|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **MATERI MODULE P.B.O-PEMOGRMAN WEBSITE MENGGUNAKAN BAHASA PHP** | | | | |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  | Kelas | : 12 RPL | | |
|  | Mapel | : P.B.O dan Pemograman Website | | |
|  | Disusun oleh | : Hasudungan Sitorus S, Kom | | |

|  |
| --- |
| **#PROLOG** |

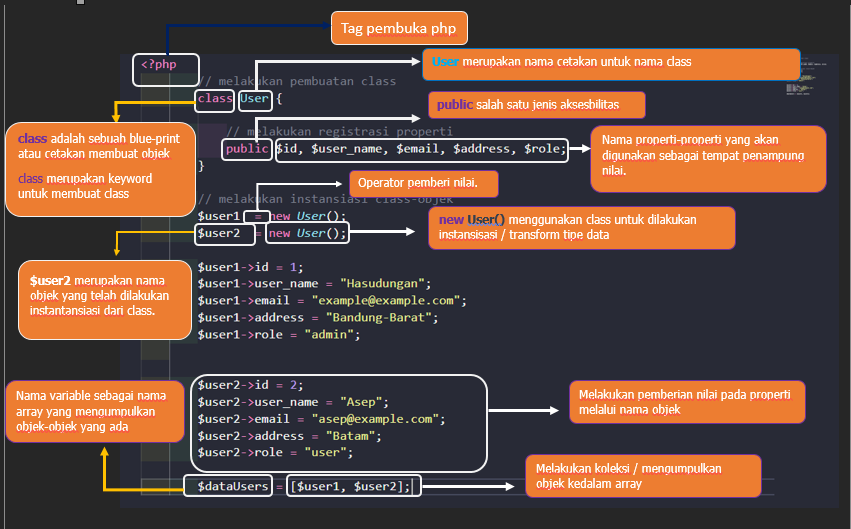
Pemrograman Berorientasi Objek atau Object Oriented Programming (OOP) adalah suatu strategi pembangunan perangkat lunak yang mengorganisasikan perangkat lunak sebagai kumpulan objek yang berisi data dan operasi yang diberlakukan terhadapnya. Pemahaman ini mengacu pada teknis dari pemrograman OOP, yakni 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 / memcahkan masalah berorientasi dasar class,objek-objek serta attribute pendukung lainnya yang digunakan.

|  |
| --- |
| **#CLASS-OBJEK** |

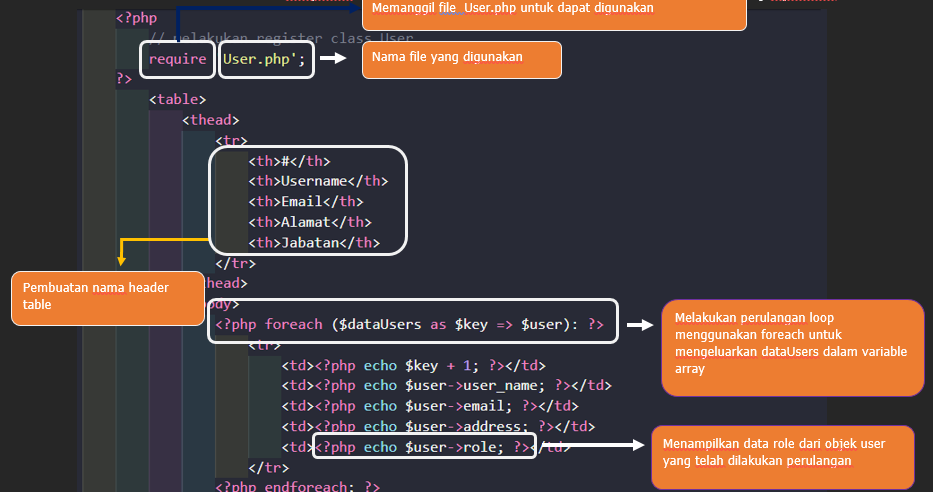
Class adalah template (pola) untuk berbagai obyek dengan fitur serupa. Class mewujudkan semua kumpulan/set fitur tertentu dari objek. Referensi: <https://staffnew.uny.ac.id/upload/132206816/pendidikan/bambangshm-pbo-matericlassobject.pdf#:~:text=Class%20adalah%20template%20%28pola%29%20untuk%20berbagai%20obyek%20dengan,Class%20mewujudkan%20semua%20kumpulan%2Fset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

Contoh base-code dan penggunaannya.

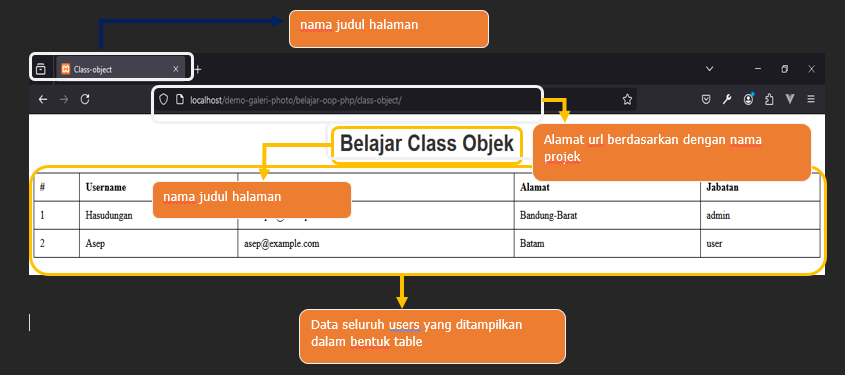


**Gambar 1.1** class User.php

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

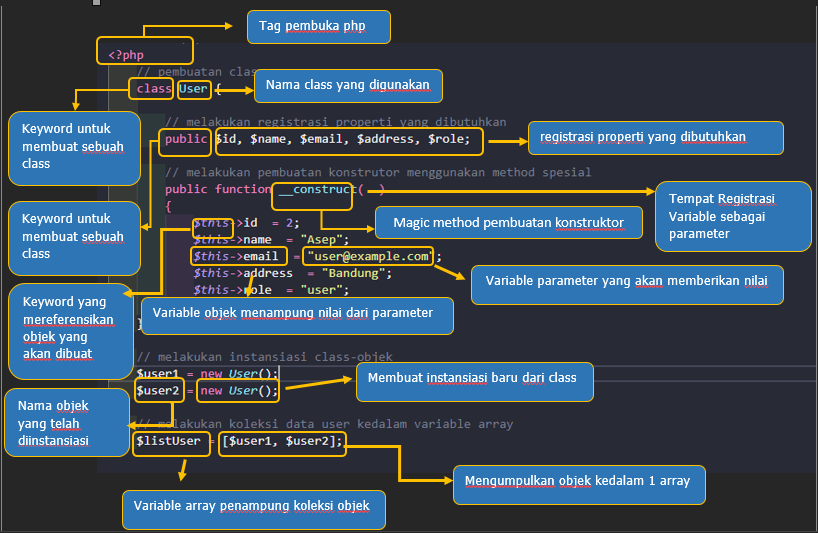


**Gambar1.3** Tampilan Halaman index.php

|  |
| --- |
| **#CONSTRUCTOR** |

Constructor adalah merupakan konsep pembuatan konstruksi yang memungkinkan melakukan inisailisasi 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** |

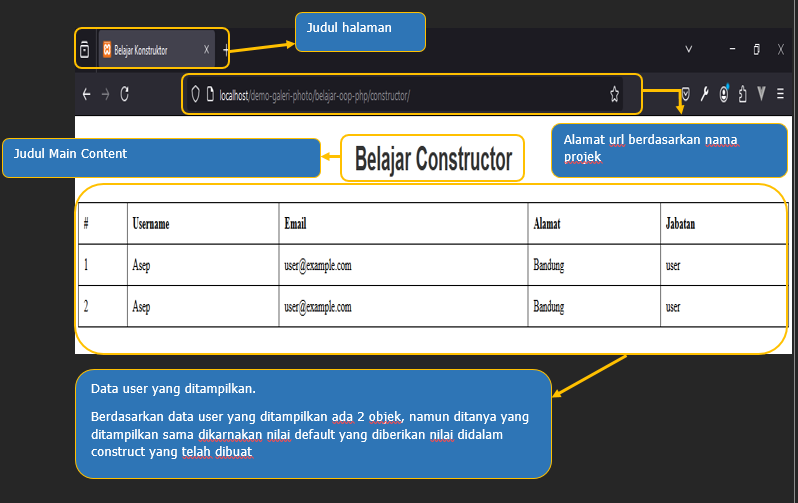


**Gambar 2.1** Constructor dengan parameter

**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 contruct 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

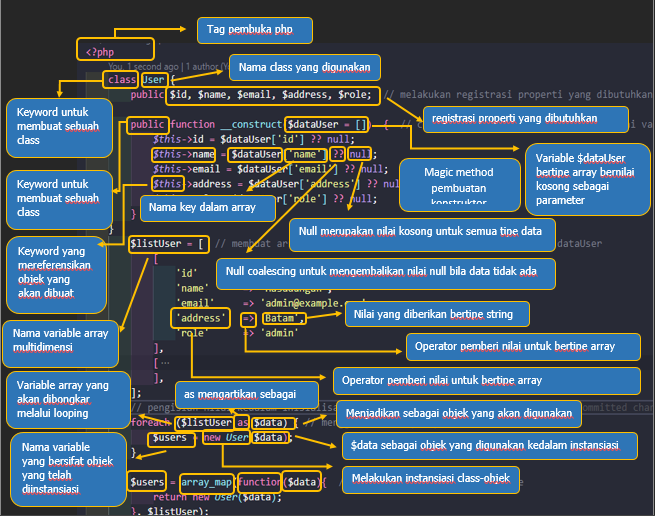


**Gambar 2.3** Tampillan Halaman table list data user

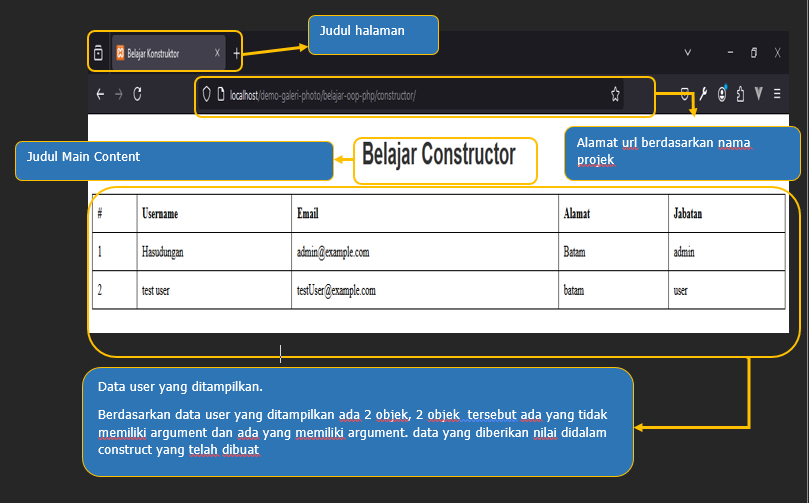
Berdasarkan data user yang ditampilkan, data user ada 2 atau objek yang telah diinstansiasi ada 2 namun data yang ditampilkan sama dikarnakan nilai yang dalam construct telah diberi nilai default.

|  |
| --- |
| **#Constructor dengan nilai default dalam parameter** |

Dalam kondisi tertentu, nilai default dalam parameter dengan tipe data aray 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



**Gambar 2.5** Tampilan Halaman list data user

|  |
| --- |
| **#Construtor dengan array kosong sebagai parameter** |



**Gambar 2.6** index.php

|  |
| --- |
| **#ACCESS MODIFIERS / AKSESBILIITAS** |

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

* Private

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



**Gambar 3.1** Penggunaan access-modifiers: private

* Protected

Jenis aksesbilitasi 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

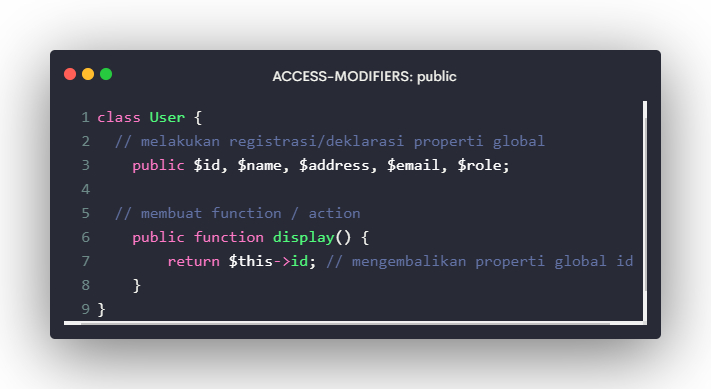


**Gambar 3.2** Penggunaan access-modifiers: protected

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 aksesbilitas ketika menerapakan 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:



**Gambar 3.3** Pengguaan access-modifiers: public

* Default

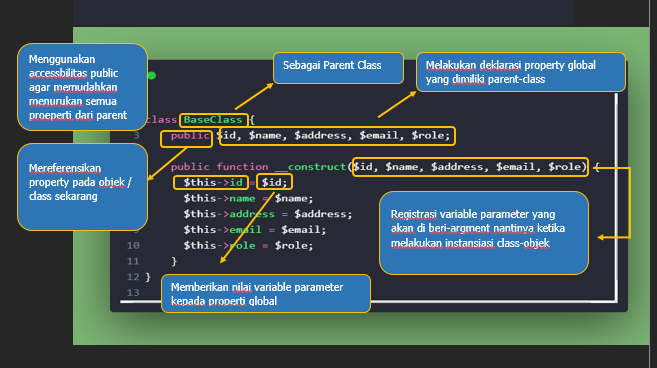
Jenis access-modifiers ini merupakan tidak menggunakan ketiga jenis aksesbilitas yang diatas(private, protected, public) dengan kata lain, ketika membuat sebuah *property* / *variable* dalam area local scope atau didalam *function* yang hanya dapat digunakan didalam function itu saja, tidak dapat diakses 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



**Gambar 3.4** Penggunaan access-modifier: default

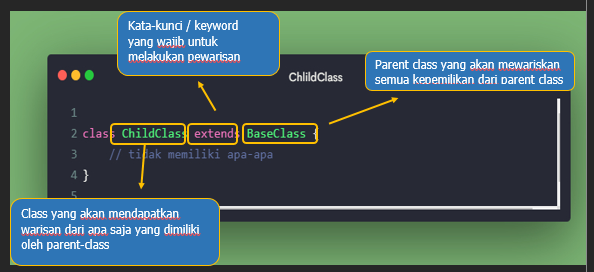
|  |
| --- |
| **# Inheritance / Pewarisan** |

Konsep Inheritance atau Pewarsan merupakan konsep yang dapat menurunkan semua yang dimiliki seperti property global, function dan yangk 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-pemograman-berorientasi-objek/tree/main/inheritance



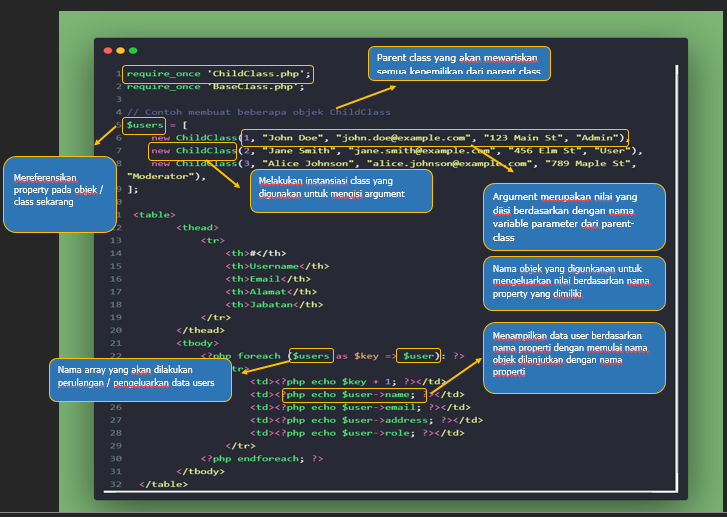
**Gambar 4.1** Merupakan Parent Class

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



**Gambar 4.2** Penggunaan pada Child-class

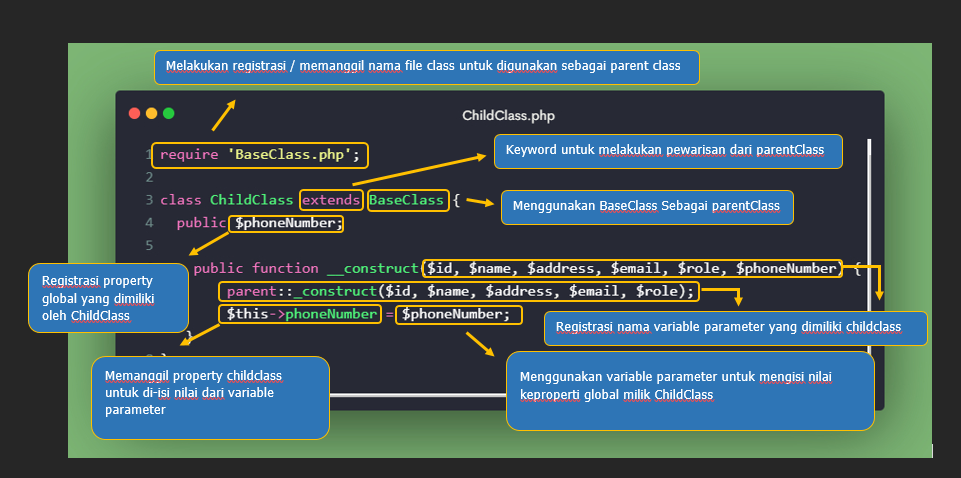
Pada bagian selanjutnya index.php file yang akan menampilkan hasil dari penggunaan pewarisan dari parent-class ke child-class. Berikut contoh codenya:



**Gambar 4.3** Menampilkan data ke halaman website

|  |
| --- |
| **#Overriding Inheritanced 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

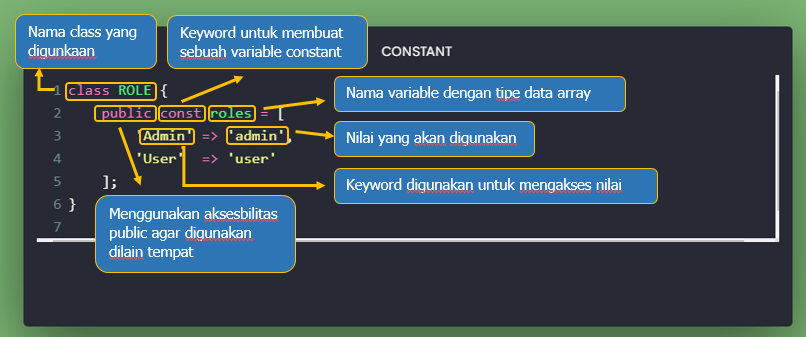


**Gambar 4.4** Overriding inheritanced method

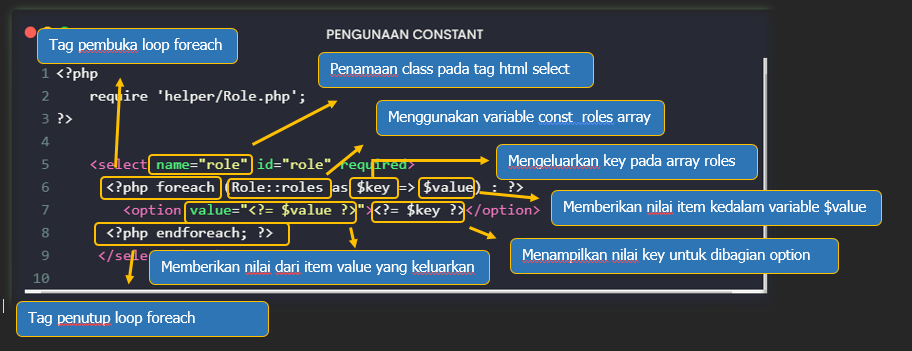
|  |
| --- |
| **#CONSTANT** |

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:

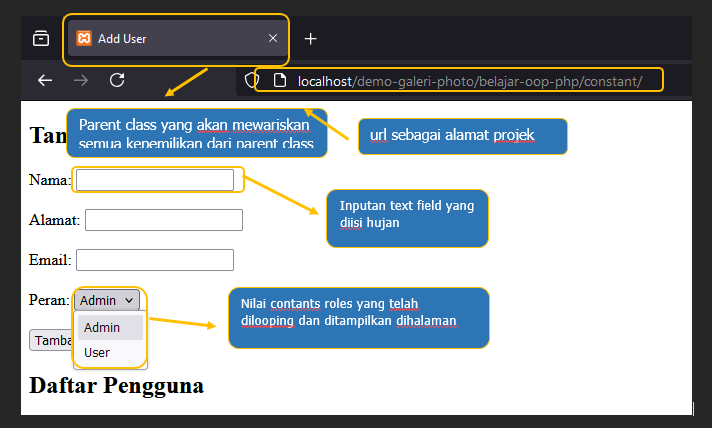
https://github.com/JunjungHasudungan/materi-pemograman-berorientasi-objek/tree/main/constant



**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 pengunaan select-option yang dilakukan perulangan agar menampilkan value / nilai dari contant yang dimiliki. Adapun URL(*Uniform RESOURCE LOCATION*) di atas merupakan alamat projek yang dimiliki

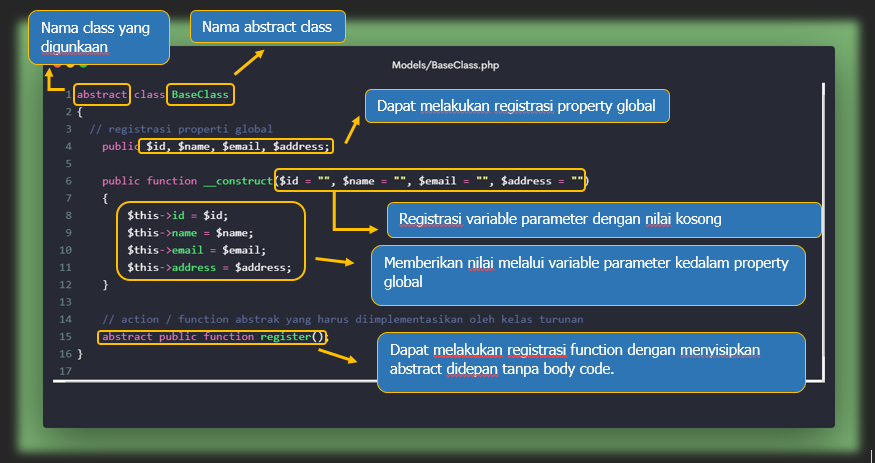
|  |
| --- |
| **#ABSTRACT CLASS** |

Abstract class adalah sebuah class yang tidak dapat dilakukan instansiasi(tidak dapat dapat dibuat menjadi objek) serta berperan menjadi kerangka dasar bagi class turunannnya. Dengan kata lain abstract class merupakan prototype untuk membuatan class yang dibutuhkan dengan syarat harus melakukan pewarisan / inheritance. <https://www.duniailkom.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

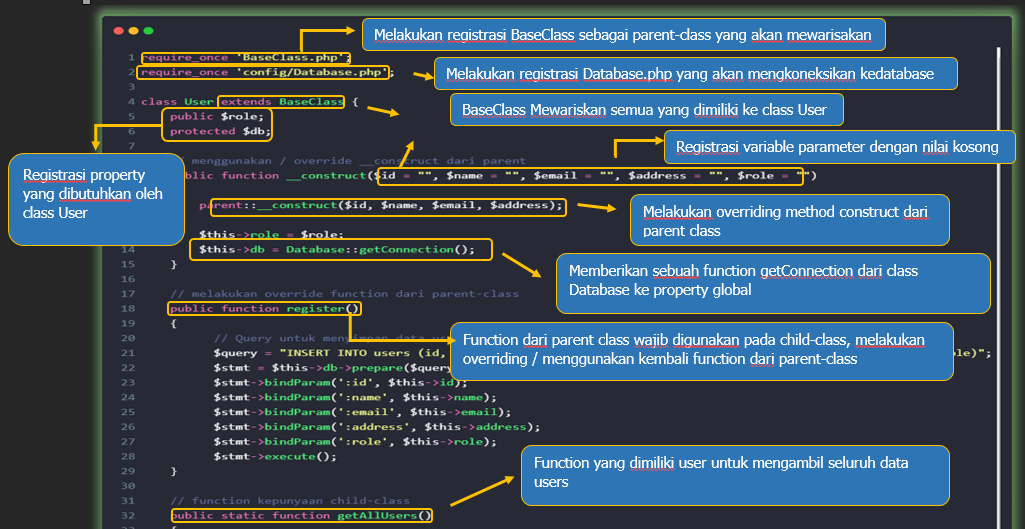
* Abstract-class tidak dapat melakukan instansiasi class-objek
* Abstract-class dapat menurukan / mewarisakan / inheritance property, serta method kepada child class
* Abstract-class dapat melakukan registrasi action / function ataupun registrasi property global tanpa body code

contoh code dan penjelasannya



**Gambar 6.1** Contoh abstract-class yang menjadi parent-class

Pada gambar diatas, Baseclass dapat mewariskan / menurunkan / inherintance 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

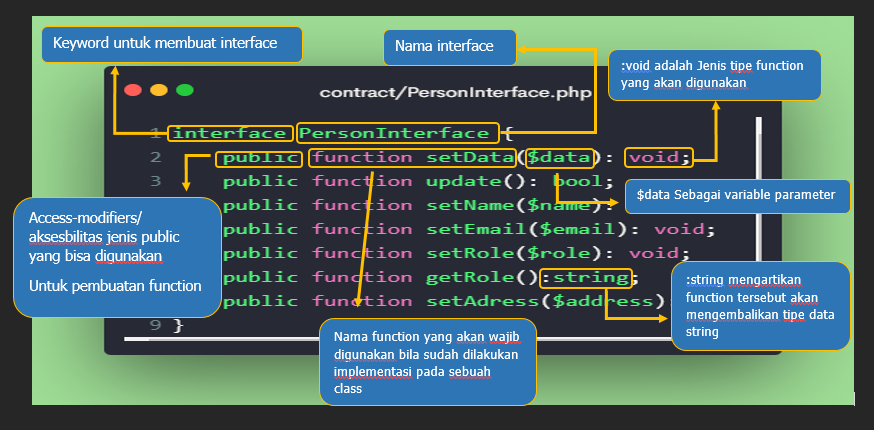


**Gambar 6.2** Melakukan pewarisan / inheritance dari BaseClass

|  |
| --- |
| **#INTERFACE** |

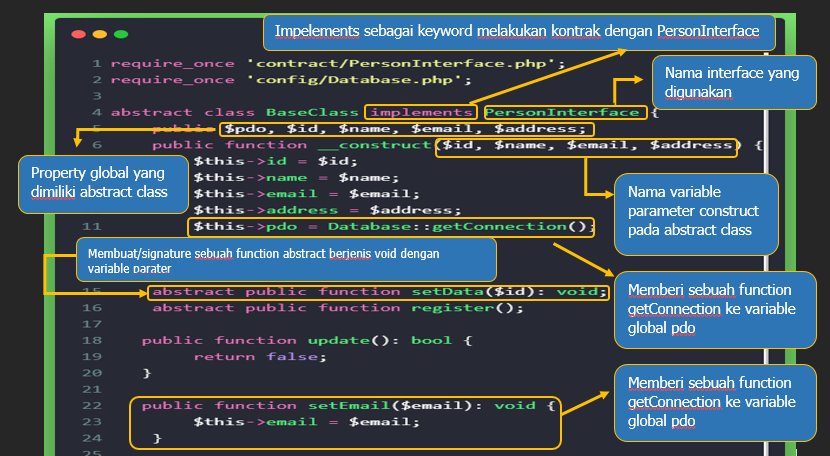
Dalam pengertian sederhana interface adalah antar-muka dimana interface berperan sebagai membuat function / registrasi / signature function yang akan digunakan sebagai implementasi pada beberapa class yang akan mengunakannya. <https://www.duniailkom.com/tutorial-belajar-oop-php-pengertian-object-interface-dalam-pemrograman-berbasis-objek/>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

* Interface hanya bisa menggunakan access-modifiers / aksesbilitas 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 k*eyword* / 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

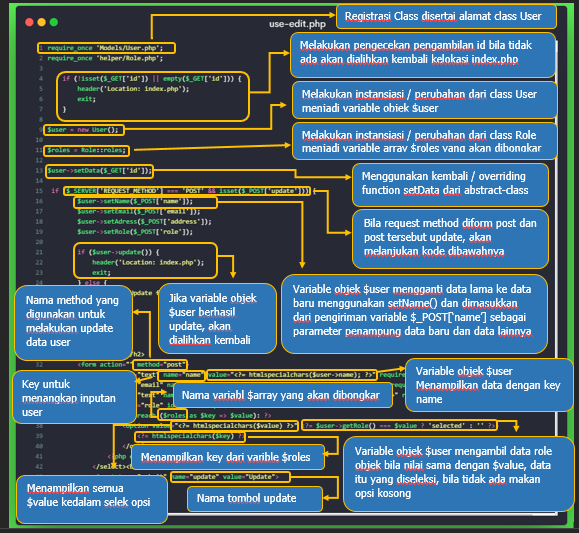
Pada base-code abstract class BaseClass melakukan implements atau kontrak yang telah dan wajib menggunakan semua function yang telah dibuat/registrasi/signature dari PersonInterface pada base-code sebelumnya.



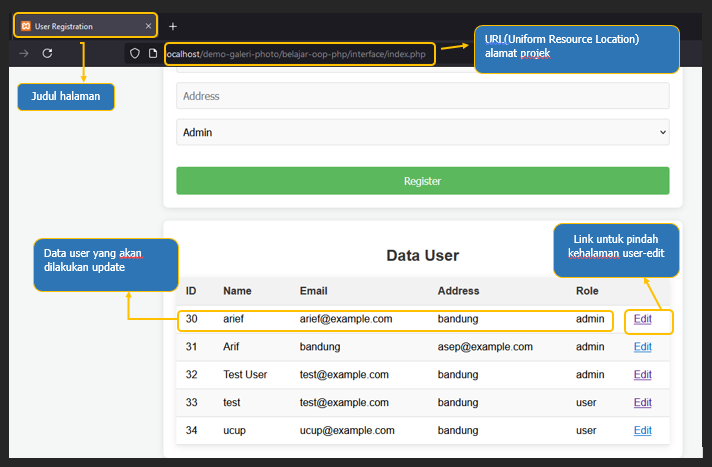
**Gambar 6.2** Melakukan implements dari PersonInterface



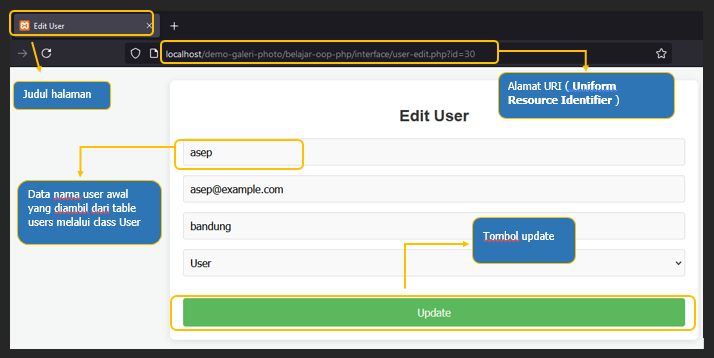
**Gambar 6.3** Melakukan pewarisan / pewarisan dari abstract-class (BaseClass)



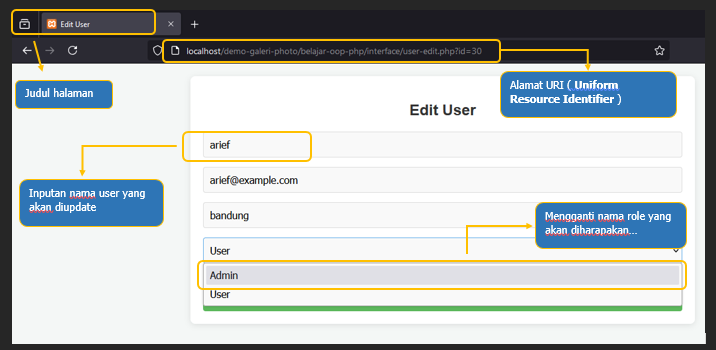
**Gambar 6**.4 base-code user-index.php



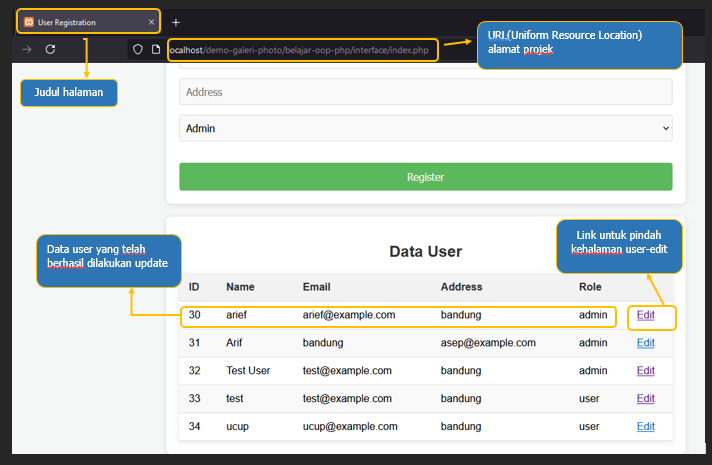
**Gambar 6. 5** tampilan halaman utama pada index.php



**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

|  |
| --- |
| **#TRAIT** |

Trait adalah Kumpulan-kumpulan function yang dapat digunakan tanpa melakukan *inheritance* / pewarisan ataupun melakukan kontrak / *implements*. <https://ngide.net/apa-itu-traits-dalam-php> . Dengan kata lain trait adalah Kumpulan sifat-sifat / function / method yang dapat digunakan di class lain tanpa harus melakukan pewarisan / *inheritance* maupun kontrak / *implements* dan juga trait itu sendiri merupakan module untuk mendukung kinerja ataupun performa untuk class yang menggunakan trait itu sendiri.

Syarat dan ketentuan membuatan dan penggunaan trait sebagai berikut:

1. Diawali dengan keyword **trait**
2. **Dapat memiliki property serta diwajibkan menggunakan jenis aksesbilitas public ataupun protected**
3. Trait tidak bisa melakukan extendsuntuk digunakan diclass lain
4. **Trait** tidak bisa melakukan kontrak / *implements* untuk class yang akan digunakan
5. **Bila ingin menggunakan trait, diwajibkakn registrasi alamat serta nama trait yang digunakan serta didalam body class di sertai dengan keyword use** untuk dapat menggunakan trait tersebut

Berikut contoh base-code dan penggunaan trait serta link yang disertakan