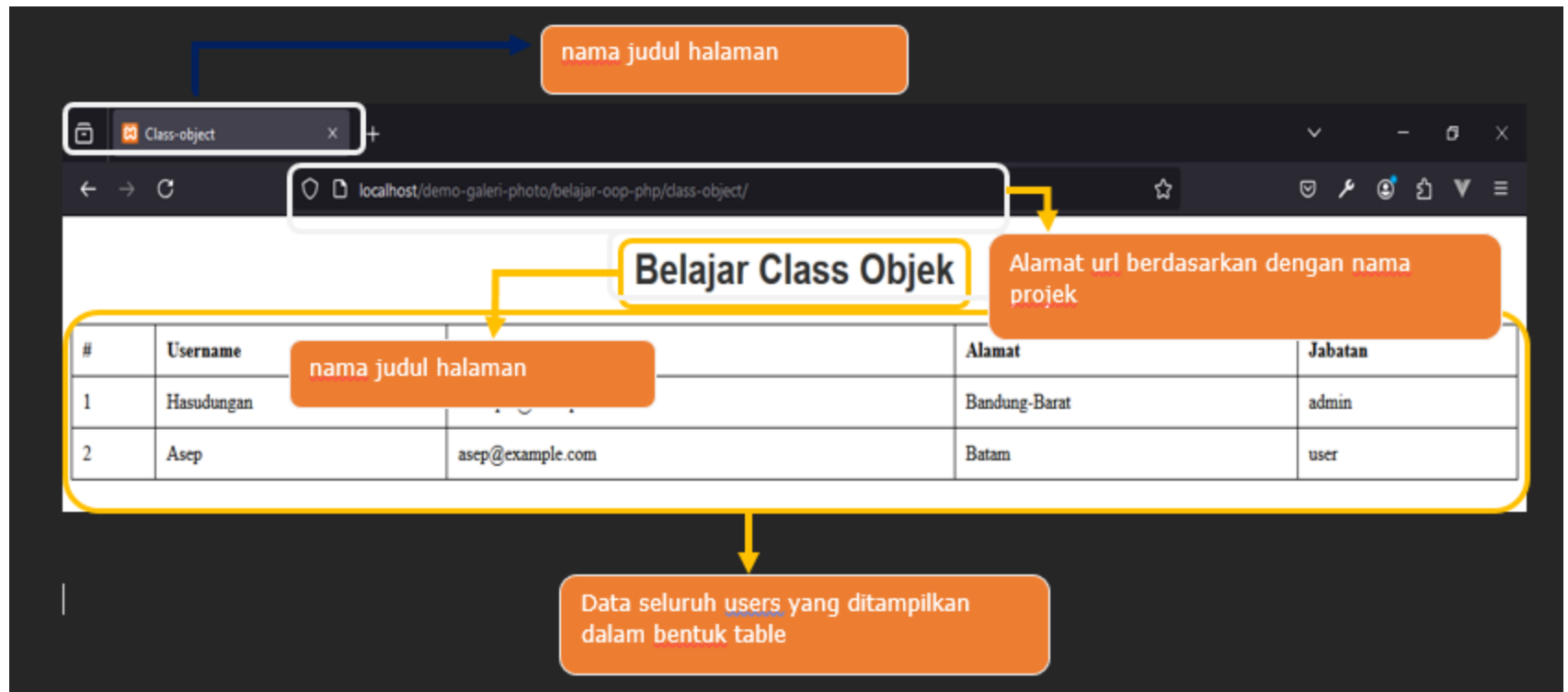


Gambar 1.1 class User.php

Dalam latihan dilakukan, dianjurkan untuk memisahkan antara class dengan front-end serta css yang digunakan. Berikut gambar dari base-code.



Gambar 1.2 index.php

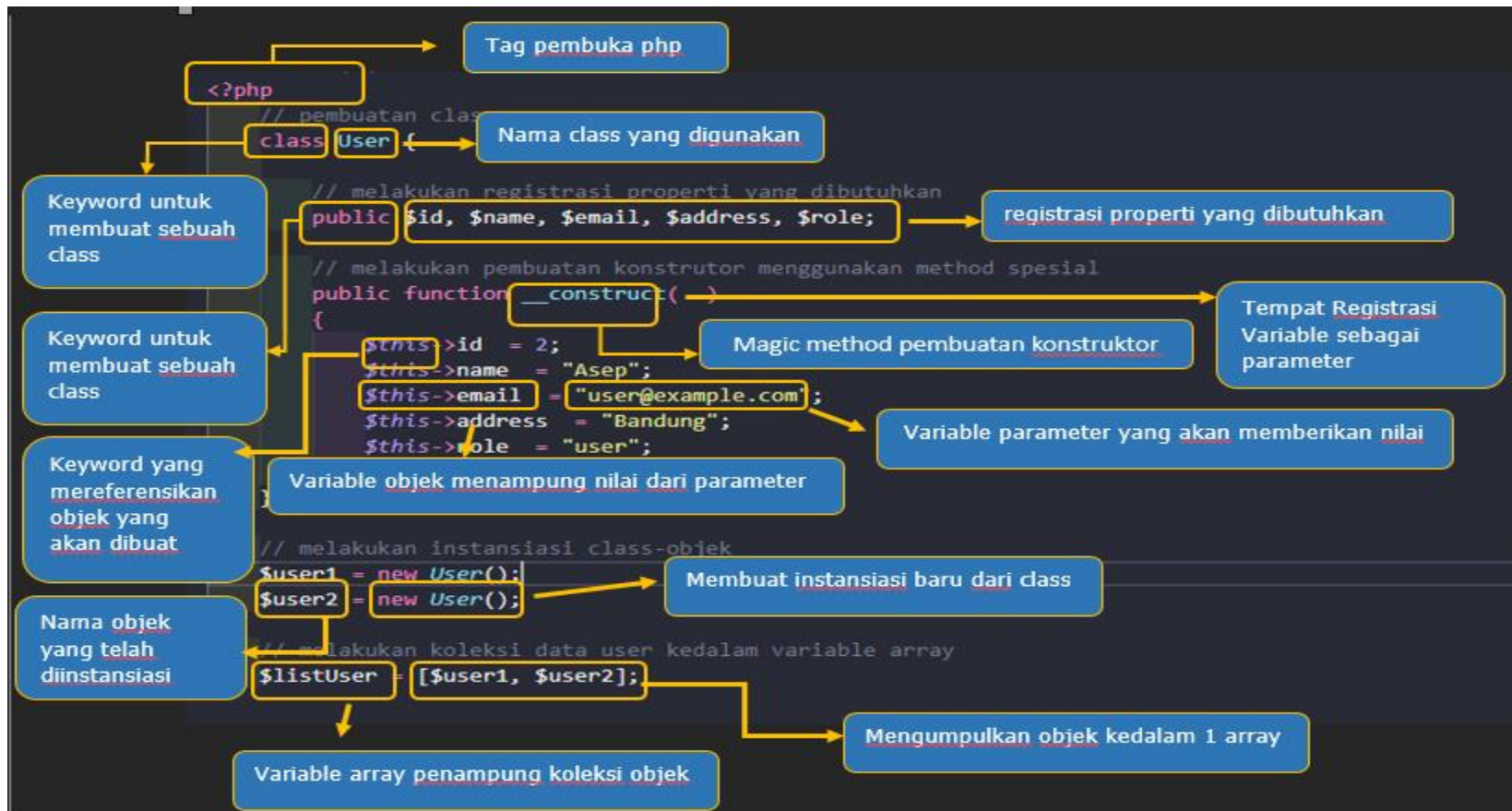


Gambar1.3 Tampilan Halaman index.php

#CONSTRUCTOR

Constructor adalah merupakan konsep pembuatan konstruksi yang memungkinkan melakukan inisialisasi sebuah property yang ketika dilakukan pembuatan objek. Dalam pembuatan Konstruktor yaitu dengan menggunakan fungsi special dari php yang memudahkan inisialisasi yang diinginkan. Berikut potongan kode, penjelasan serta beberapa jenis construct yang dapat dibangun.

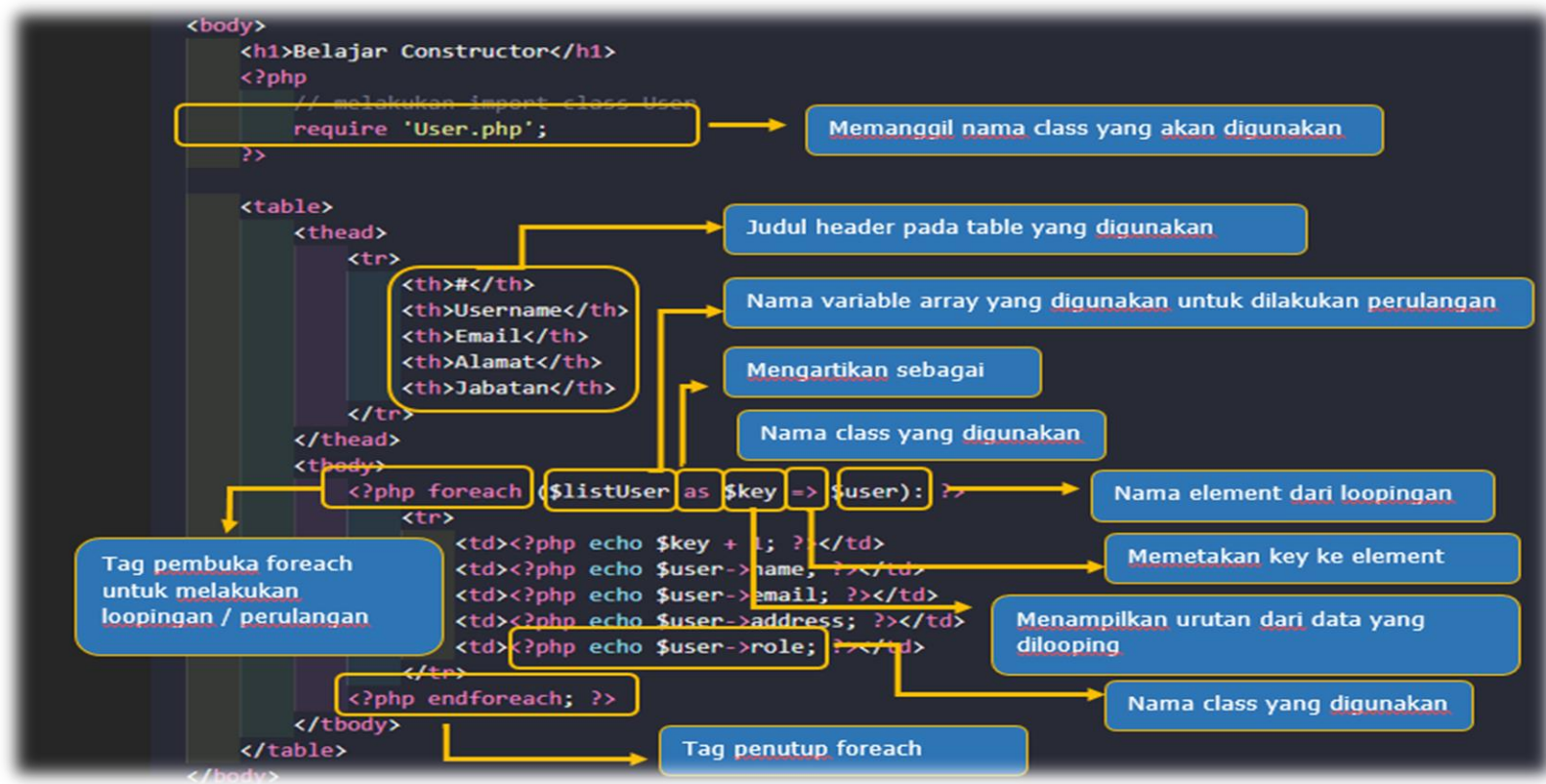
#Constructor dengan nilai default tanpa memberi variable parameter



Gambar 2.1 Constructor dengan parameter

Instansiasi adalah proses pembuatan / transformasi dari class menjadi sebuah objek. Dalam contoh base-code diatas, kita melakukan konstruktor dengan nilai default namun tidak memiliki variable parameter didalam method construct

yang kita gunakan. Setelah itu dilanjutkan dengan melakukan instansiasi class objek serta kita sudah memiliki 2 objek yang dikumpulkan dalam 1 variable array, berikut base-code untuk menampilkan data user



Gambar 2.2 index.php sebagai code menampilkan ke dalam tampilan halaman

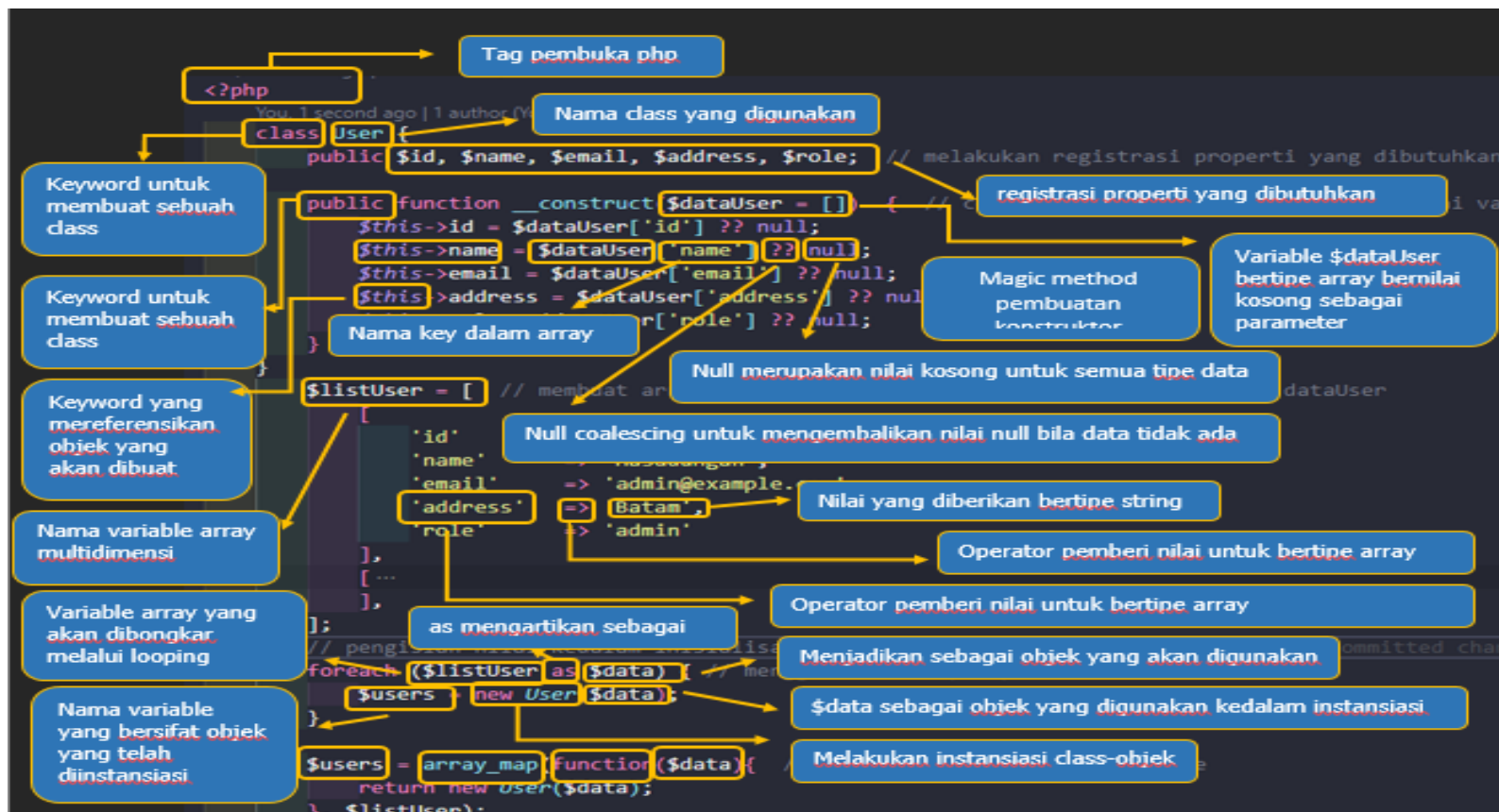


Gambar 2.3 Tampilan Halaman table list data user

Berdasarkan data user yang ditampilkan, data user ada 2 atau objek yang telah diinstansiasi ada 2 namun data yang ditampilkan sama dikarenakan nilai yang dalam construct telah diberi nilai default.

#Constructor dengan nilai default dalam parameter

Dalam kondisi tertentu, nilai default dalam parameter dengan tipe data array didalam method construct diperlukan agar bila tidak ada nilai yang akan diberi ketika melakukan instansiasi class-objek dilakukan. Berikut contoh potongan code dibawah ini.



Gambar 2.4 Constructor dengan variable parameter bertipe array kosong

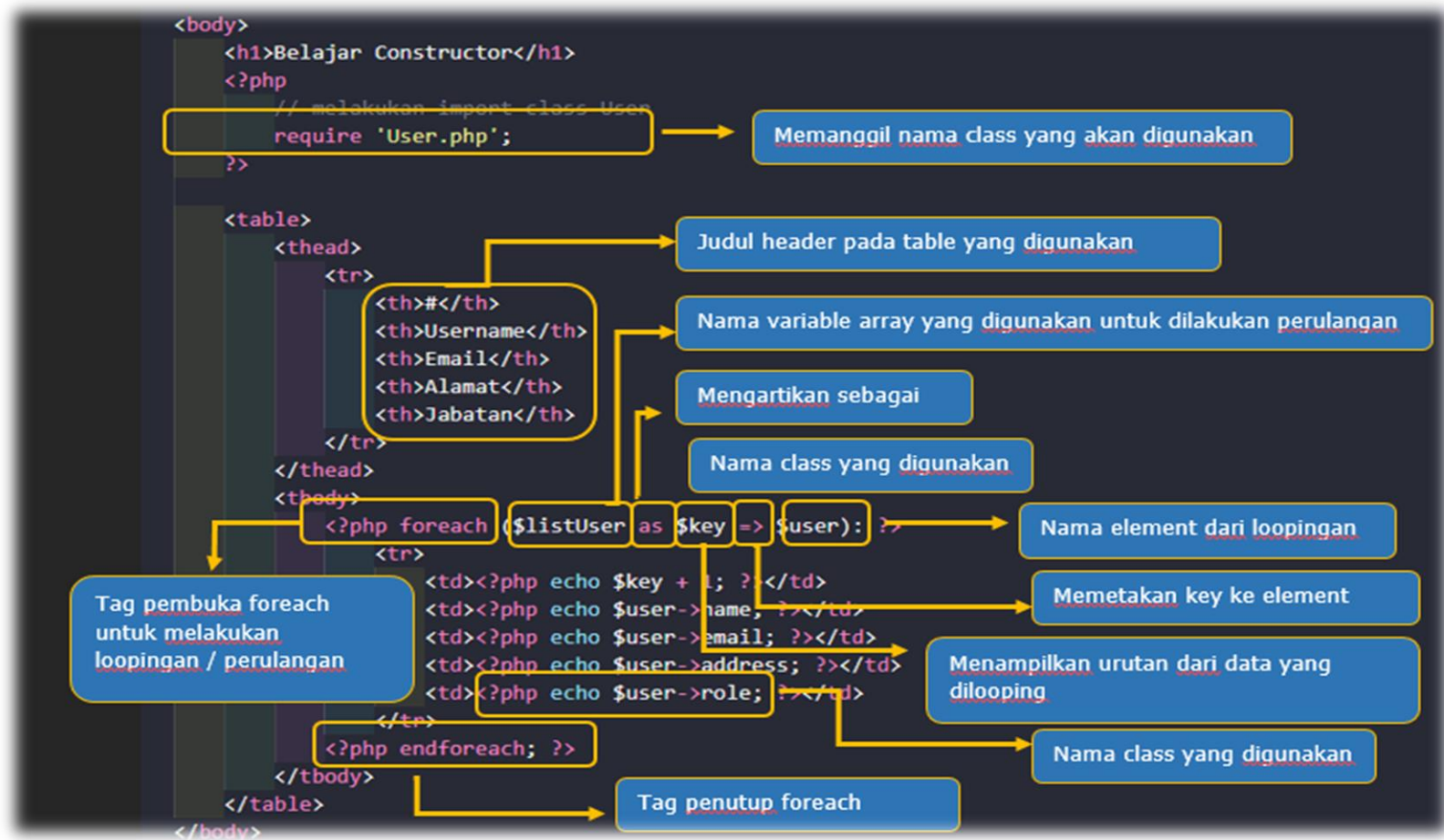
The screenshot shows a web browser window with the following elements and annotations:

- Page Title:** The browser tab is labeled "Belajar Konstruktor". An annotation box labeled "Judul halaman" points to this tab.
- URL:** The address bar shows "localhost/demo-galeri-photo/belajar-oop-php/constructor/". An annotation box labeled "Alamat url berdasarkan nama proyek" points to the URL.
- Main Content Title:** The page header displays "Belajar Constructor". An annotation box labeled "Judul Main Content" points to this text.
- User Data Table:** A table with 5 columns: #, Username, Email, Alamat, and Jabatan. It contains two rows of user data.

#	Username	Email	Alamat	Jabatan
1	Hasudungan	admin@example.com	Batam	admin
2	test user	testUser@example.com	batam	user
- Data Description:** A large blue annotation box at the bottom explains the data: "Data user yang ditampilkan. Berdasarkan data user yang ditampilkan ada 2 objek, 2 objek tersebut ada yang tidak memiliki argument dan ada yang memiliki argument. data yang diberikan nilai didalam construct yang telah dibuat".

Gambar 2.5 Tampilan Halaman list data user

#Construtor dengan array kosong sebagai parameter



Gambar 2.6 index.php

#ACCESS MODIFIERS / AKSESBIILITAS

Akses modifiers atau aksesibilitas merupakan deklarasi pemberian sifat untuk *class*, *property*, *function*. ada 4 jenis aksesibilitas yang sering digunakan dalam bekerja mengembangkan aplikasi menggunakan konsep Pemograman Berorientasi Objek, yaitu

- Private

Jenis aksesibilitas private pada property ataupun function dalam sebuah class, hanya bisa diakses dan digunakan dalam class itu sendiri, tidak bisa diakses melalui pewarisan atau declass manapun. Contoh penggunaan sebagai berikut:

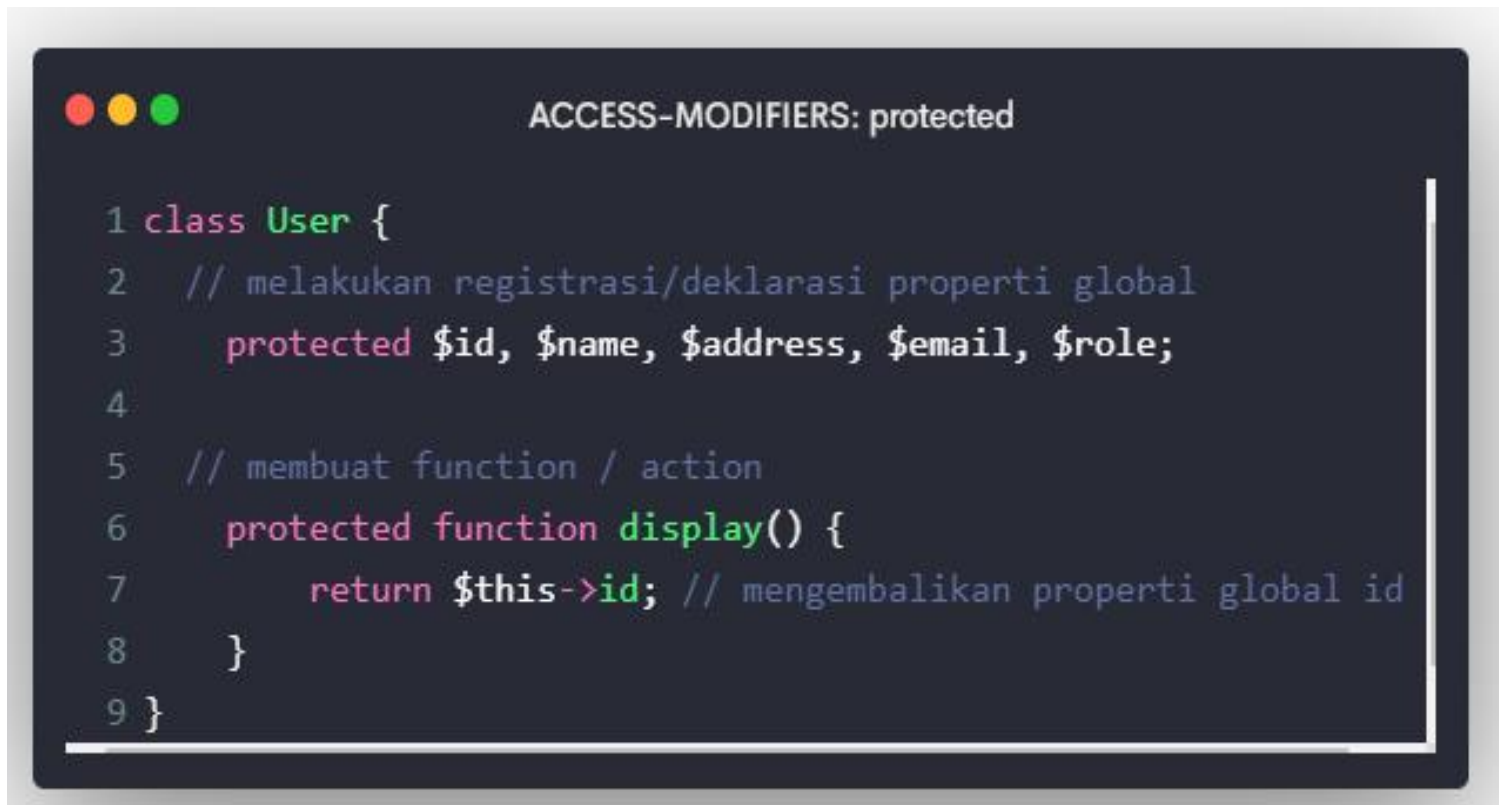
A screenshot of a code editor window titled "ACCESS-MODIFIERS: protected". The editor has a dark background with syntax-highlighted PHP code. The code defines a class named 'User' with two private properties (\$id, \$name, \$address, \$email, \$role) and a private function named 'display()' that returns the value of \$this->id. The code is numbered from 1 to 9.

```
1 class User {  
2     // melakukan registrasi/deklarasi properti global  
3     private $id, $name, $address, $email, $role;  
4  
5     // membuat function / action  
6     private function display() {  
7         return $this->id; // mengembalikan properti global id  
8     }  
9 }
```

Gambar 3.1 Penggunaan access-modifiers: private

- Protected

Jenis aksesibilitas protected pada *Property* atau pun *function* yang dimiliki pada class dapat diakses didalam class itu sendiri atau digunakan pada *class* yang melakukan pewarisan dari BaseClass / ParentClass, namun tidak bisa untuk kesemua class

A screenshot of a code editor window with a dark background. The title bar at the top says "ACCESS-MODIFIERS: protected". The code is written in PHP and is as follows:

```
1 class User {  
2     // melakukan registrasi/deklarasi properti global  
3     protected $id, $name, $address, $email, $role;  
4  
5     // membuat function / action  
6     protected function display() {  
7         return $this->id; // mengembalikan properti global id  
8     }  
9 }
```

Gambar 3.2 Penggunaan access-modifiers: protected

Property atau pun function yang dimiliki dapat diakses maupun digunakan di class yang diwariskan maupun declass dimana saja, tergantung dengan kebutuhan pemakaian.

- Public

Jenis Access-modifiers atau aksesibilitas ketika menerapkan tipe akses pada property global atau pun function / action dapat digunakan ketika menggunakan property ataupun function didalam class itu sendiri, declass yang dilakukan pewarisan, ataupun declass lain tanpa melakukan pewarisan. Berikut contoh penggunaannya:

A screenshot of a code editor window with a dark background. The title bar at the top reads "ACCESS-MODIFIERS: public". The code is written in PHP and is as follows:

```
1 class User {
2     // melakukan registrasi/deklarasi properti global
3     public $id, $name, $address, $email, $role;
4
5     // membuat function / action
6     public function display() {
7         return $this->id; // mengembalikan properti global id
8     }
9 }
```

Gambar 3.3 Penggunaan access-modifiers: public

- Default

Jenis access-modifiers ini merupakan tidak menggunakan ketiga jenis aksesibilitas yang diatas(private, protected, public) dengan kata lain, ketika membuat sebuah *property / variable* dalam area local scope atau didalam *function* yang hanya dapat digunakan didalam function itu saja, tidak dapat diakses kedalam functon lain meskipun dalam 1 class yang sama ataupun juga class yang melakukan pewarisan dan juga class lainnya. Berikut contoh penggunaan access-modifiers: default



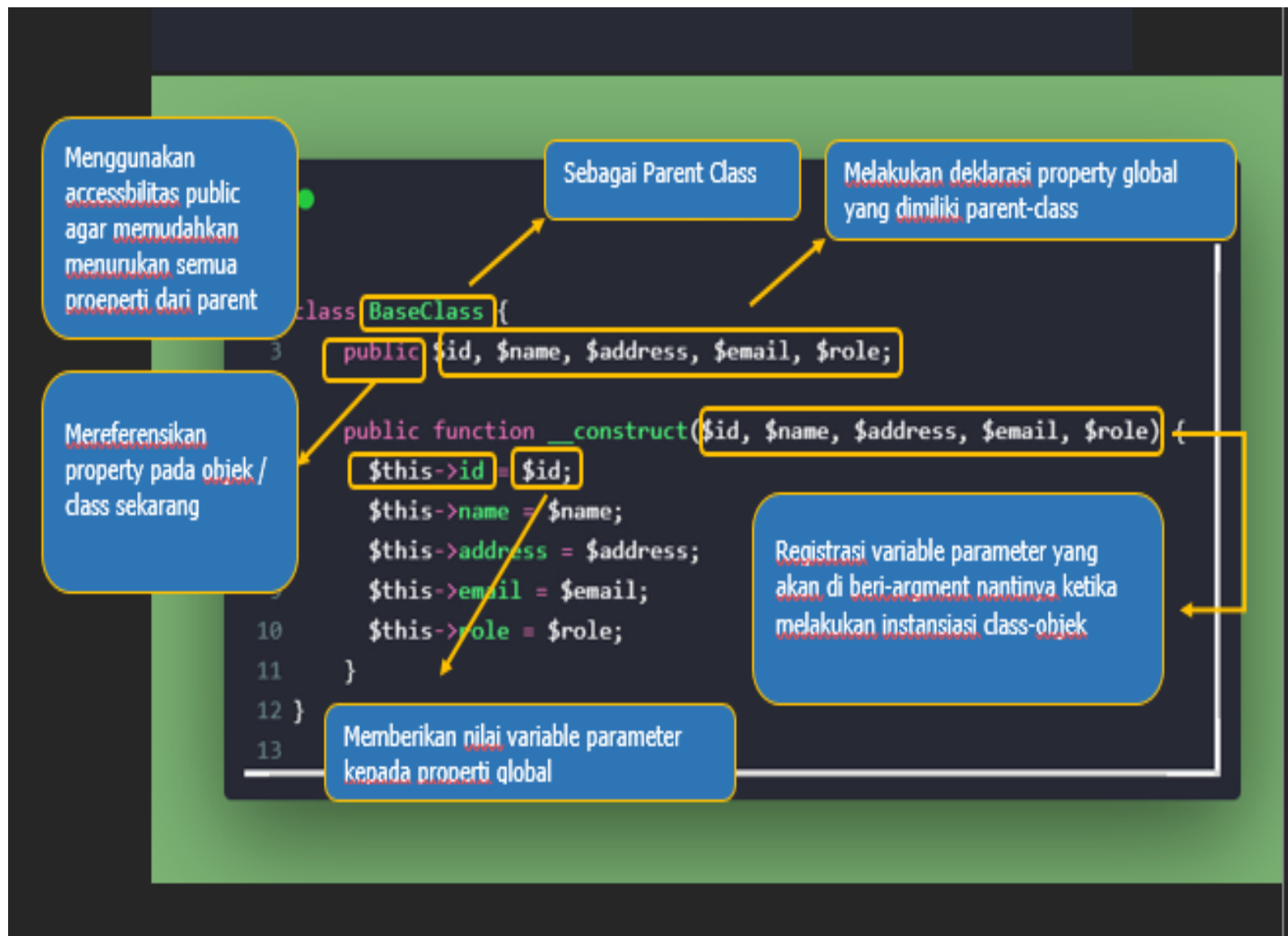
```
1 class User {
2     // melakukan registrasi/deklarasi properti global
3     public $id, $name, $address, $email, $role;
4
5     // membuat function / action
6     public function display($idUser) { // idUser sebagai parameter
7         // area local scope
8         $id = $idUser;
9         echo $id; // menampilkan properti local id
10    }
11 }
```

Gambar 3.4 Penggunaan access-modifier: default

Inheritance / Pewarisan

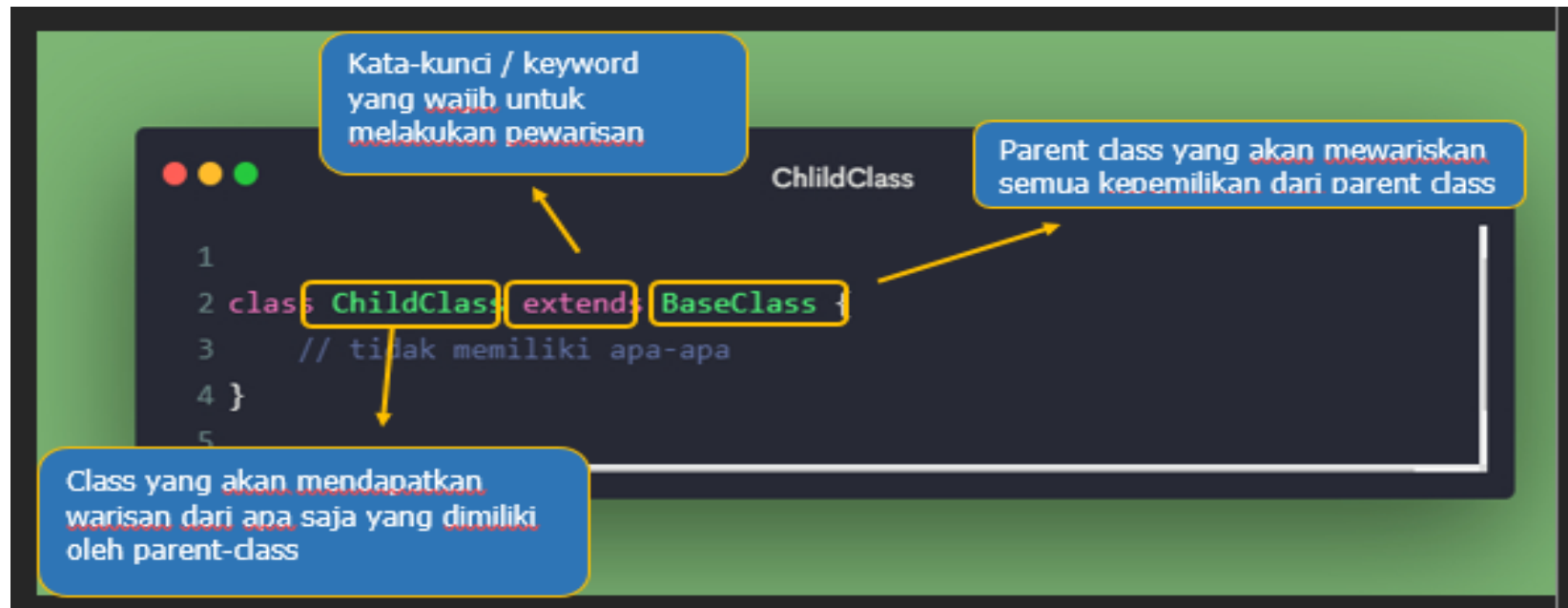
Konsep Inheritance atau Pewarsan merupakan konsep yang dapat menurunkan semua yang dimiliki seperti property global, function dan yang lainnya oleh parent-class yang dapat diakses oleh child-class dengan beberapa syarat dan key yang wajib digunakan. Berikut contoh codenya serta penulis menyematkan link full base-code:

<https://github.com/JunjungHasudungan/materi-pemrograman-berorientasi-objek/tree/main/inheritance>



Gambar 4.1 Merupakan Parent Class

Pada Child class akan menerima warisan dari parent class setelah melakukan pewarisan. Berikut contoh penggunaannya



Gambar 4.2 Penggunaan pada Child-class

Pada bagian selanjutnya index.php file yang akan menampilkan hasil dari penggunaan pewarisan dari parent-class ke child-class. Berikut contoh codenya:

The image shows a PHP script with several annotations explaining its components:

- Parent class yang akan mewariskan semua kenemilikan dari parent class**: Points to the `require_once 'BaseClass.php';` line.
- Merefereasikan property pada objek / class sekarang**: Points to the `$users` array.
- Melakukan instansiasi class yang digunakan untuk mengisi argument**: Points to the `new ChildClass` constructor calls.
- Argument merupakan nilai yang diisi berdasarkan dengan nama variable parameter dari parent-class**: Points to the arguments passed to the `new ChildClass` constructor.
- Nama objek yang digunakan untuk mengeluarkan nilai berdasarkan nama property yang dimiliki**: Points to the `$user->name` property access.
- Menampilkan data user berdasarkan nama properti dengan memulai nama objek dilanjutkan dengan nama properti**: Points to the `<td><?php echo $user->name; ?></td>` line.
- Nama array yang akan dilakukan perulangan / mengeluarkan data users**: Points to the `$users` array in the `foreach` loop.

```
1 require_once 'ChildClass.php';
2 require_once 'BaseClass.php';
3
4 // Contoh membuat beberapa objek ChildClass
5 $users = [
6     new ChildClass(1, "John Doe", "john.doe@example.com", "123 Main St", "Admin"),
7     new ChildClass(2, "Jane Smith", "jane.smith@example.com", "456 Elm St", "User"),
8     new ChildClass(3, "Alice Johnson", "alice.johnson@example.com", "789 Maple St",
9         "Moderator"),
10 ];
11
12 <table>
13     <thead>
14         <tr>
15             <th>#</th>
16             <th>Username</th>
17             <th>Email</th>
18             <th>Alamat</th>
19             <th>Jabatan</th>
20         </tr>
21     </thead>
22     <tbody>
23         <?php foreach ($users as $key => $user): ?>
24             <tr>
25                 <td><?php echo $key + 1; ?></td>
26                 <td><?php echo $user->name; ?></td>
27                 <td><?php echo $user->email; ?></td>
28                 <td><?php echo $user->address; ?></td>
29                 <td><?php echo $user->role; ?></td>
30             </tr>
31         <?php endforeach; ?>
32     </tbody>
33 </table>
```

Gambar 4.3 Menampilkan data ke halaman website

#Overriding Inheritance method

Dalam kondisi tertentu ada beberapa konsep yang dapat diterapkan ketika ChildClass memiliki properti yang berbeda dengan properti yang dimiliki BaseClass, maka dari itu method `__construct` dapat dioverride dalam `__construct` yang dimiliki ChildClass. Berikut contoh codenya

```
ChildClass.php
1 require 'BaseClass.php';
2
3 class ChildClass extends BaseClass {
4     public $phoneNumber;
5
6     public function __construct($id, $name, $address, $email, $role, $phoneNumber) {
7         parent::__construct($id, $name, $address, $email, $role);
8         $this->phoneNumber = $phoneNumber;
9     }
10 }
```

Melakukan registrasi / memanggil nama file class untuk digunakan sebagai parent class

Keyword untuk melakukan pewarisan dari parentClass

Menggunakan BaseClass sebagai parentClass

Registrasi property global yang dimiliki oleh ChildClass

Registrasi nama variable parameter yang dimiliki childclass

Memanggil property childclass untuk diisi nilai dari variable parameter

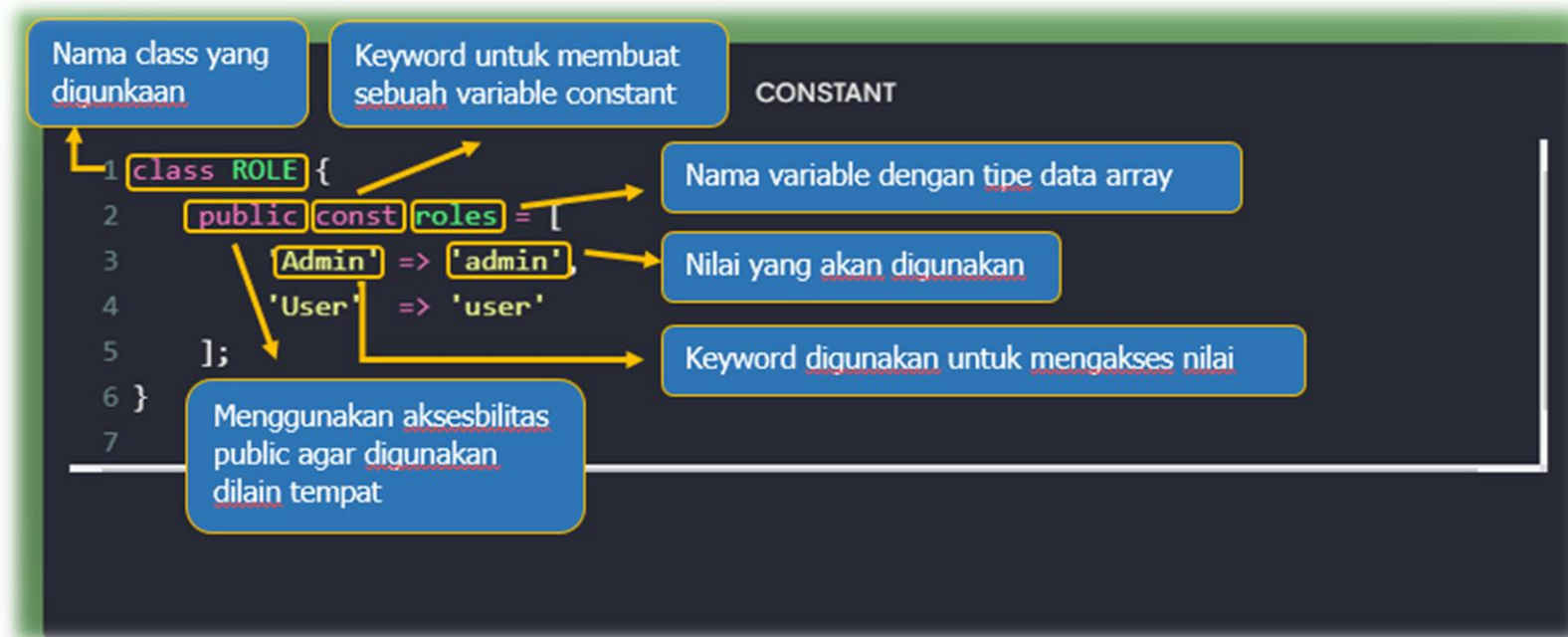
Menggunakan variable parameter untuk mengisi nilai keproperty global milik ChildClass

Gambar 4.4 Overriding inheritance method

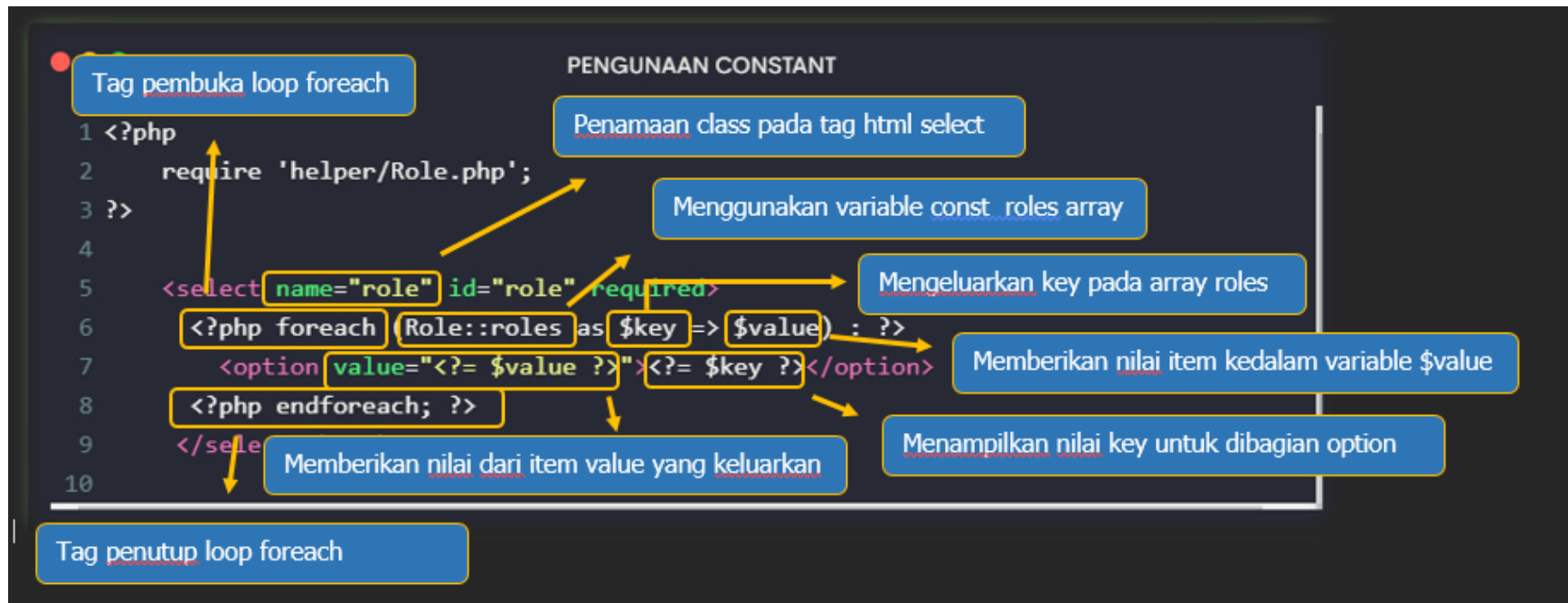
#CONSTANT

Constant merupakan konsep pemberian nilai tetap pada sebuah variable yang akan digunakan sesuai dengan kebutuhan. Berikut contoh base-code dan penjelasannya serta link full base-code:

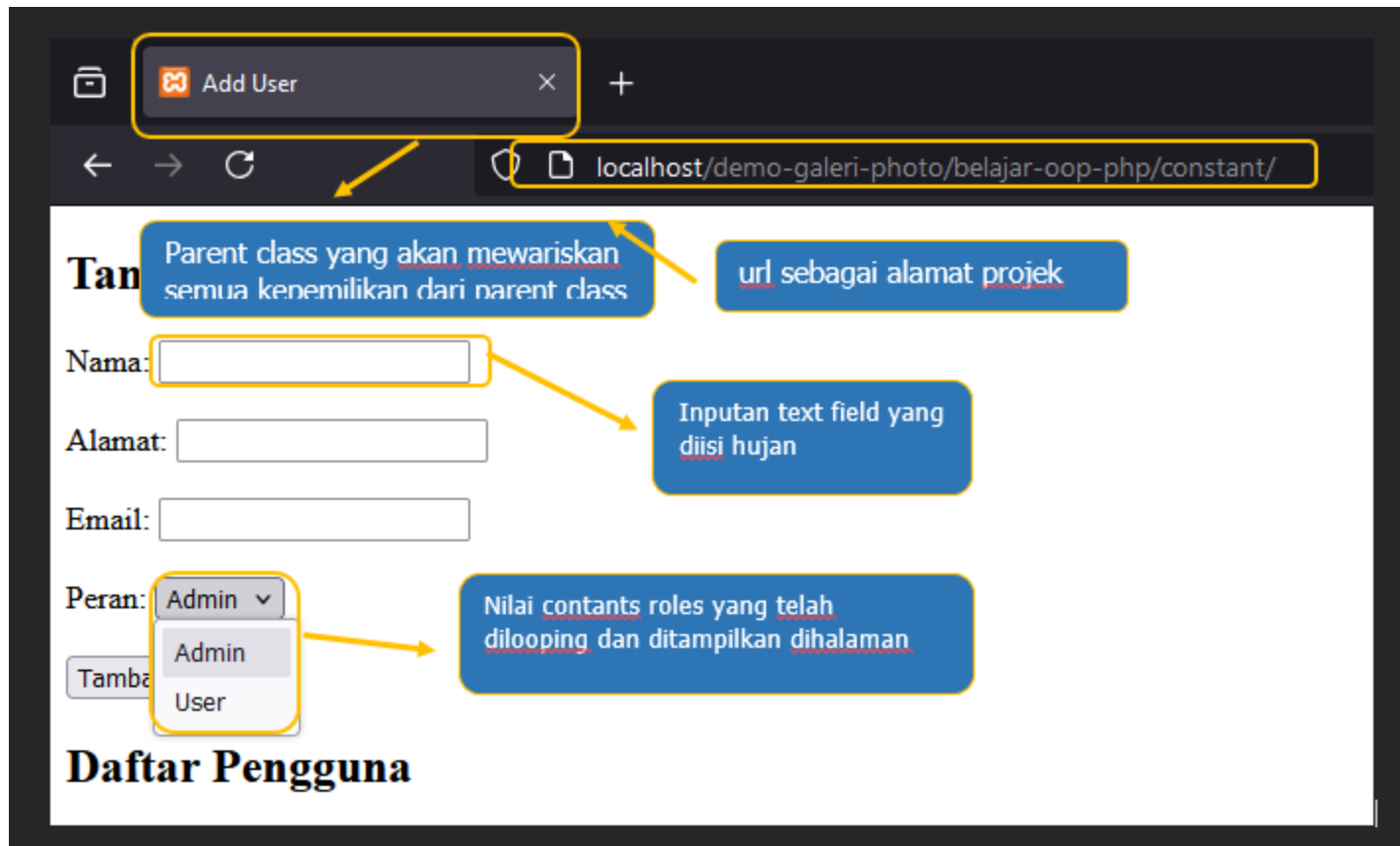
<https://github.com/JunjungHasudungan/materi-pemograman-berorientasi-objek/tree/main/constant>



Gambar 5.1 Penjelasan constant



Gambar 5.2 Base-code penggunaan Constant kedalam halaman website



Gambar 5.3 Tampilan pada form tambah pengguna

Pada gambar diatas merupakan tampilan halaman dari base-code yang digunakan serta penggunaan select-option yang dilakukan perulangan agar menampilkan value / nilai dari constant yang dimiliki. Adapun URL(*Uniform RESOURCE LOCATION*) di atas merupakan alamat proyek yang dimiliki

#ABSTRACT CLASS

Abstract class adalah sebuah class yang tidak dapat dilakukan instansiasi(tidak dapat dibuat menjadi objek) serta berperan menjadi kerangka dasar bagi class turunannya. Dengan kata lain abstract class merupakan prototype untuk membuat class yang dibutuhkan dengan

syarat harus melakukan pewarisan / inheritance. <https://www.duniaikom.com/tutorial-belajar-oop-php-pengertian-abstract-class-dan-abstract-method-php/> . Berikut syarat dan kemampuan dari abstract class. Alamat link:

<https://github.com/JunjungHasudungan/materi-pemograman-berorientasi-objek/tree/main/abstract-class>

- Abstract-class tidak dapat melakukan instansiasi class-objek
- Abstract-class dapat menurunkan / mewariskan / inheritance property, serta method kepada child class
- Abstract-class dapat melakukan registrasi action / function ataupun registrasi property global tanpa body code

contoh code dan penjelasannya

The image shows a code editor with a PHP file named `Models/BaseClass.php`. The code defines an abstract class `BaseClass` with the following structure:

```
1 abstract class BaseClass
2 {
3     // registrasi properti global
4     public $id, $name, $email, $address;
5
6     public function __construct($id = "", $name = "", $email = "", $address = "")
7     {
8         $this->id = $id;
9         $this->name = $name;
10        $this->email = $email;
11        $this->address = $address;
12    }
13
14    // action / function abstrak yang harus diimplementasikan oleh kelas turunan
15    abstract public function register();
16 }
17
```

Annotations and explanations are provided for various parts of the code:

- Nama class yang digunakan**: Points to the `BaseClass` name in the class declaration.
- Nama abstract class**: Points to the `abstract` keyword.
- Dapat melakukan registrasi property global**: Points to the public properties `$id, $name, $email, $address`.
- Registrasi variable parameter dengan nilai kosong**: Points to the default empty string values in the `__construct` method signature.
- Memberikan nilai melalui variable parameter kedalam property global**: Points to the assignments `$this->id = $id;` through `$this->address = $address;` inside the constructor.
- Dapat melakukan registrasi function dengan menyisipkan abstract didepan tanpa body code.**: Points to the `abstract public function register();` declaration.

Gambar 6.1 Contoh abstract-class yang menjadi parent-class

Pada gambar diatas, Baseclass dapat mewariskan / menurunkan / inheritance kepada child class dengan syarat setiap action ataupun property yang dimiliki oleh parent-class wajib digunakan. Berikut contoh penggunaan pewarisan pada class User dari BaseClass. Dapat juga diakses untuk melihat full base-code di link ini

<https://github.com/JunjungHasudungan/materi-pemograman-berorientasi-objek/tree/main/abstract-class>

```
1 require_once 'BaseClass.php';
2 require_once 'config/Database.php';
3
4 class User extends BaseClass {
5     public $role;
6     protected $db;
7
8     // menggunakan / override __construct dari parent
9     public function __construct($id = "", $name = "", $email = "", $address = "", $role = "") {
10         parent::__construct($id, $name, $email, $address);
11
12         $this->role = $role;
13         $this->db = Database::getConnection();
14     }
15
16     // melakukan override function dari parent-class
17     public function register()
18     {
19         // Query untuk menyimpan data user
20         $query = "INSERT INTO users (id, name, email, address, role) VALUES (:id, :name, :email, :address, :role)";
21         $stmt = $this->db->prepare($query);
22         $stmt->bindParam(':id', $this->id);
23         $stmt->bindParam(':name', $this->name);
24         $stmt->bindParam(':email', $this->email);
25         $stmt->bindParam(':address', $this->address);
26         $stmt->bindParam(':role', $this->role);
27         $stmt->execute();
28     }
29
30     // function kepunyaan child-class
31     public static function getAllUsers()
```

Melakukan registrasi BaseClass sebagai parent-class yang akan mewariskan

Melakukan registrasi Database.php yang akan mengkoneksikan kedatabase

BaseClass Mewariskan semua yang dimiliki ke class User

Registrasi variable parameter dengan nilai kosong

Melakukan overriding method construct dari parent class

Memberikan sebuah function getConnection dari class Database ke property global

Function dari parent class wajib digunakan pada child-class, melakukan overriding / menggunakan kembali function dari parent-class

Function yang dimiliki user untuk mengambil seluruh data users

Registrasi property yang dibutuhkan oleh class User

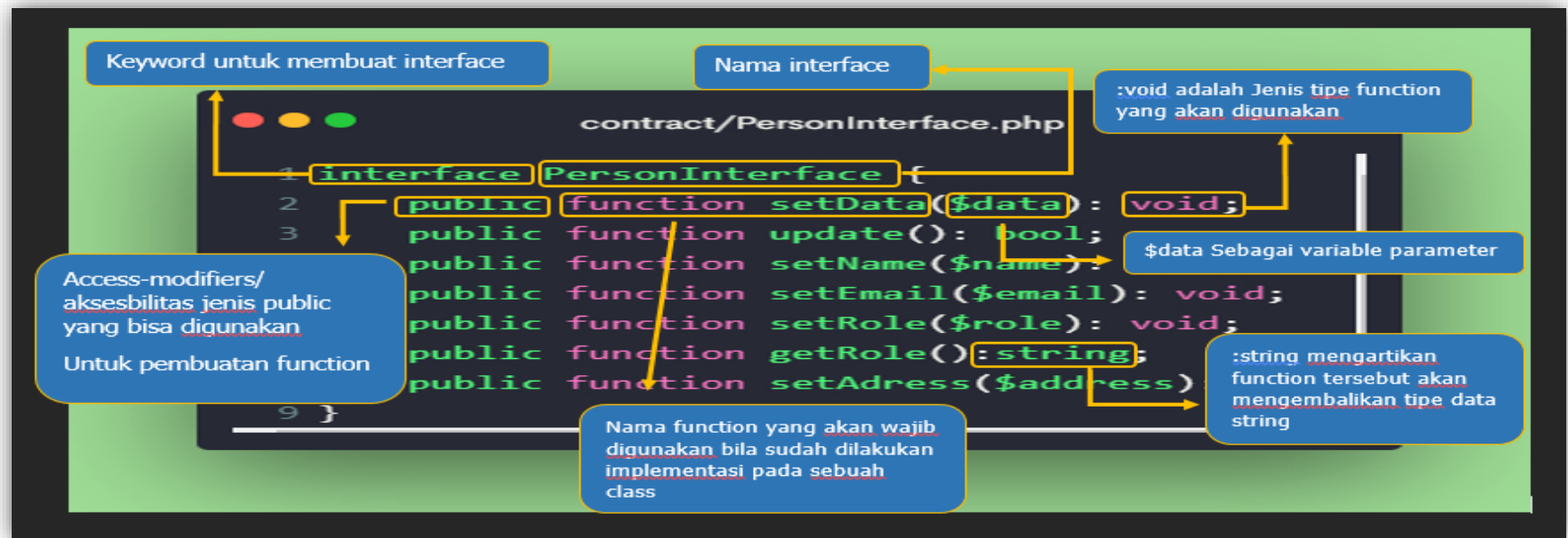
Gambar 6.2 Melakukan pewarisan / inheritance dari BaseClass

#INTERFACE

Dalam pengertian sederhana interface adalah antar-muka dimana interface berperan sebagai membuat function / registrasi / signature function yang akan digunakan sebagai implementasi pada beberapa class yang akan menggunakannya. <https://www.duniaikom.com/tutorial-belajar-oop-php-pengertian-object-interface-dalam-pemrograman-berbasis-objek/> ada pun syarat dan ketentuan dalam membuat dan menggunakan interface tersebut, diantaranya:

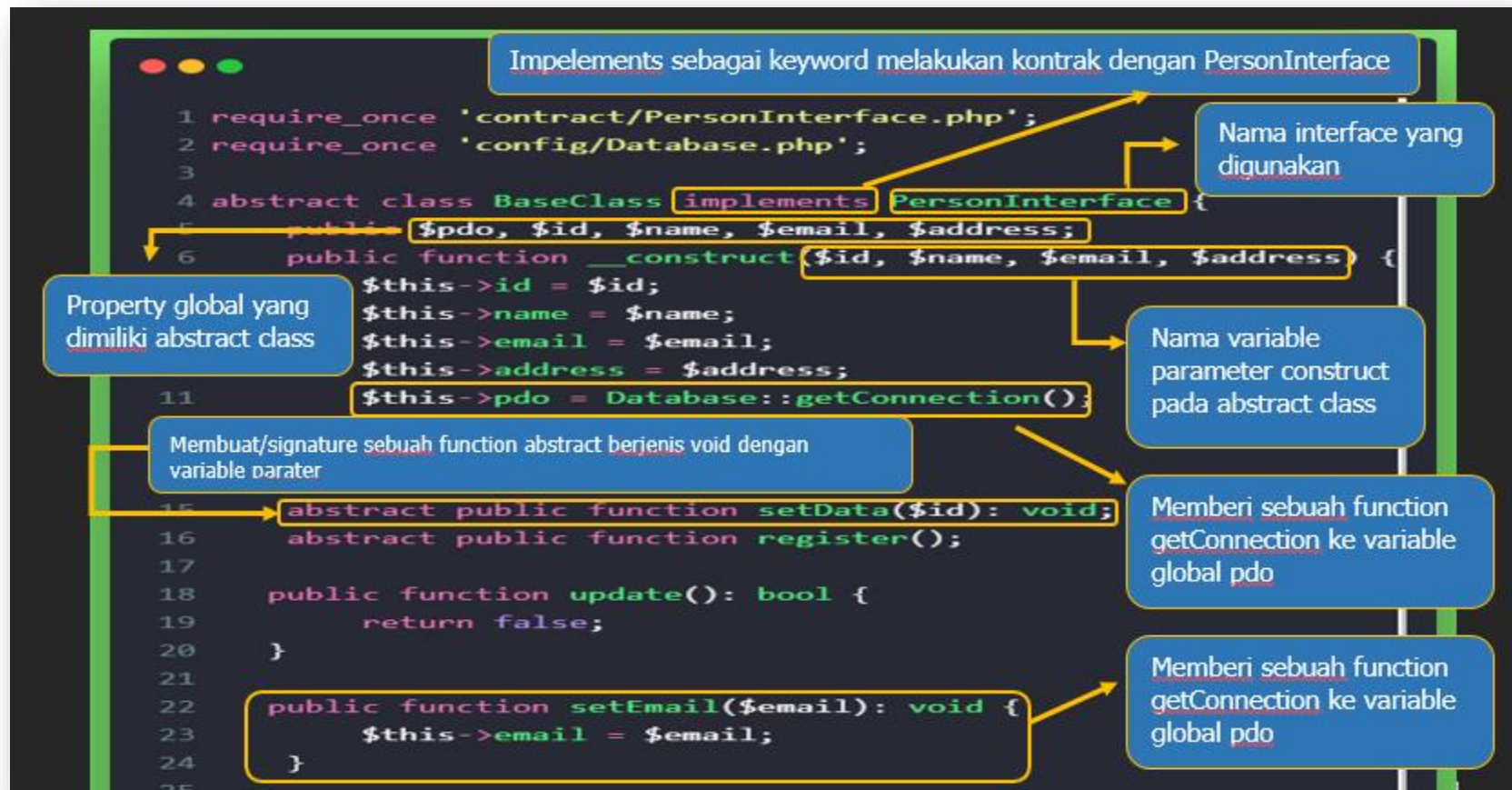
- Interface hanya bisa menggunakan access-modifiers / aksesibilitas jenis public
- Interface hanya bisa membuat / registrasi / signature sebuah function yang memiliki tipe function void, void parameter, return tipe data
- Interface tidak bisa melakukan registrasi / deklarasi property global
- Interface tidak bisa membuat sebuah constructor
- Interface merupakan sebuah kontrak bagi class yang menggunakannya, dengan kata lain pada class yang menggunakan interface wajib memberikan *keyword* / kata kunci implements sebagai arti kontrak digunakan atau mengimplementasi dari interface tersebut
- Pada class yang menggunakan interface, wajib menggunakan seluruh function yang diimplementasi dari interface yang dipakai

Berikut contoh base-code serta link full base-code: <https://github.com/JunjungHasudungan/materi-pemograman-berorientasi-objek/tree/main/interface>



Gambar 6.1 base-code PersonInterface

Pada base-code abstract class BaseClass melakukan implements atau kontrak yang telah dan wajib menggunakan semua function yang telah dibuat/registrasi/signature dari PersonInterface pada base-code sebelumnya.



Gambar 6.2 Melakukan implements dari PersonInterface


```

5 class User extends BaseClass {
6     public $role, $pdo;
7
8     public function __construct(
9         $id = "", $name = "", $email = "", $address = "", $role = ""
10     ) {
11         parent::__construct($id, $name, $email, $address);
12         $this->role = $role;
13         $this->pdo = Database::getConnection();
14     }
15
16     public function setData($id): void {
17         $stmt = $this->pdo->prepare("SELECT * FROM users WHERE id = :id");
18         $stmt->bindParam(':id', $id);
19         $stmt->execute();
20         $data = $stmt->fetch(PDO::FETCH_ASSOC);
21
22         if ($data) {
23             $this->id = $data['id'];
24             $this->name = $data['name'];
25             $this->email = $data['email'];
26             $this->address = $data['address'];
27             $this->role = $data['role'];
28         }
29
30     }
31
32     public function update(): bool {
33         $stmt = $this->pdo->prepare(
34             "UPDATE users SET name = :name, email = :email, address = :address, role = :role
35             WHERE id = :id");
36         $stmt->bindParam(':name', $this->name);
37         $stmt->bindParam(':email', $this->email);
38         $stmt->bindParam(':address', $this->address);
39         $stmt->bindParam(':role', $this->role);
40         $stmt->bindParam(':id', $this->id);
41         $stmt->execute();
42         return true;
43     }
44 }

```

Property global yang dimiliki oleh class User

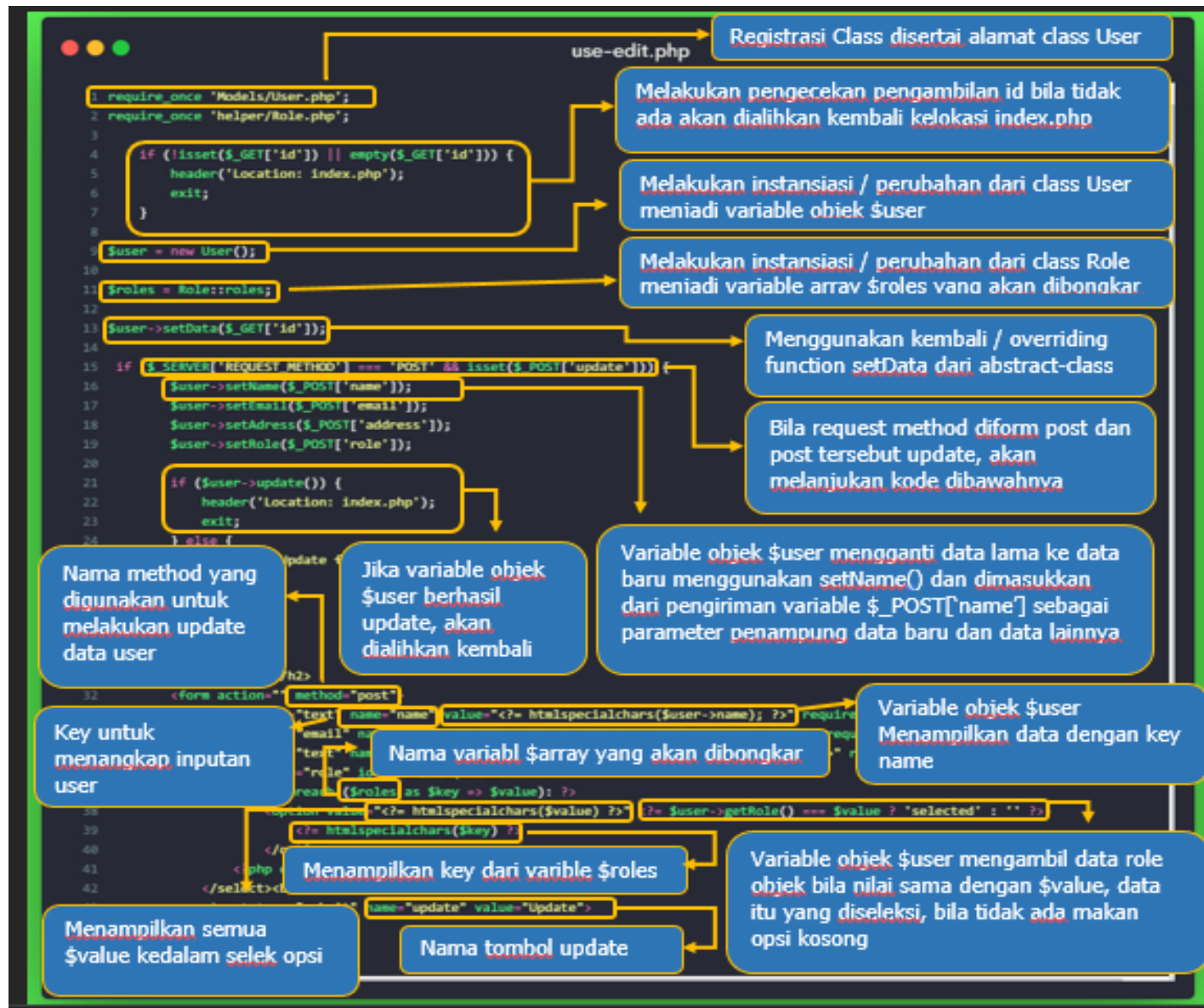
Menggunakan kembali / overriding method constructor dari parent-class

Membuat variable parameter dengan nilai default empty string

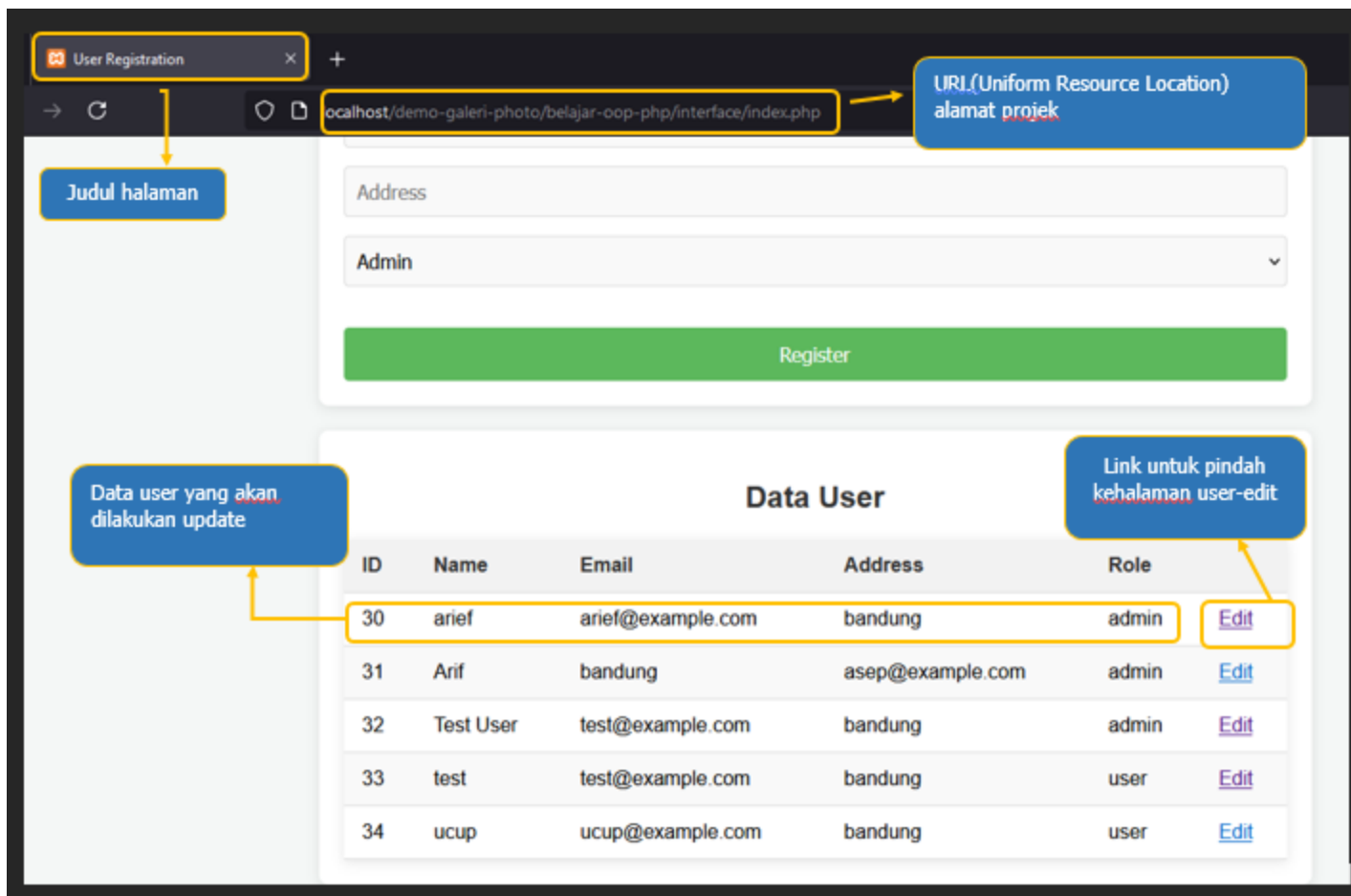
Menggunakan kembali / overriding function setData dari abstract-class

Menggunakan kembali / overriding function update yang dimiliki dari Interface yang diimplement ke Basecase dan diwariskan ke User

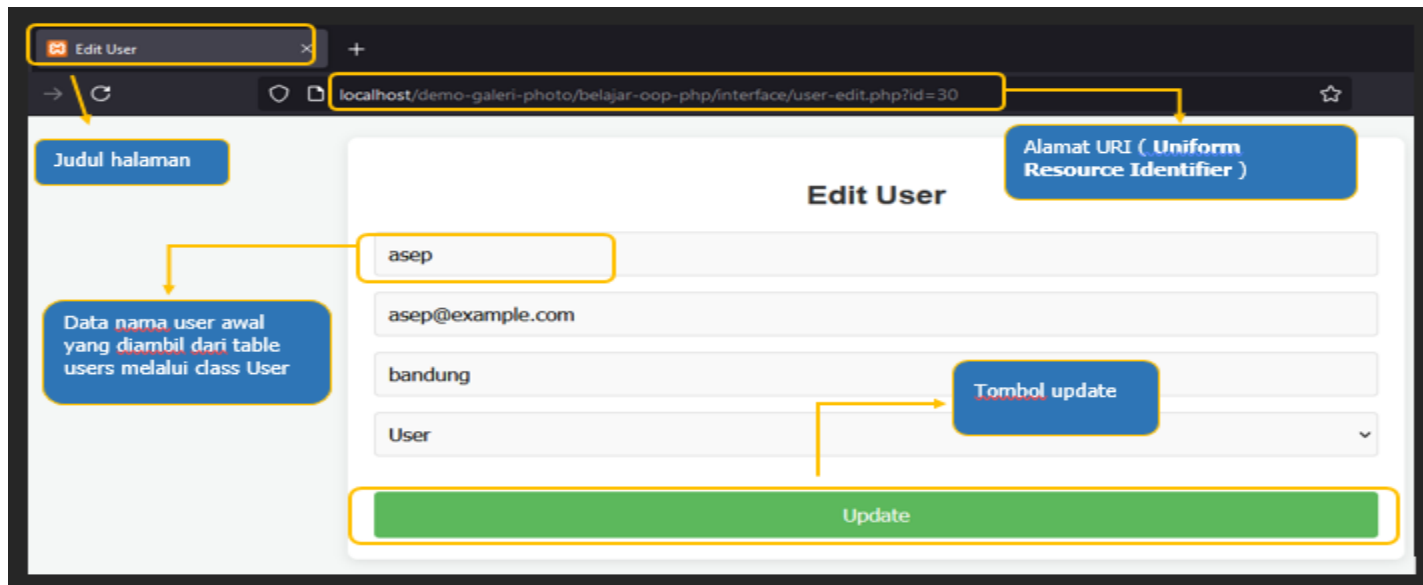
Gambar 6.3 Melakukan pewarisan / pewarisan dari abstract-class (BaseClass)



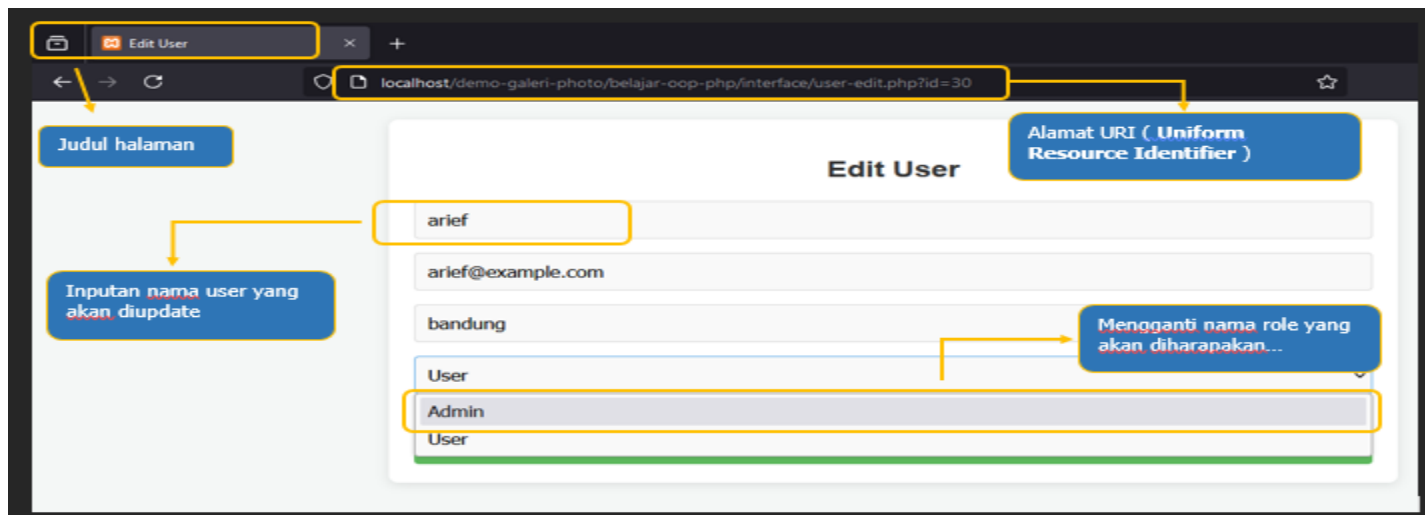
Gambar 6.4 base-code user-index.php



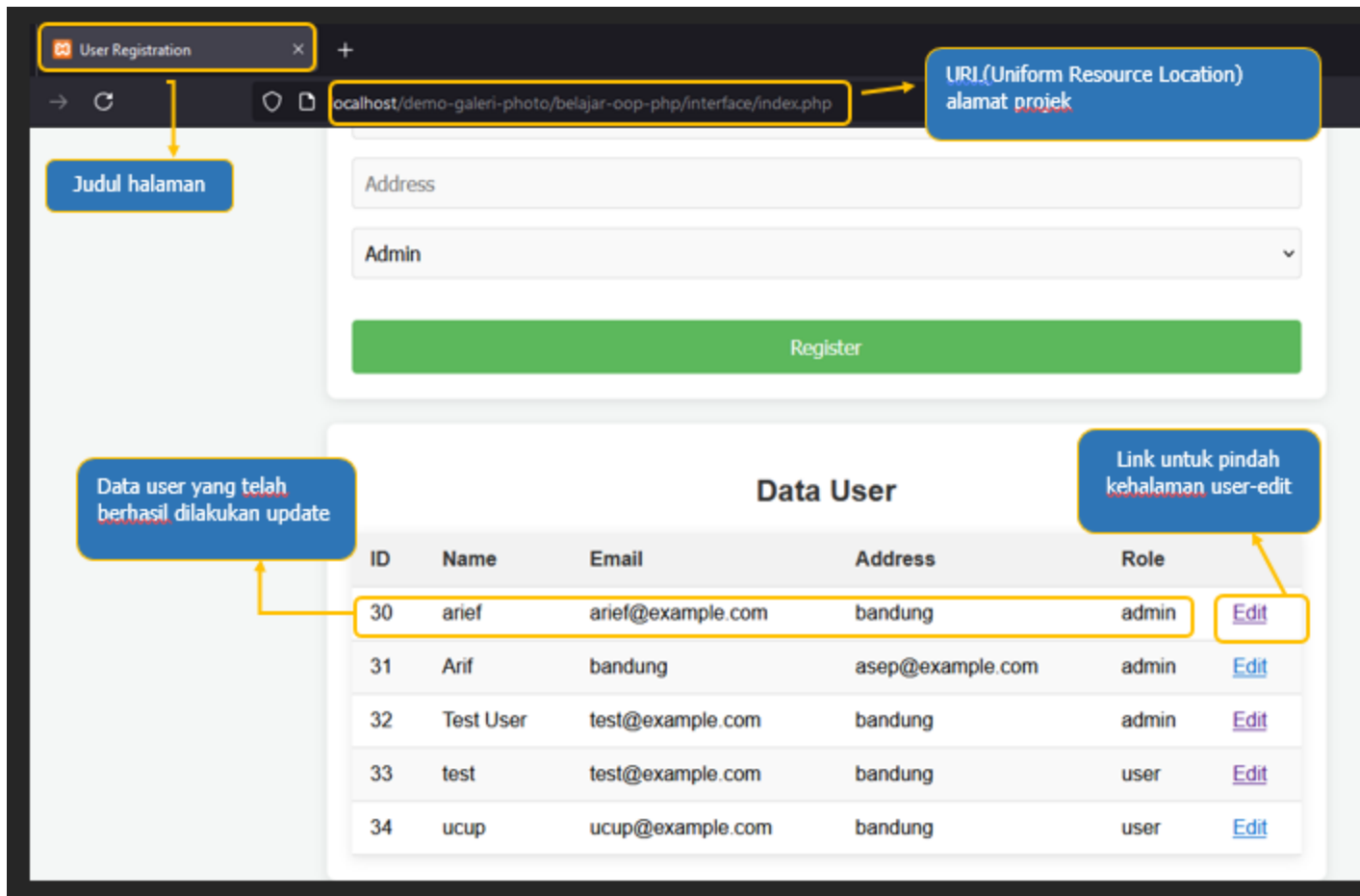
Gambar 6. 5 tampilan halaman utama pada index.php



Gambar 6.6 Tampilan Halaman user-edit menampilkan data awal user



Gambar 6. 7 Tampilan penginputan data user yang akan di update



Gambar 6. 7 Tampilan Halaman index yang telah dilakukan update data user berdasarkan id