# BAB VI

# *CLASS* AND *CONSTRUCTOR*

## Tujuan

1. Dapat memahami pengertian Pemrograman Berbasis Objek.
2. Dapat memahami konsep dasar dari *Class* dan *Constructor*.
3. Dapat mengetahui dan memahami tujuan dari *Class* dan *Constructor*.
4. Mampu mengimplementasikan *Class* dan *Constructor* dalam bahasa pemrograman Python, C#, Java, dan PHP.
5. Mengetahui perbedaan implementasi *Class* dan *Constructor* dengan bahasa pemrograman Python, C#, Java, dan PHP.
6. Dapat menggunakan *Class* dan *Constructor* dalam aplikasinya pada pembuatam program secara cepat.

## Dasar Teori

* + 1. **Pemrograman Berbasis Objek (*Object-Oriented Programming* / OOP)**

*OOP* adalah suatu metode pemrograman yang berorientasi kepada objek. Tujuan dari *OOP* diciptakan adalah untuk mempermudah pengembangan program dengan cara mengikuti model yang telah ada di kehidupan sehari-hari. Jadi setiap bagian dari suatu permasalahan adalah objek, nah objek itu sendiri merupakan gabungan dari beberapa objek yang lebih kecil lagi. Contoh Pesawat, Pesawat adalah sebuah objek. Pesawat itu sendiri terbentuk dari beberapa objek yang lebih kecil lagi seperti mesin, roda, baling-baling, kursi, dll. Pesawat sebagai objek yang terbentuk dari objek-objek yang lebih kecil saling berhubungan, berinteraksi, berkomunikasi dan saling mengirim pesan kepada objek-objek yang lainnya. Begitu juga dengan program, sebuah objek yang besar dibentuk dari beberapa objek yang lebih kecil, objek-objek itu saling berkomunikasi, dan saling berkirim pesan kepada objek yang lain.

* + 1. ***Class***

*Class* merupakan suatu blueprint atau cetakan untuk menciptakan suatu instant dari object. class juga merupakan grup suatu *object* dengan kemiripan *attributes/properties*, *behaviour* dan relasi ke *object* lain.  
Contoh : *Class Person, Vehicle, Tree, Fruit* dan lain-lain.

* + 1. ***Construction***

*Construction* merupakan suatu method yang akan memberikan nilai awal pada saat suatu objek dibuat. Pada saat program dijalankan, *constructor* akan langsung memberikan nilai awal pada saat perintah new, membuat suatu objek  
Pada saat kita bekerja dengan *constructor,* hal mendasar yang perlu diperhatikan, yaitu:  
1. Nama *Constructor* sama dengan nama *Class.*

2. Tidak ada *return type* yang diberikan kedalam *Constructor Signature*.

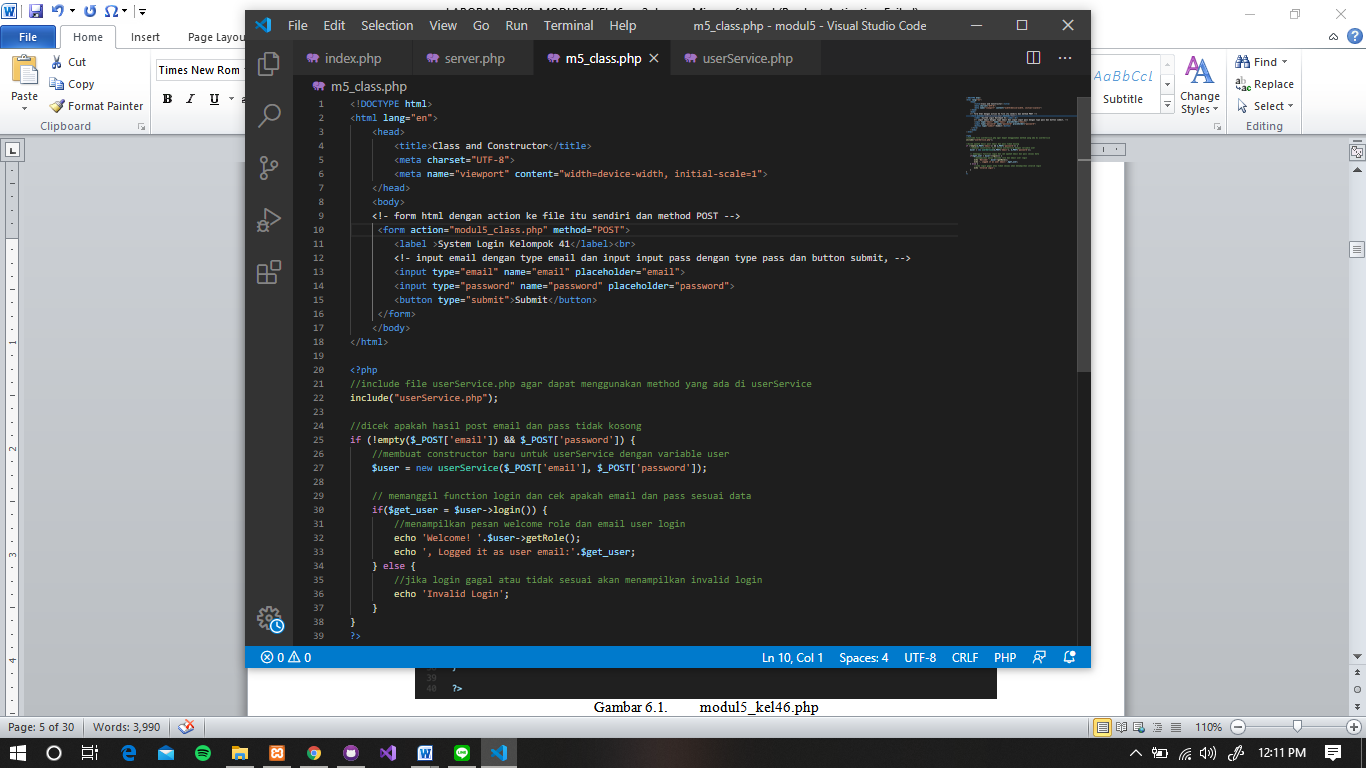
3. Tidak ada *return stetement*, didalam tubuh *constructor*.

*(sumber :* [*http://sekelebetinfo.blogspot.com/2014/08/pengertian-dari-oop.html*](http://sekelebetinfo.blogspot.com/2014/08/pengertian-dari-oop.html)*)*

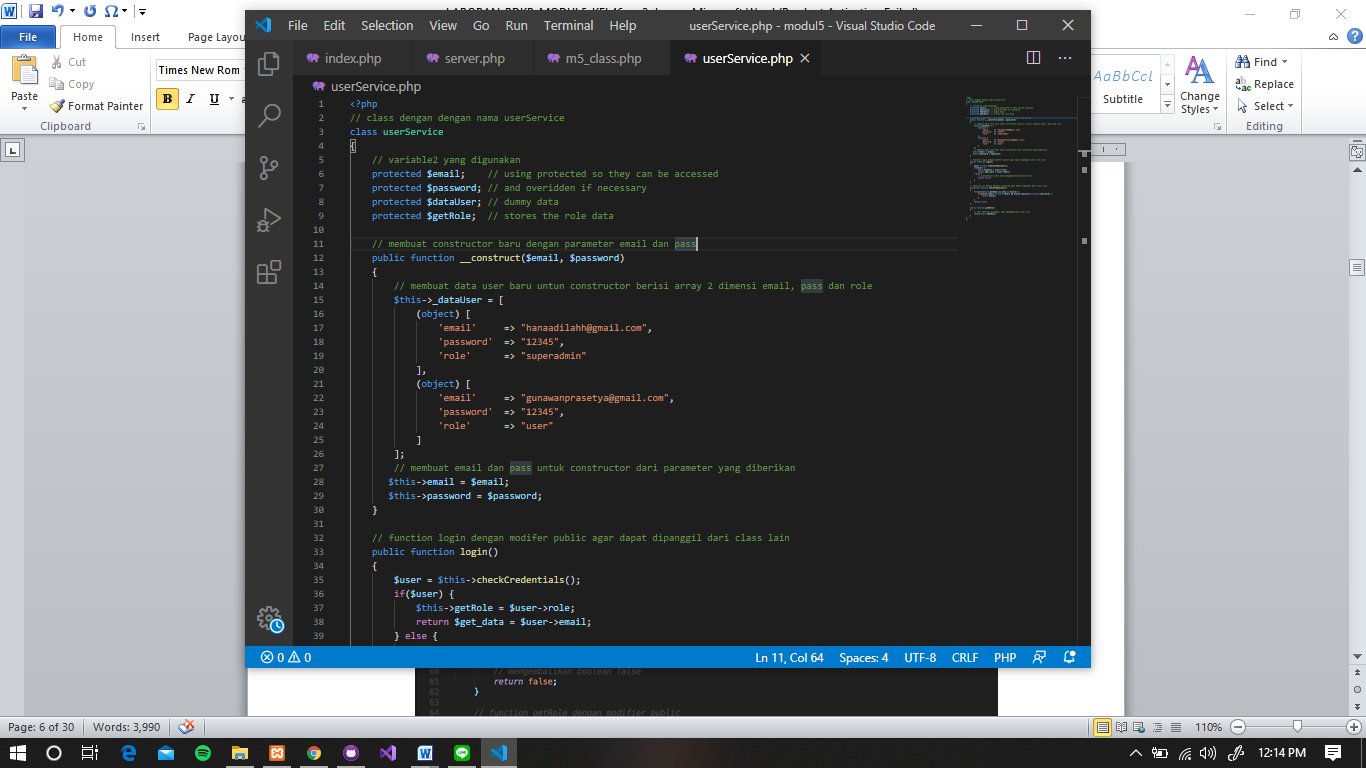
## Hasil dan Analisa Percobaan

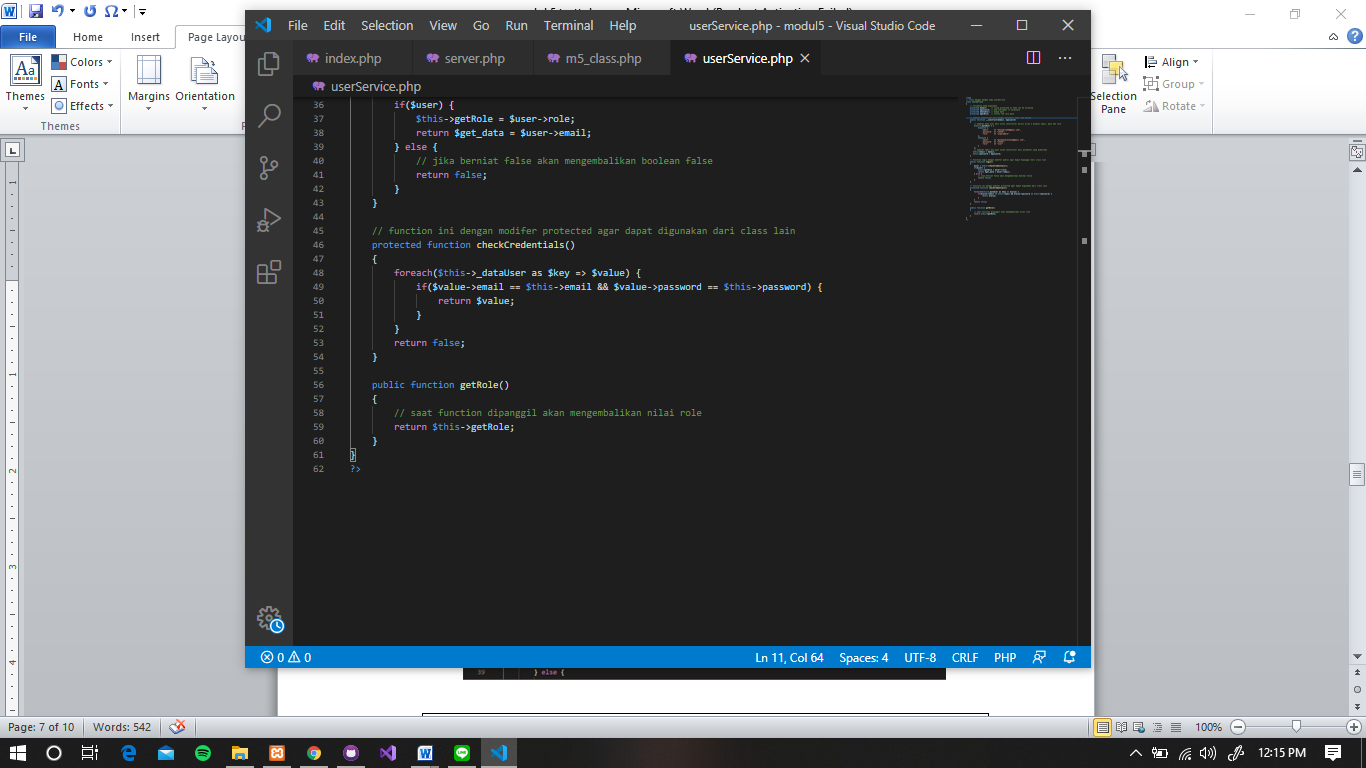
1. ***Class* pada PHP**

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <title>Class and Constructor</title>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1">  </head>  <body>  <!-- form html dengan action ke file itu sendiri dan method POST -->  <form action="modul5\_class.php" method="POST">  <label >System Login Kelompok 41</label><br>  <input type="email" name="email" placeholder="email">  <input type="password" name="password" placeholder="password">  <button type="submit">Submit</button>  </form>  </body>  </html>  <?php  // include file userService.php agar bisa menggunakan method yang ada didalam class userService  include("userService.php");  // mengecek apakah hasil post email dan password tidak kosong  if (!empty($\_POST['email']) && $\_POST['password']) {  // membuat constructor baru untuk userService dengan variabel user  $user = new userService($\_POST['email'], $\_POST['password']);  // memanggil function login dan mengecek apakah email dan password sudah sesuai dengan data  if($get\_user = $user->login()) {  // menampilkan pesan welcome role dan email user yang login  echo 'Welcome '.$user->getRole();  echo ', Logged it as user email:'.$get\_user;  } else {  // jika login gagal atau email dan password tidak sesuai akan menampilkan invalid login  echo 'Invalid Login';  }  }  ?> |

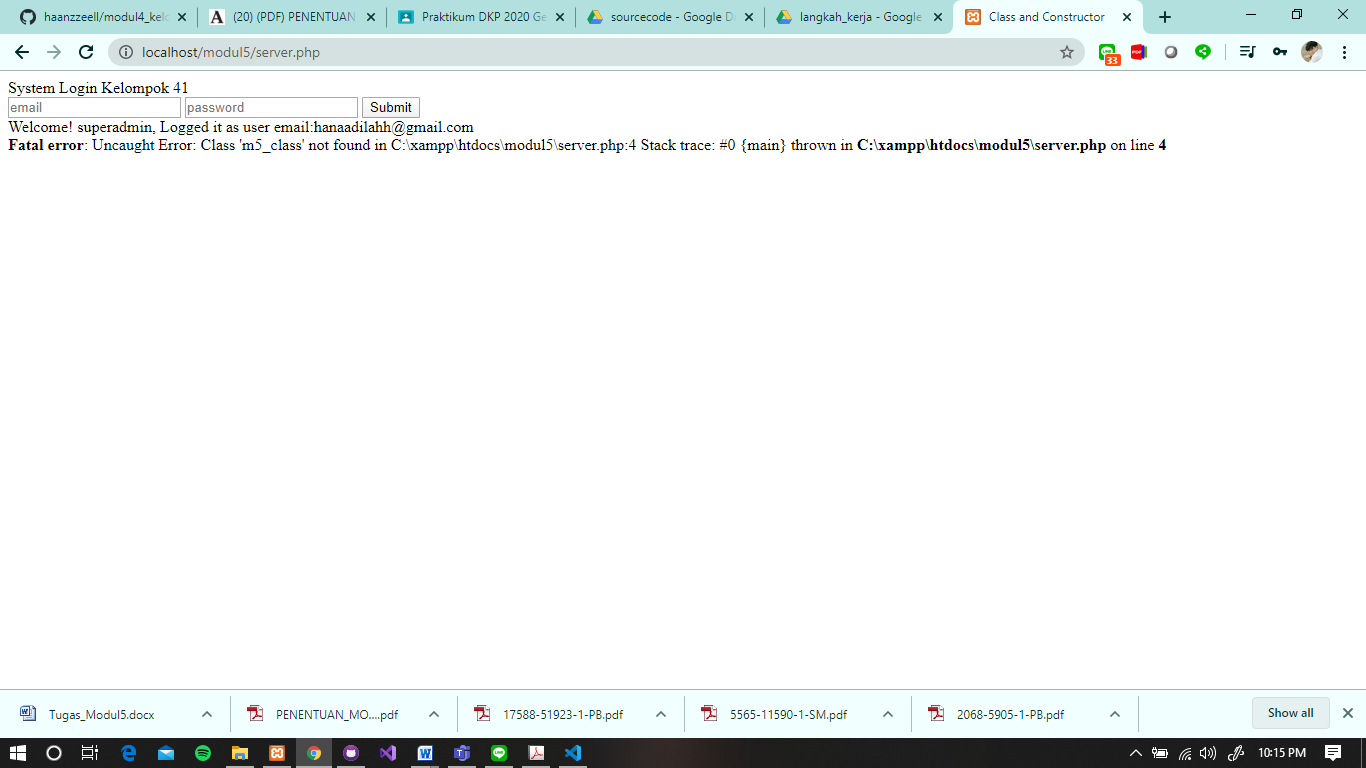


Gambar 6. 1 m5\_class.php





Gambar 6. 2 Class userService.php



Gambar 6. 3 Output Class pada PHP

Pada program PHP diatas dalam membuat *class.* Dengan include("userService.php"); yang fungsinya untuk menyertakan file lain ke dalam suatu program dengan *string* userService. Namun jika file yang disertakan tidak ada maka program masih akan berjalan walaupun tampilannya eror. kemudian dilanjutkan dengan *variabel* *object user* yang kemudian dibuat *class* baru dengan *string* *email* dan *password*. Dilanjutkan dengan pengkondisian *if* jika get\_user yang diambil dengan *method* login yang ditandai dengan panah. Khusus PHP, *method* ditandai dengan panah (->) bukan dengan titik (.). jika kondisi pertama benar maka akan ditampilkan teks berhasil *login*. Dan jika kondisi pertama salah maka kondisi kedua akan berlaku dan *output* yang ditampilkan adalah *invalid login*.

1. ***Constructor* pada PHP**

<?php

// class dengan dengan nama userService

class userService

{

    // variable2 yang digunakan

    protected $email;    // using protected so they can be accessed

    protected $password; // and overidden if necessary

    protected $dataUser; // dummy data

    protected $getRole;  // stores the role data

    // membuat constructor baru dengan parameter email dan pass

    public function \_\_construct($email, $password)

    {

        // membuat data user baru untun constructor berisi array 2 dimensi email, pass dan role

        $this->\_dataUser = [

            (object) [

                'email'     => "hanaadilahh@gmail.com",

                'password'  => "12345",

                'role'      => "superadmin"

            ],

            (object) [

                'email'     => "gunawanprasetya@gmail.com",

                'password'  => "12345",

                'role'      => "user"

            ]

        ];

        // membuat email dan pass untuk constructor dari parameter yang diberikan

       $this->email = $email;

       $this->password = $password;

    }

    // function login dengan modifer public agar dapat dipanggil dari class lain

    public function login()

    {

        $user = $this->checkCredentials();

        if($user) {

            $this->getRole = $user->role;

            return $get\_data = $user->email;

        } else {

            // jika berniat false akan mengembalikan boolean false

            return false;

        }

    }

    // function ini dengan modifer protected agar dapat digunakan dari class lain

    protected function checkCredentials()

    {

        foreach($this->\_dataUser as $key => $value) {

            if($value->email == $this->email && $value->password == $this->password) {

                return $value;

            }

        }

        return false;

    }

    public function getRole()

    {

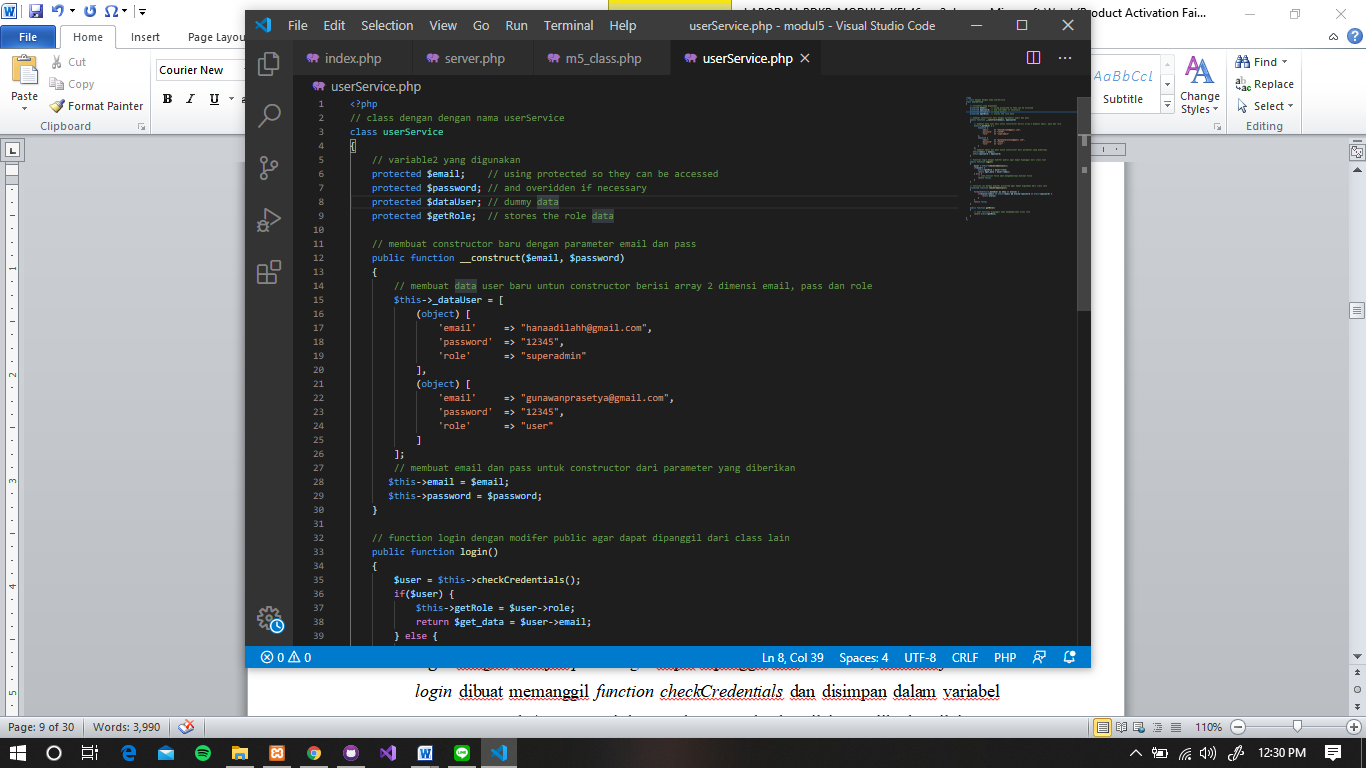
        // saat function dipanggil akan mengembalikan nilai role

        return $this->getRole;

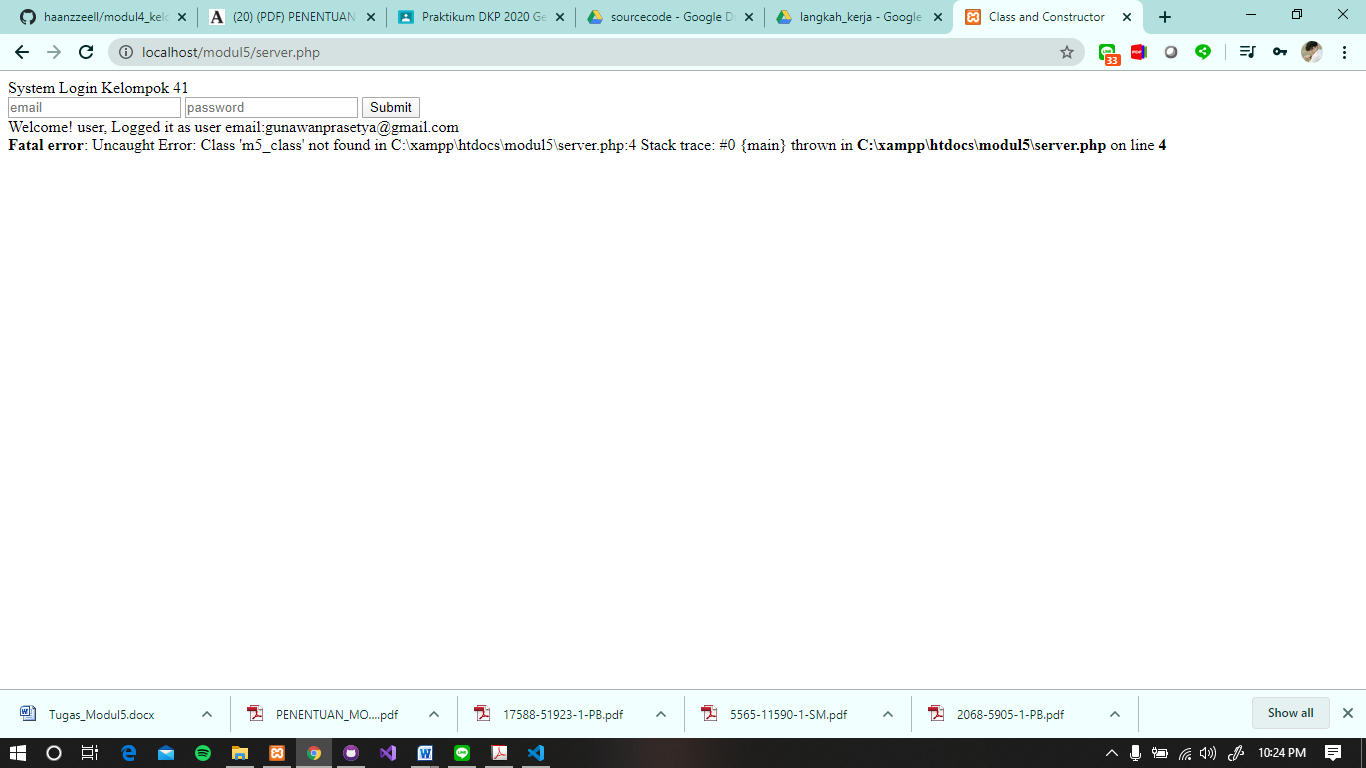
    }

}

?>



Gambar 6. 4 Constructor pada userService.php



Gambar 6. 5 Output Constructor pada PHP

Pada program PHP diatas dalam membuat *constructor*. Dengan *modifier* protect agar bisa di akses. Lalu dilanjutkan dengan public function\_\_construct($email, $password) yang maksudnya *function* dengan *construct* *string* email dan password. Kemudian *construct* $this yaitu data user lalu *variabel* *object email, password* dan *role* *diinput* oleh *superAdmin* dan *user*. Selanjutnya mengeset email yang di *class* nya dengan isinya *email* dari input yang dimasukkan *user*. Kemudian dilanjutkan dengan *func* login() dimana *method* *checkCredentials* akan mengecek apakah data yang dimasukkan oleh user sama dengan data di *constructor*. Dikatakan benar jika kondisi pertama data role yang diambil sama dengan data user maka nilai yang dikembalikan adalah $get\_data = $user->email; dan jika salah maka nilai yang dikembalikan adalah false. Selanjutnya protected *funct checkCredentials*. *foreach* fungsinya untuk *looping array*, dimana mengeset data user dan mengeset $key menjadi $value atau nilai. jika benar *value* yang di set email sama dengan yang di set email oleh user dan *value* yang di set password sama dengan yang di set password oleh user maka nilai yang dikembalikan adalah $value; yaitu *email, password dan role* dan jika salah maka nilai yang dikembalikan adalah *false*.

1. ***Class* pada Python**

# mengimport class userService dari file userService

from userService import userService

print("System Login Kelompok 41!\n")

# membaca input user dan menyimpan dalam variabel email

email = input("Email: ")

# membaca input user dan menyimpan dalam variabel password

password = input("Password: ")

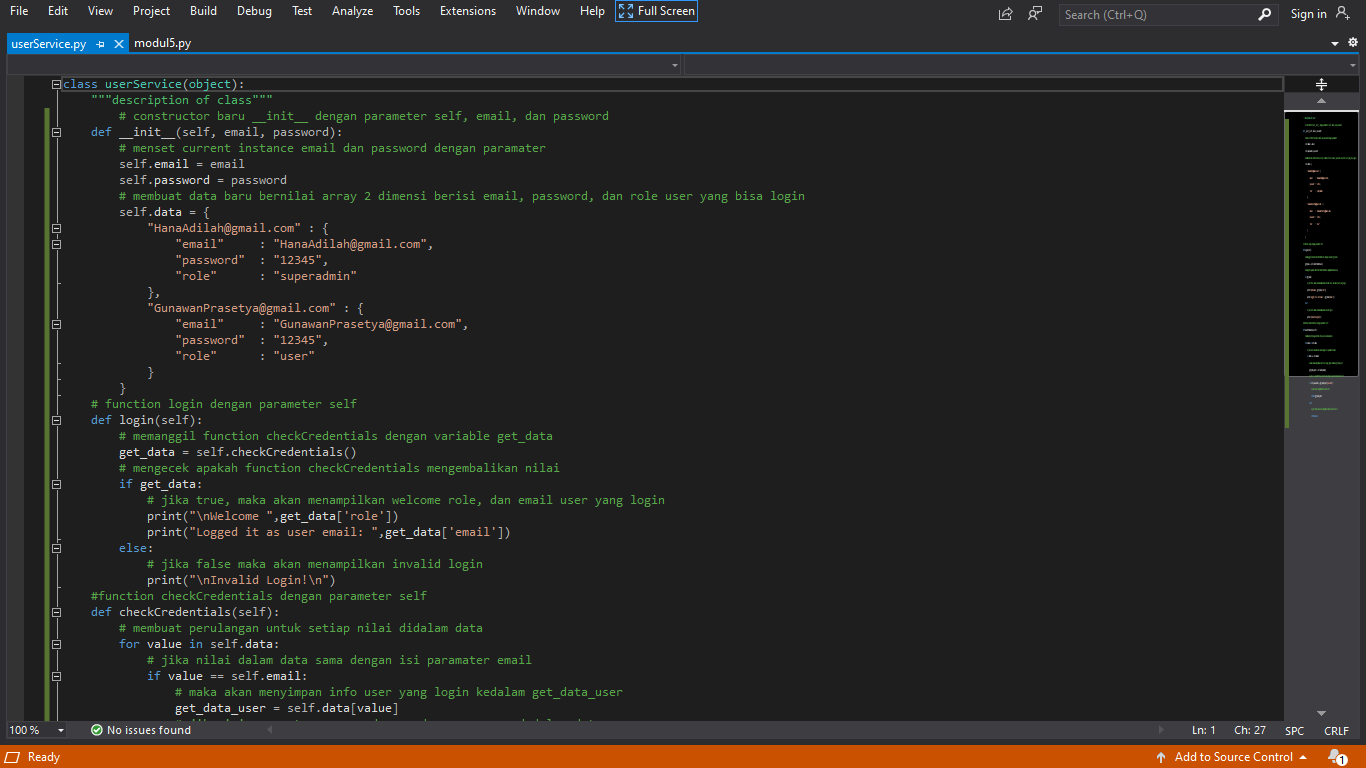
# membuat constructor baru userService dengan isi parameter input email dan password

# dengan variabel get\_class

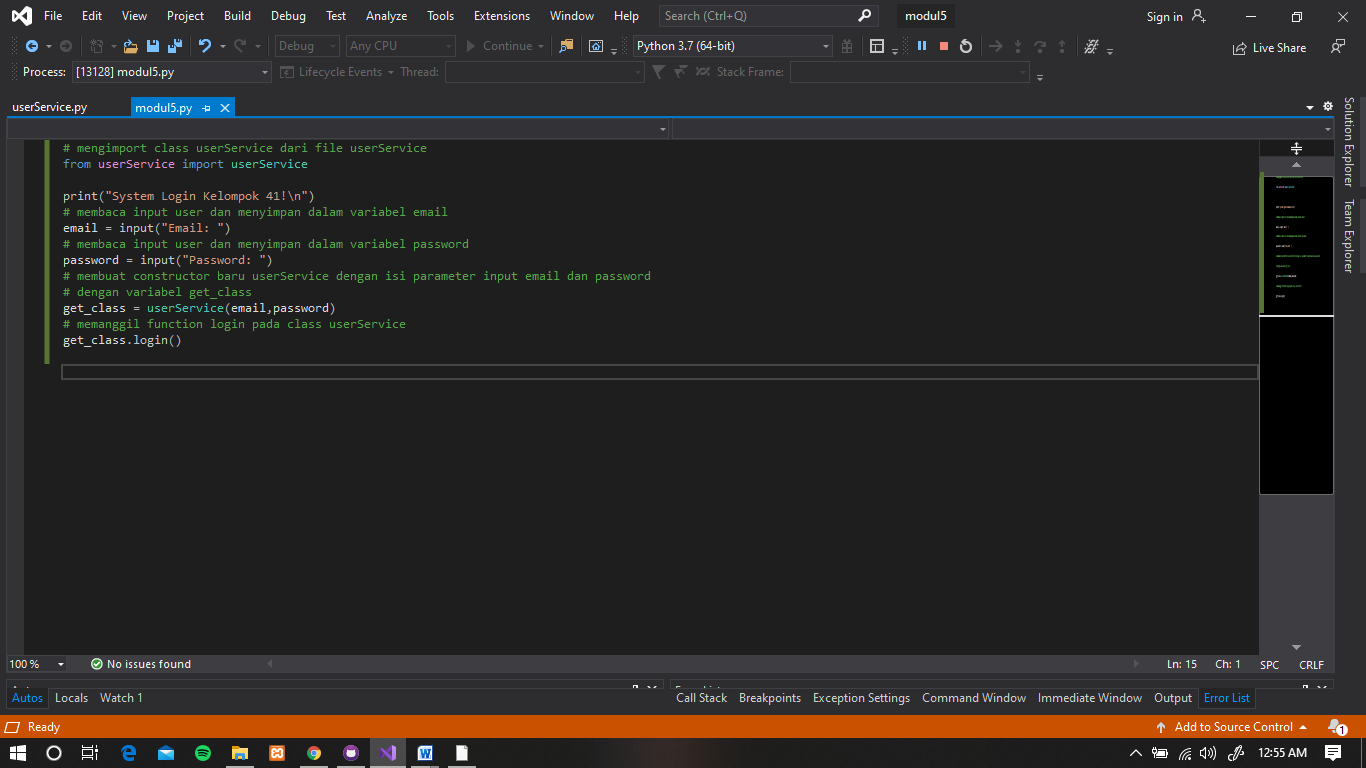
get\_class = userService(email,password)

# memanggil function login pada class userService

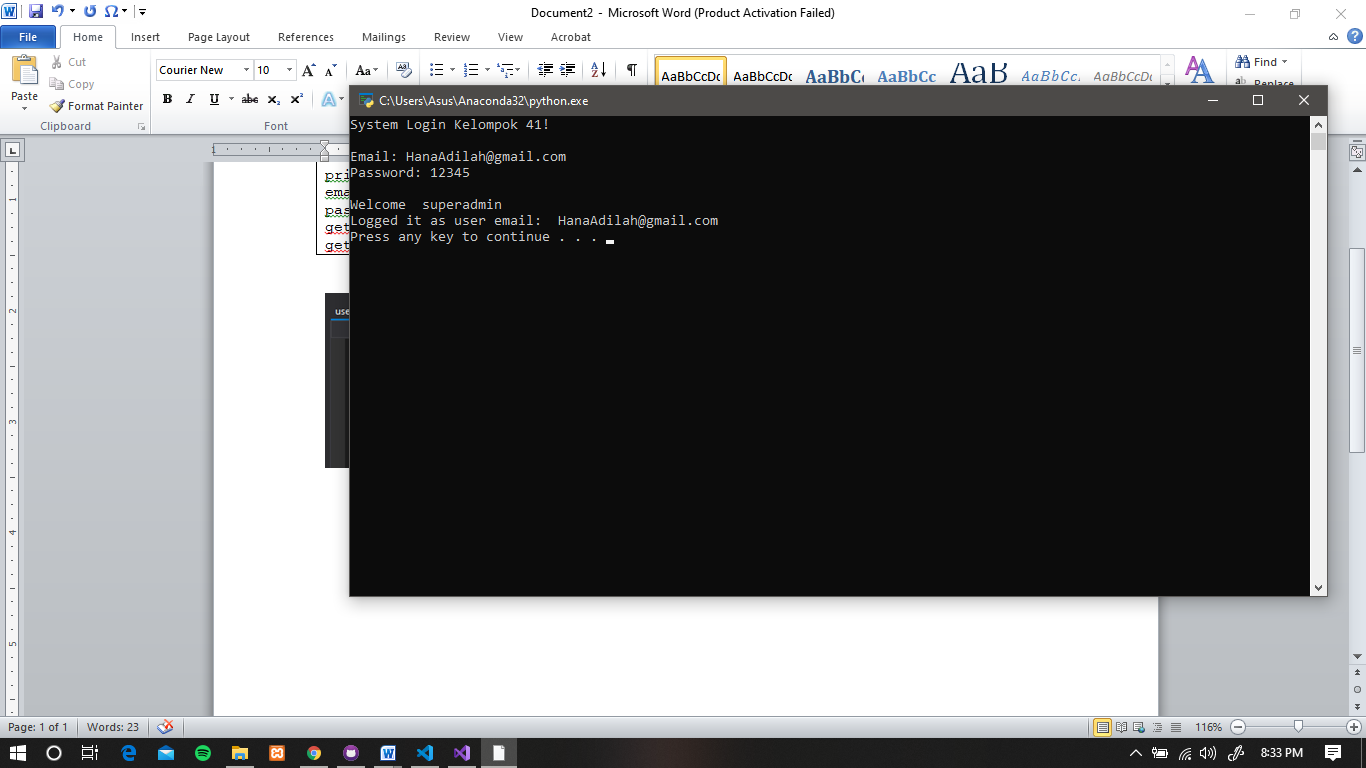
get\_class.login()



Gambar 6. 5 Class userService pada userService.py



Gambar 6. 6 modul5.py



Gambar 6. 7 Output pada Python

Pada program diatas menggunakan *class* pada python. from userService import userService merupakan pemanggilan berkas dan juga sekaligus memanggil *class* lain pada berkas tersebut. Kemudian *print* pernyataan *login, email* dan *password* yang akan *diinput* ketika program dijalankan. Lalu mengambil data dari *object* *class* dengan menambah get yang merupakan cara pengambilan data *object* dari *class* tersebut. Lalu untuk memanggil *method* login() dapat dilakukan dengan *variabel* ditambahkan .login().

1. ***Construction* pada Python**

class userService(object):

"""description of class"""

# constructor baru \_\_init\_\_ dengan parameter self, email, dan password

def \_\_init\_\_(self, email, password):

# menset current instance email dan password dengan paramater

self.email = email

self.password = password

# membuat data baru bernilai array 2 dimensi berisi email, password, dan role user yang bisa login

self.data = {

"HanaAdilah@gmail.com" : {

"email" : "HanaAdilah@gmail.com",

"password" : "12345",

"role" : "superadmin"

},

"GunawanPrasetya@gmail.com" : {

"email" : "GunawanPrasetya@gmail.com",

"password" : "12345",

"role" : "user"

}

}

# function login dengan parameter self

def login(self):

# memanggil function checkCredentials dengan variable get\_data

get\_data = self.checkCredentials()

# mengecek apakah function checkCredentials mengembalikan nilai

if get\_data:

# jika true, maka akan menampilkan welcome role, dan email user yang login

print("\nWelcome ",get\_data['role'])

print("Logged it as user email: ",get\_data['email'])

else:

# jika false maka akan menampilkan invalid login

print("\nInvalid Login!\n")

#function checkCredentials dengan parameter self

def checkCredentials(self):

# membuat perulangan untuk setiap nilai didalam data

for value in self.data:

# jika nilai dalam data sama dengan isi paramater email

if value == self.email:

# maka akan menyimpan info user yang login kedalam get\_data\_user

get\_data\_user = self.data[value]

# jika isi parameter password sama dengan password dalam data user

if self.password == get\_data\_user['password']:

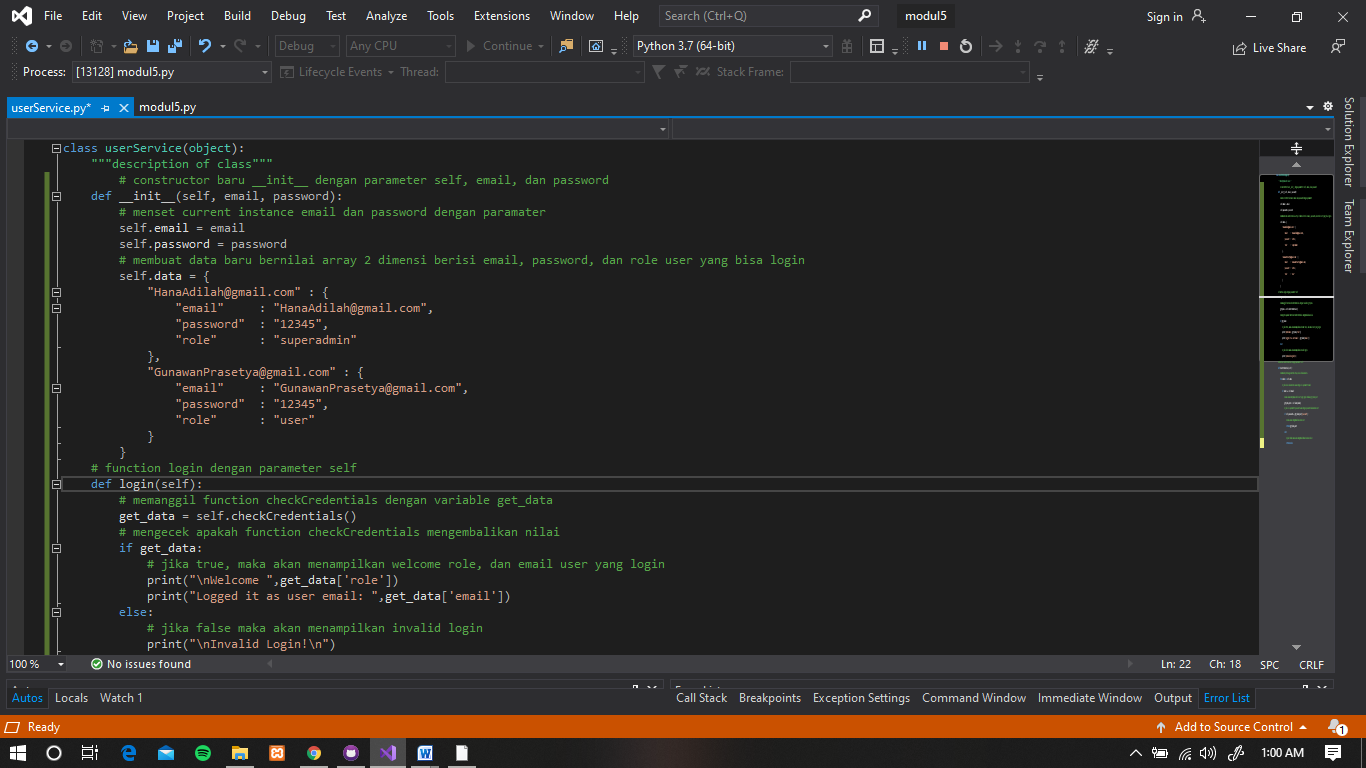
# maka akan mengembalikan data user

return get\_data\_user

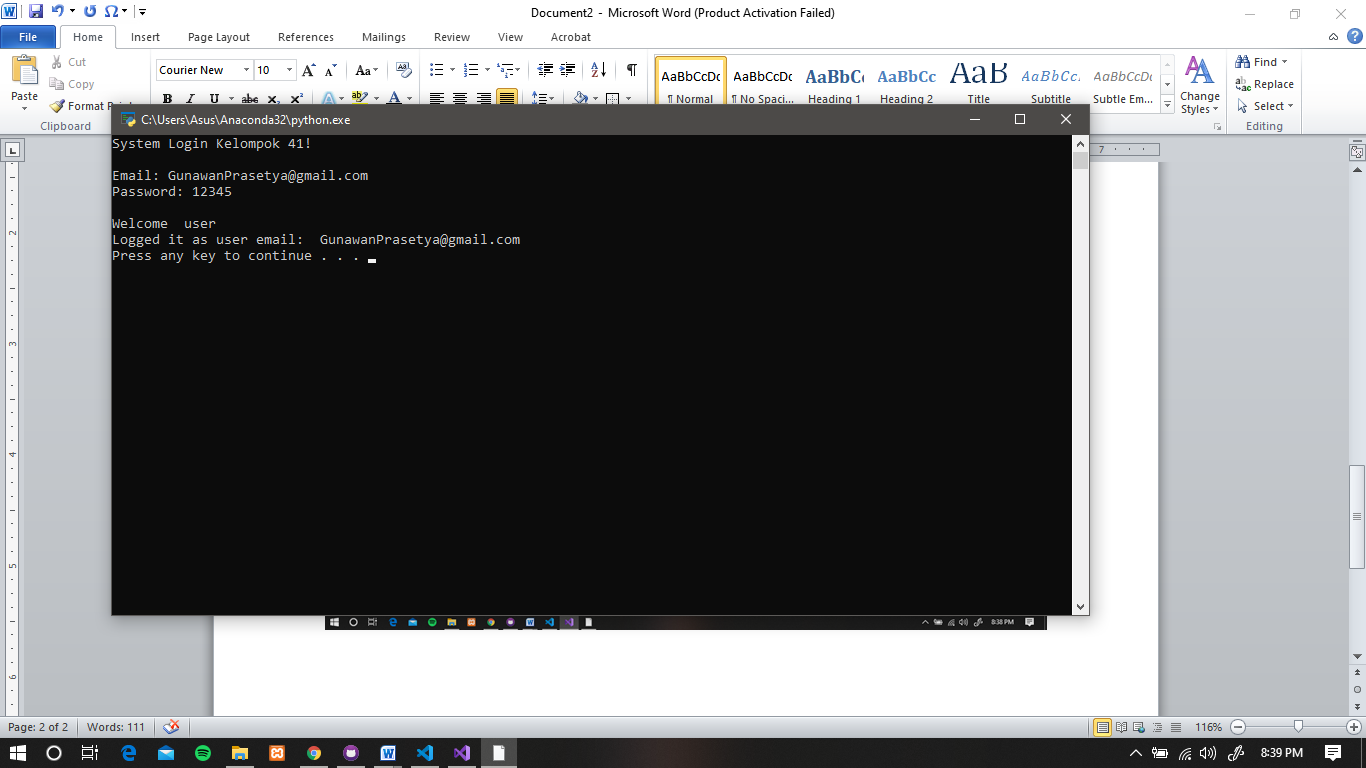
else:

#jika tidak maka akan mengembalikkan boolean false

return False



Gambar 6. 8 Constructor pada userService.py



Gambar 6. 9 Output pada Python

Pembuatan *constructor* dengan penambahan \_\_init\_\_. Di python menggunakan *parameter* self. Seperti sama dengan this. yang fungsinya menunjukkan *object* itu sendiri atau sebagai akses ke dalam *constructor* di *class* tersebut. Lalu *inisiasi* data *email, password* dan *role* yang telah dibuat, yang merupakan *variabel* *object* nya. Kemudian dilanjutkan dengan mengambil *object* data yaitu get\_data = self.checkCredentials(). Lalu, ada pengkondisian yaitu *if* jika data *login* benar dan *else* jika kondisi pertama salah. Lalu definisi yang pengkondisian *if* jika email dan data user berhasil di *check* maka nilai akan dikembalikan. Kemudian jika salah maka nilai data tidak akan diambil.

1. ***Class* pada C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace m5\_41

{

class Program

{

// main function

static void Main(string[] args)

{

// deklarasi variabel

string email, password;

Console.Write("Email : ");

// membaca input user dan disimpan dalam variabel email

email = Console.ReadLine();

Console.Write("Password : ");

// membaca input user dan disimpan dalam variabel password

password = Console.ReadLine();

// membuat object baru dengan variabel data dan menginisialisasi constructor userService

// dengan isi parameter input email dan password

userService data = new userService(email, password);

// memanggil function login

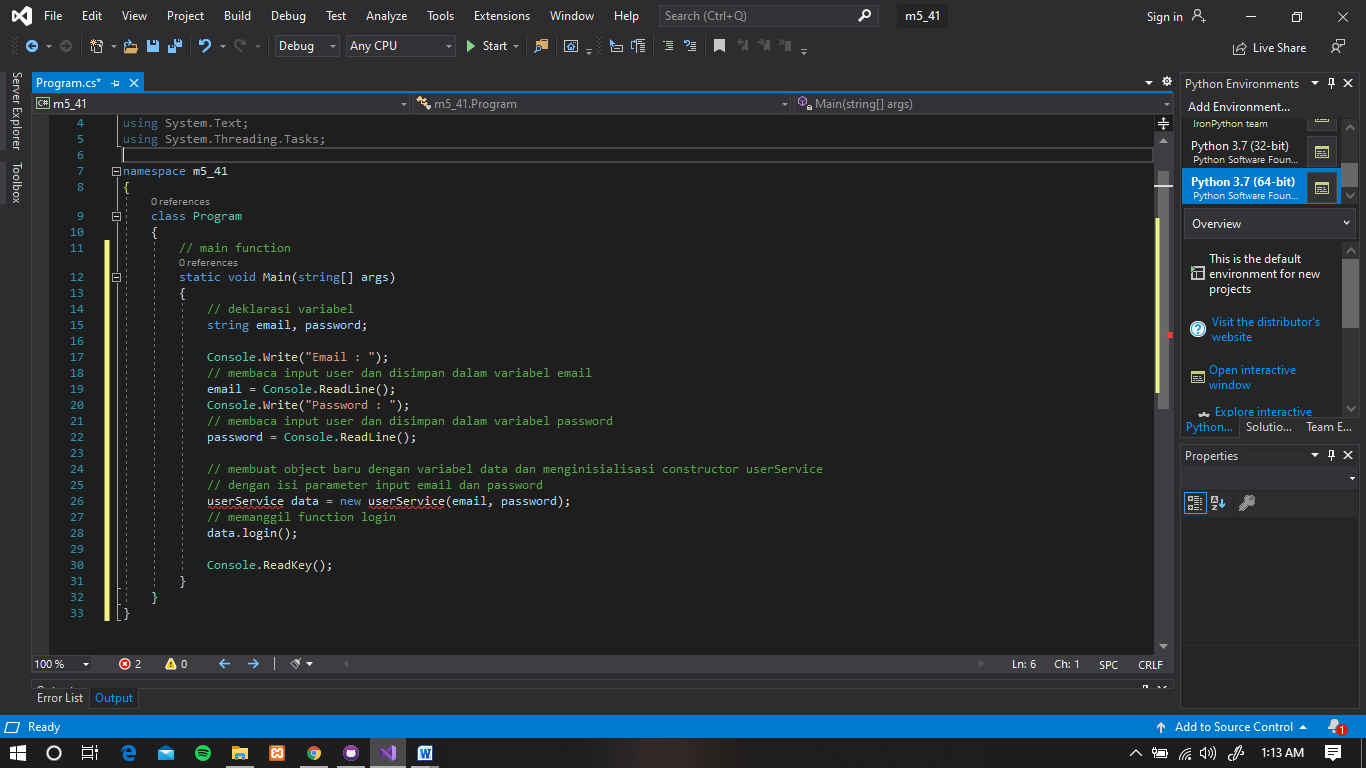
data.login();

Console.ReadKey();

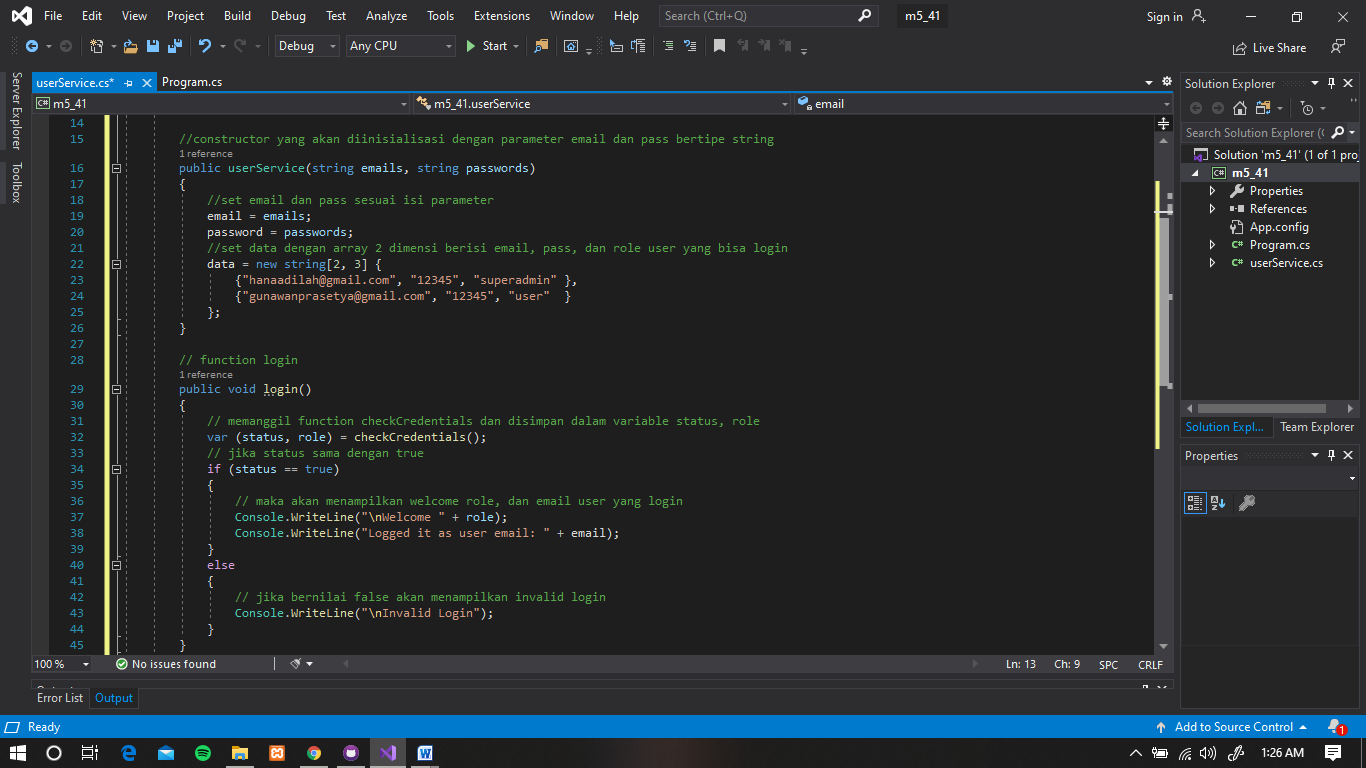
}

}

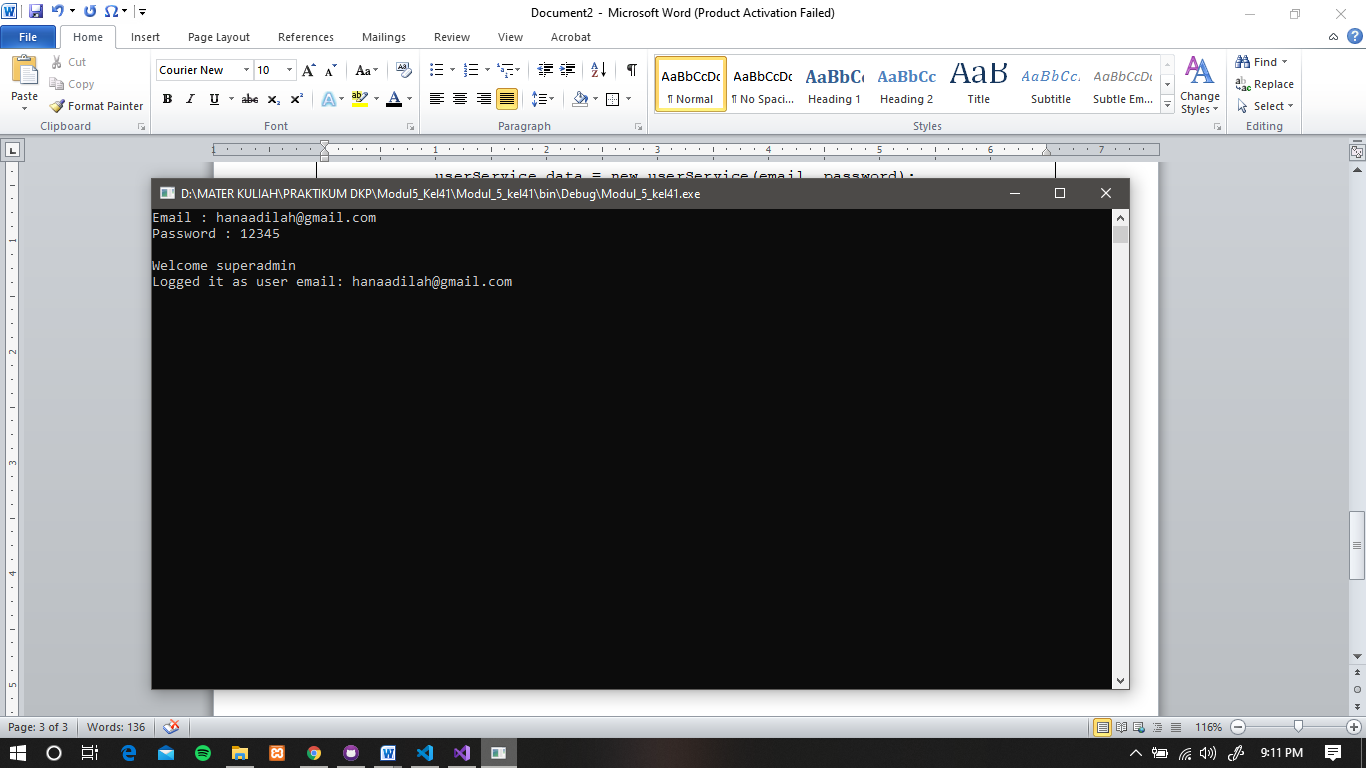
}



Gambar 6. 10 Class Program pada m5\_41.cs



Gambar 6. 11 Class userService pada userService.cs



Gambar 6. 12 Output Class pada C#

Pada program diatas, *class* dibuat dengan bahasa C#. pemberian *string* nama dan *password*. Kemudian masukkan *variabel* *email* dan *password*. Console.Write fungsinya tidak menambah baris baru. Sedangkan Console.ReadLine(); (bertipe data *string*) yaitu meminta nilai atau *menginput* data dari *variabel* *email* dan *password*. Lalu membuat *object* *class* baru dengan cara menambahkan new pada *userservice* dengan parameter *email* dan *password* yang nilainya didapat dari *inputan* yang dimasukkan. Untuk memanggil *method* login() cukup dengan data.login(). Lalu teks akan ditampilkan pada program

1. ***Construction* pada C#**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace m5\_41

{

//class userService

class userService {

// deklarasi variable untuk class userService

private string[,] data;

private string email, password, roles = "";

//constructor yang akan diinisialisasi dengan parameter email dan pass bertipe string

public userService(string emails, string passwords)

{

//set email dan pass sesuai isi parameter

email = emails;

password = passwords;

//set data dengan array 2 dimensi berisi email, pass, dan role user yang bisa login

data = new string[2, 3] {

{"hanaadilah@gmail.com", "12345", "superadmin" },

{"gunawanprasetya@gmail.com", "12345", "user" }

};

}

// function login

public void login()

{

// memanggil function checkCredentials dan disimpan dalam variable status, role

var (status, role) = checkCredentials();

// jika status sama dengan true

if (status == true)

{

// maka akan menampilkan welcome role, dan email user yang login

Console.WriteLine("\nWelcome " + role);

Console.WriteLine("Logged it as user email: " + email);

}

else

{

// jika bernilai false akan menampilkan invalid login

Console.WriteLine("\nInvalid Login");

}

}

private (bool, string) checkCredentials()

{

for (int i = 0; i < data.GetLength(0); i++)

{

if (data[i, 0] == email && data[i, 1] == password)

{

roles = data[i, 2];

return (true, roles);

}

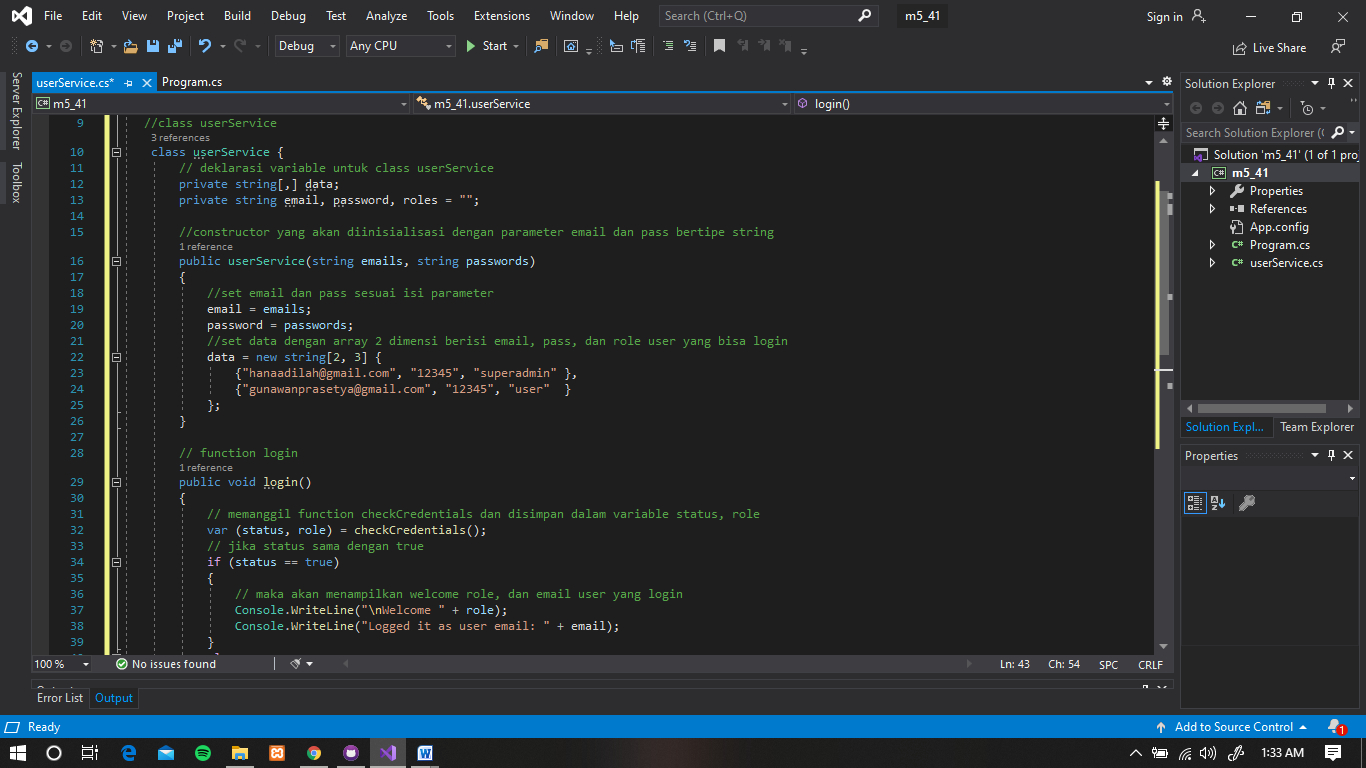
}

return (false, roles);

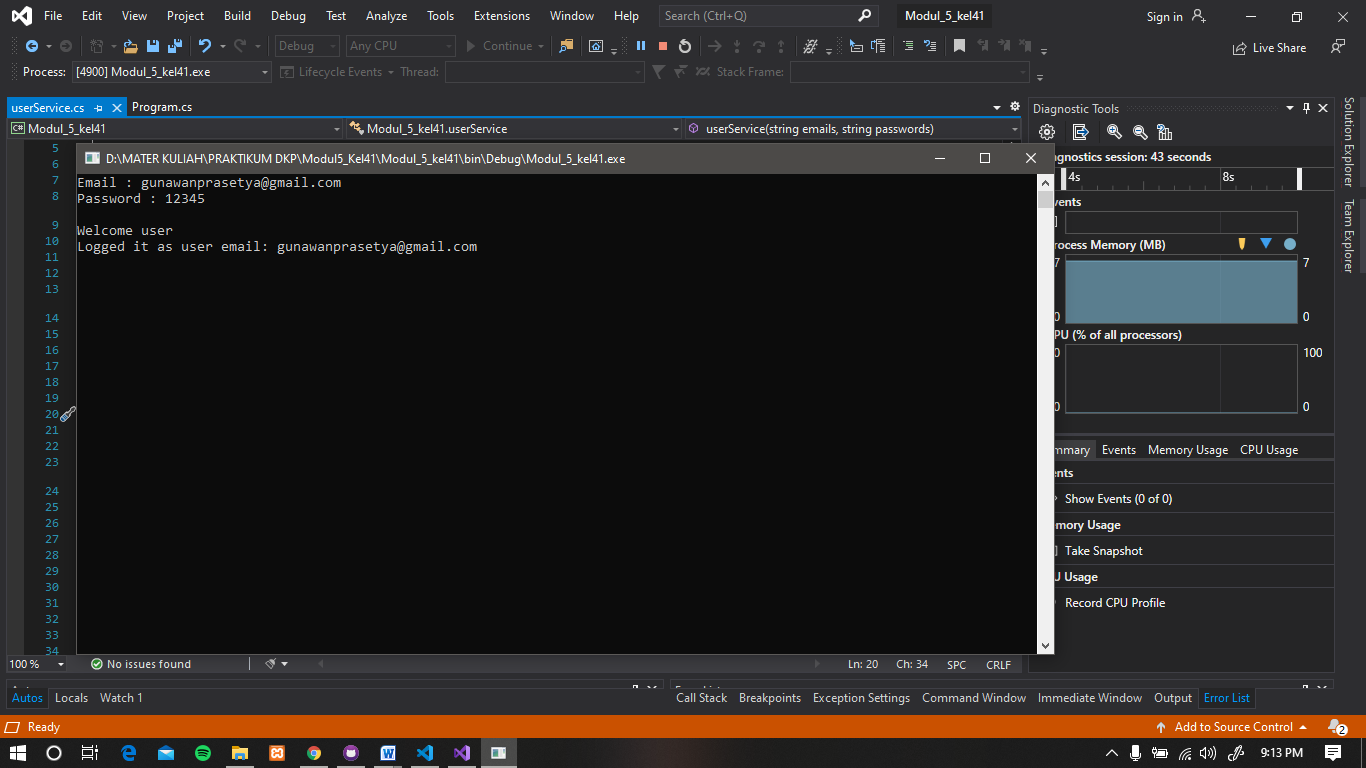
}

}

}



Gambar 6. 13 Constructor pada class userService.cs



Gambar 6. 14 *Output Constructor* pada C#

Pada program diatas menggunakan bahasa C# dalam membuat *Constructor*. Diawali dengan *modifier* private yang maksudnya tidak dapat diakses *class* lain kecuali melalui *function* dari *class* itu sendiri. Lalu dilanjutkan dengan *menginput variabel* object email dan password. Untuk data dibuat *string* baru dari *parameter* email dan password. Lalu dengan *modifier* public *function* *login()* dan disimpan dalam *variabel* status dan role jika status benar maka pengkondisian *if* dengan data *logged email* akan ditampilkan namun jika kondisi salah maka login tidak valid. Lalu *function* dengan check.credentials dengan *modifier* private yang akan mengembalikan *Boolean* dan nilai *string*. Dengan perulangan *for*, *insialisasi* i = 0, pengehenti dengan range 1 < dari jumlah data yang di ambil kemudian *increment* 1++ yaitu setiap perulangan data ditambah 1. Kondisi pertama jika data sama dengan inputan email dan password maka akan dimasukkan ke roles. Dan data akan dikembalikan ke *variabel true* dan *roles*. Namun jika gagal akan mengembalikan nilai yang salah ke *roled user*

1. ***Class* pada Java**

import java.util.Scanner;

// class baru dengan nama MODUL\_5

public class MODUL5\_5 {

// main function

public static void main(String[] args) {

// deklarasi variabel

String email, password;

// membuat object baru dengan nama input dan membentuk Scanner

Scanner input = new Scanner(System.in);

System.out.print("Email : ");

// membaca input user dan disimpan dalam variabel email

email = input.nextLine();

System.out.print("Password : ");

// membaca input user dan disimpan dalam variabel password

password = input.nextLine();

// membuat object satu bernilai constructor baru untuk userService dengan isi parameter input email dan password

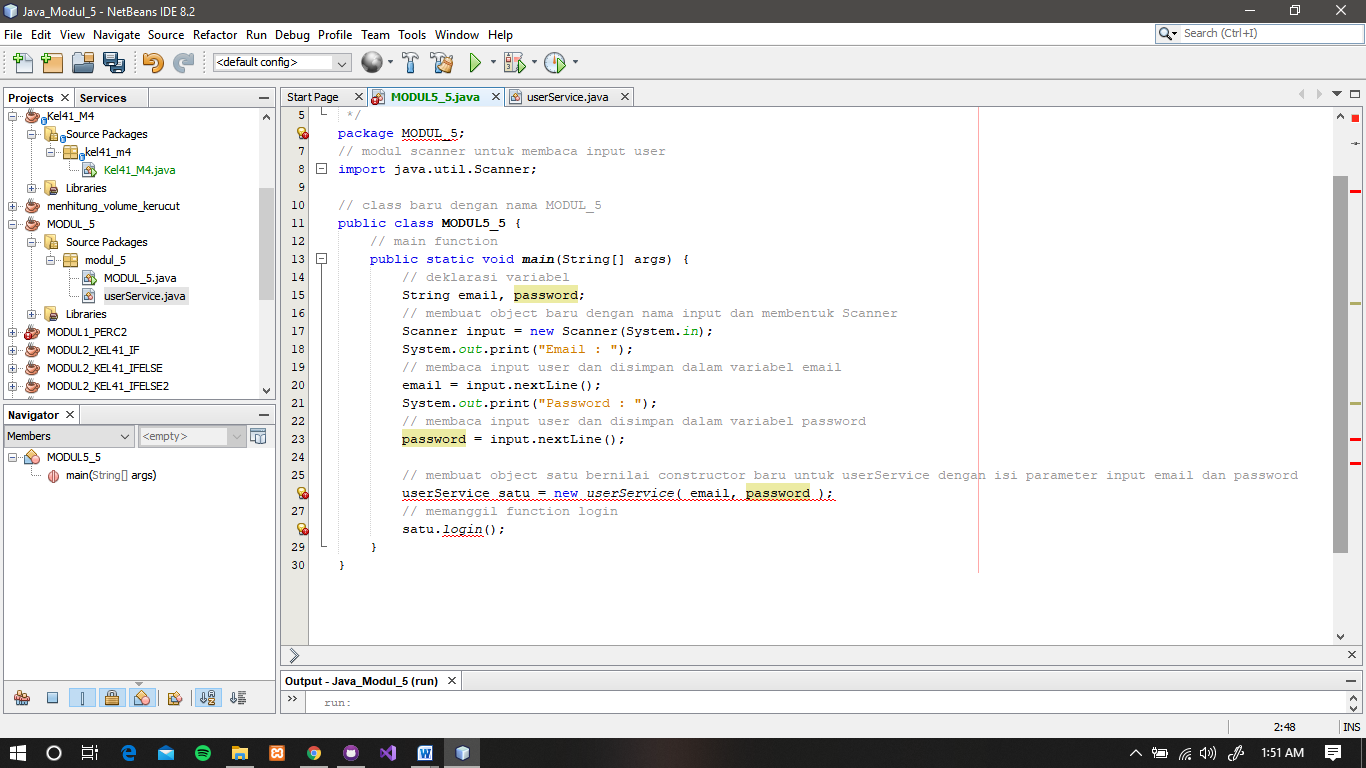
userService satu = new userService( email, password );

// memanggil function login

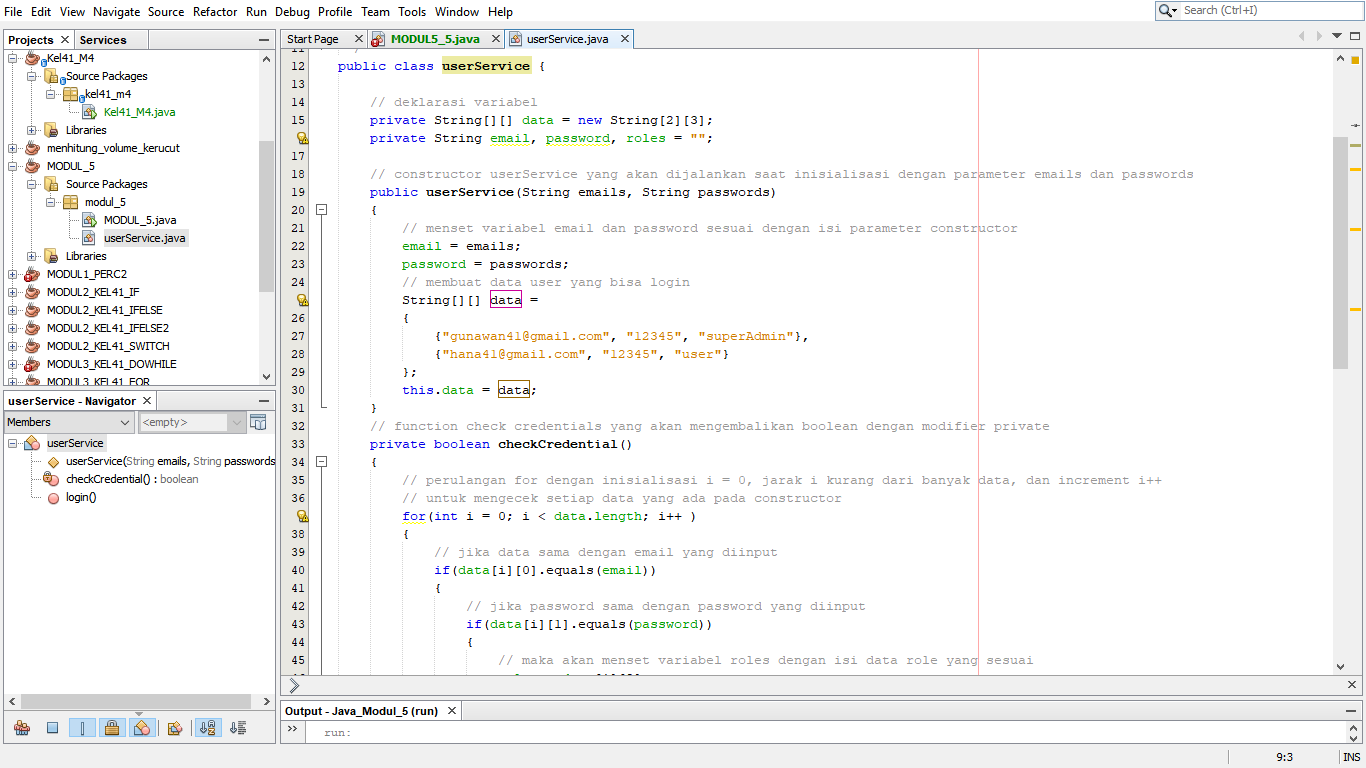
satu.login();

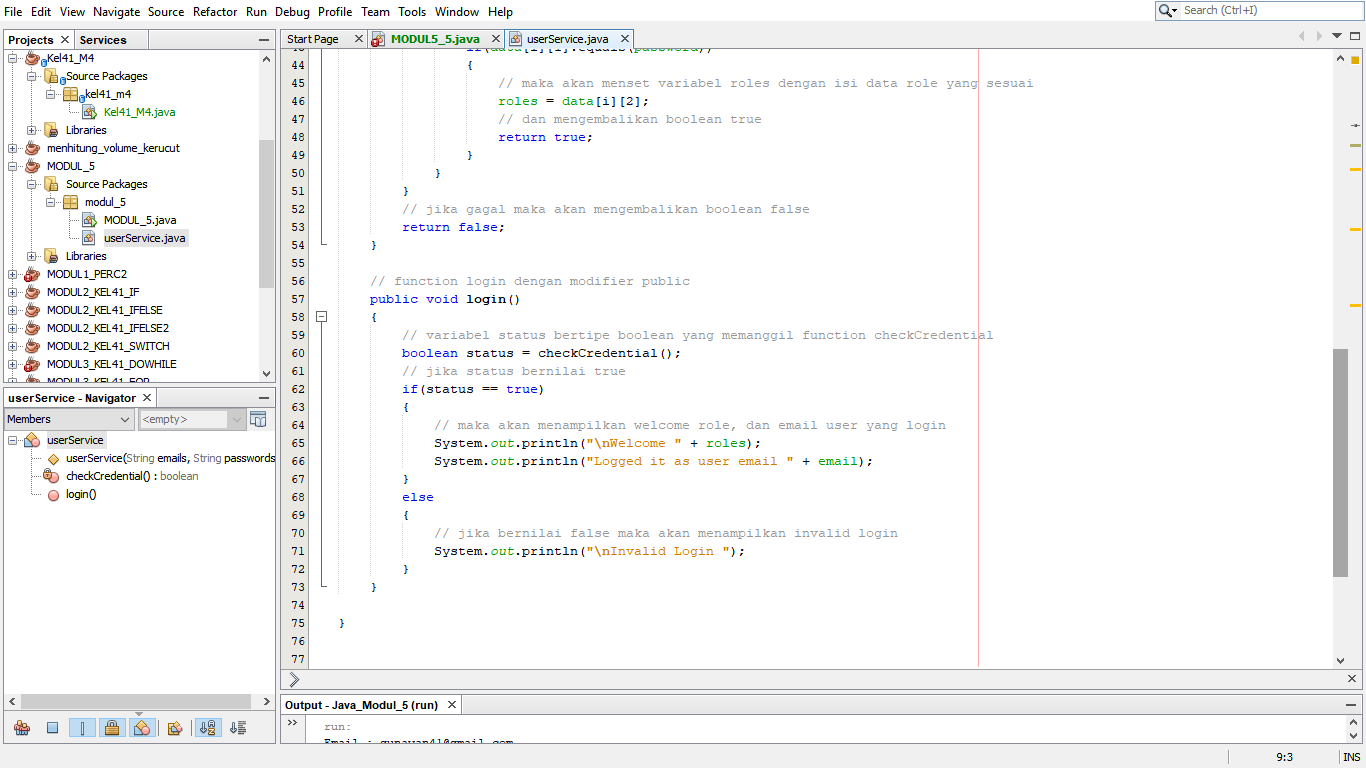
}

}

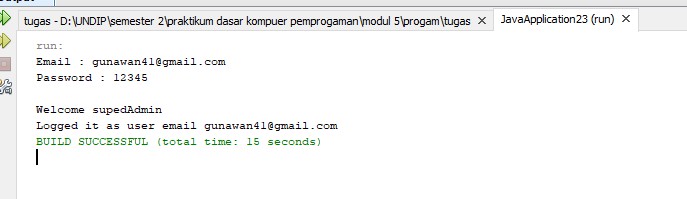


Gambar 6. 15 Class Modul\_5.java





Gambar 6. 16 Class userService.java



*Gambar 6*. 17 *Output Class* pada Java

Pada program diatas menggunakan bahasa Java untuk membuat *class.* import java.util.Scanner; maksudnya mengimport *scanner* ke program. Lalu dengan *modifier* public yang bisa di akses oleh *class* lain dengan *class* utama tipe data *string*. Lalu dengan tipe *string* email dan password. Scanner input = new Scanner(System.in); maksudnya mencetak hasil dari data dengan membuat *new scanner*. System.out.print("Email : "); menampilkan data email. Kemudian memasukkan nilai inputan email dari *string*. Selanjutnya menampilkan data password lalu memasukkan nilai inputan password dari *string*. Kemudian membuat object *class* baru dengan userService satu = new userService( email, password );. lalu untuk memanggil *method* login() cukup dengan satu.login(); maka *method* akan terpanggil.

1. ***Construction* pada Java**

public class userService {

// deklarasi variabel

private String[][] data = new String[2][3];

private String email, password, roles = "";

// constructor userService yang akan dijalankan saat inisialisasi dengan parameter emails dan passwords

public userService(String emails, String passwords)

{

// menset variabel email dan password sesuai dengan isi parameter constructor

email = emails;

password = passwords;

// membuat data user yang bisa login

String[][] data =

{

{"gunawan41@gmail.com", "12345", "superAdmin"},

{"hana41@gmail.com", "12345", "user"}

};

this.data = data;

}

// function check credentials yang akan mengembalikan boolean dengan modifier private

private boolean checkCredential()

{

// perulangan for dengan inisialisasi i = 0, jarak i kurang dari banyak data, dan increment i++

// untuk mengecek setiap data yang ada pada constructor

for(int i = 0; i < data.length; i++ )

{

// jika data sama dengan email yang diinput

if(data[i][0].equals(email))

{

// jika password sama dengan password yang diinput

if(data[i][1].equals(password))

{

// maka akan menset variabel roles dengan isi data role yang sesuai

roles = data[i][2];

// dan mengembalikan boolean true

return true;

}

}

}

// jika gagal maka akan mengembalikan boolean false

return false;

}

// function login dengan modifier public

public void login()

{

// variabel status bertipe boolean yang memanggil function checkCredential

boolean status = checkCredential();

// jika status bernilai true

if(status == true)

{

// maka akan menampilkan welcome role, dan email user yang login

System.out.println("\nWelcome " + roles);

System.out.println("Logged it as user email " + email);

}

else

{

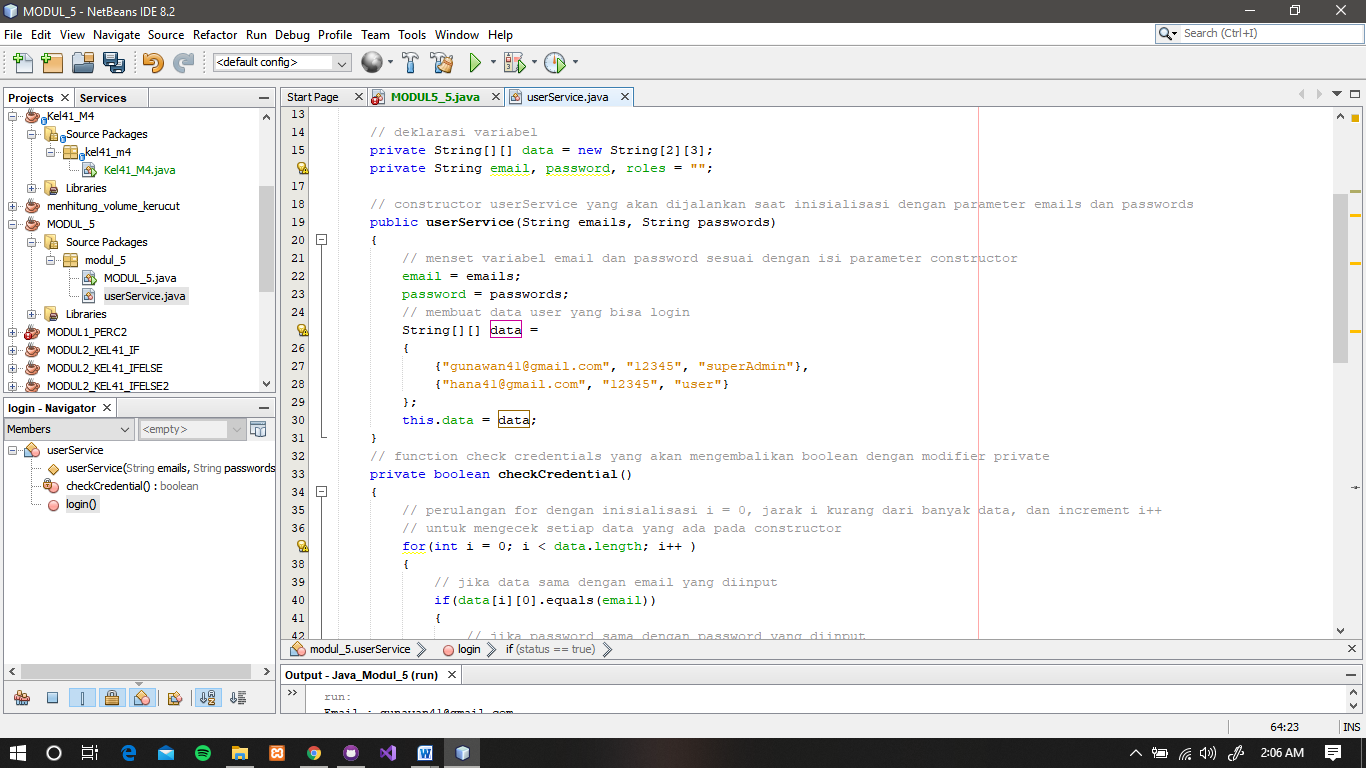
// jika bernilai false maka akan menampilkan invalid login

System.out.println("\nInvalid Login ");

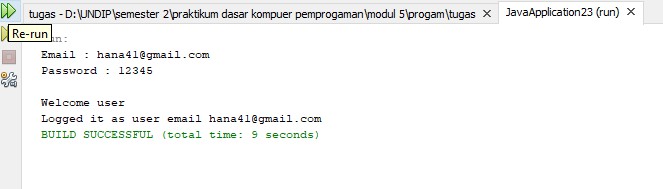
}

}

}



Gambar 6. 18 Constructor pada userService.java



*Gambar 6*. 19 *Output Constructor* pada Java

Pada program diatas menggunakan bahasa Java untuk membuat *construction.* Dengan *modifier* public untuk userService kemudian *variabelnya* ada *email* dan *password*. Dan juga tipe *array* 2 dimensi. Kemudian *string* data yang akan dimasukkan oleh *superAdmin* dan *User.* Pada java menggunakan *parameter* khusus yaitu this. Sama seperti python yang menggunakan *parameter* self. yang fungsinya untuk menunjukkan *object* itu sendiri. Kemudian dilanjutkan dengan *method* *modifier* private yang tidak bisa diakses *class* lain. *Method checkCredential* fungsinya untuk mengecek apakah inputan yang dimasukkan sama pada data *constructor*. Maka adanya perulangan *for* dengan *insialisasi* i = 0, dengan range i < dari jumlah data pada *string* lalu ada *increment* i++ yaitu setiap perulangan akan ada penambahan nilai. Lalu pengkondisian bertingkat dengan kondisi *if* pertama yaitu data i dan 1 akan dibandingkan dengan *string* .equals() dengan *variabel* yang dimaksud adalah email. Kemudian kondisi *if* yang kedua data yang dibandingan yaitu *variabel* *password*. Jika benar maka akan mengembalikan nilai *true* dan jika salah maka nilai yang dikembalikan adalah *false*. Kemudian dilanjutkan dengan *method modifier* public void login() dengan tipe data *Boolean* mengecek status. Jika status benar maka hasil akan ditampilkan yaitu berhasil *login* atau berupa teks dan jika salah maka hasil yang ditampilkan adalah *invalid login*.

Link github :

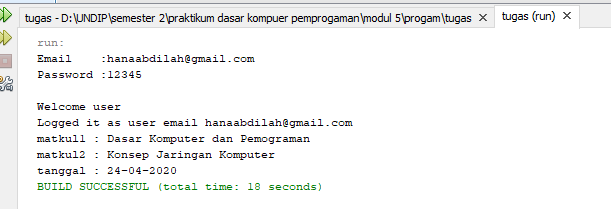
## Tugas

Main *Class*

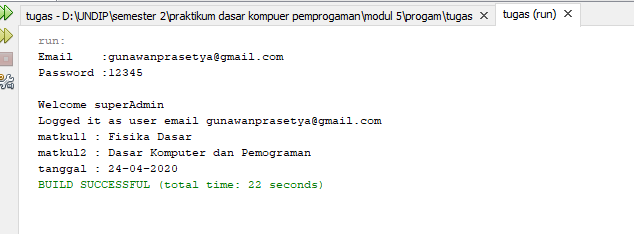
|  |
| --- |
| package tugas;  import java.util.Scanner;  /\*\*  \*  \* @author Lenovo  \*/  public class Tugas {  /\*\*  \* @param args the command line arguments  \*/  public static void main(String[] args) {  Scanner in = new Scanner(System.in);  String email,pass;  System.out.print("Email :");  email = in.nextLine();  System.out.print("Password :");  pass = in.nextLine();  data user = new data (email,pass);  user.login(); |

*Constructor*

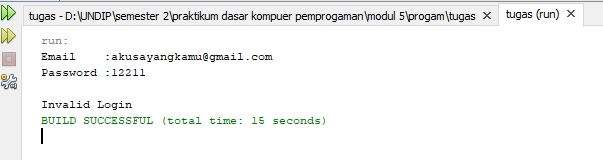
|  |
| --- |
| public class data {  private String[][] data = new String[2][3];  private String[][] historis = new String[2][4];  private String email, password, roles = "";  private String matkul1, matkul2, tanggal ="";    //ini namanya constructor yang akan dijalankan setiap class diinisialisasikan  public data(String emails, String passwords)  {  email = emails;  password = passwords;  String[][] data =  {  {"gunawanprasetya@gmail.com", "12345", "superAdmin"},  {"hanaabdilah@gmail.com", "12345", "user"}  };  String[][] historis =  {  {"gunawanprasetya@gmail.com", "Fisika Dasar", "Dasar Komputer dan Pemograman","24-04-2020"},  {"hanaabdilah@gmail.com", "Dasar Komputer dan Pemograman", "Konsep Jaringan Komputer","24-04-2020"}  };  this.data = data;  this.historis = historis;  }    private boolean checkCredential()  {  for(int i = 0; i < data.length; i++ )  {  if(data[i][0].equals(email))  {  if(data[i][1].equals(password))  {  roles = data[i][2];  matkul1 = historis [i][1];  matkul2 = historis [i][2];  tanggal = historis [i][3];  return true;  }  }  }  return false;  }    public void login()  {  boolean status = checkCredential();  if(status == true)  {  System.out.println("\nWelcome " + roles);  System.out.println("Logged it as user email " + email);  System.out.println("matkul1 : " + matkul1);  System.out.println("matkul2 : " + matkul2);  System.out.println("tanggal : " + tanggal);    }  else  {  System.out.println("\nInvalid Login ");  }  } |



Gambar 6. 20 Output user pada Java



Gambar 6. 21 Output Superadmin pada Java



Gambar 6. 22 Output invalid login pada Java

Program ini memiliki *class* yang dijadikan sebagai *constructor*. Dalam main program *constructor* di panggil dengan *sintax* user.login();yang mana terlebih dahulu memanggil *class*nya data user = new data (email,pass);, *class* yang diberikan parameter dari inputan 1 dan 2. Dalam *class* tardapat kumpulan fungsi, dan juga berisi *array* data dan histori yang di gunakan sebagai database. Dalam *constructor login* terdapat pencocokan terlebih dahulu dari hasil *class boolean* yang mengunakan parameter yang mana telah di masuakan dan di ubah manjadi variabel *mails* dan *passwords*. Variabel ini dicocokaln menggunakan *class* ini yang terdapat perulangan untuk mencocokan dari data *array*. Sehingga didapat mencocokan lebih dari satu data akun didata base, hasil akhir adalah melanjutkan perintah cetakan yang ada dalam *class login.*

## Kesimpulan

1. Construction ditandai dengan adanya method this. Pada C#, java dan PHP. Dan untuk python ditandai dengan self.
2. *Method* *checkCredentials* maksudnya adalah untuk melihat atau mengecek kembali apakah data yang ada sama dengan data yang dimasukkan *user.* Jika tidak sama maka program akan *invalid* atau gagal login.
3. Pada python untuk pembuatan *constructor* ada penambahan \_\_init\_\_. Yang maksudnya pemanggilan *class* saat membuat object baru. Khusus untuk python menggunakan *variabel* self. yang merujuk pada *object* dari *class* tersebut.
4. Untuk membedakan antara *class* dan *object* dilihat jika *class* itu kerangka buat ngebentuk object, misalnya dibuat *class* Mahasiswa maka nanti didalamnya ada object nama, nim, sama hobi. Sedangkan,kalau *object* sendiri itu data nya, contohnya nama, nim, jurusan.
5. Untuk membuat *class* baru cukup menambahkan new pada fungsinya. Dan apabila ingin mengambil data dari *class* itu sendiri maka ada penambahan get.