**BAB VI**

**CLASS DAN CONSTRUCTOR**

### **Tujuan**

1. Untuk memenuhi Tugas Laporan Praktikum Modul 5 Dasar Komputer dan Pemrograman.
2. Mengenal dan mengetahui perbedaan penggunaan beberapa *IDE* atau aplikasi pemrograman dan beberapa jenis bahasa pemrograman.
3. Memahami pengertian dan konsep dari *Class* dan *Constructor*.
4. Mengetahui penggunaan *Class* dan *Constructor* dengan bahasa pemrograman Phyton, C#, Java, dan PHP.
5. Mengetahui perbedaan *Class* dan *Constructor* dalam bahasa pemrograman Phython, C#, Java, dan PHP.
6. Mampu mengimplementasikan *Class* dan *Constructor* dalam bahasa pemrograman Phython, C#, Java, dan PHP.
7. Menjalin silaturahmi dengan kakak tingkat dan teman se-angkatan.

### **Dasar Teori**

* + 1. **Pemrograman Berbasis Objek**

Pemrograman Berbasis Objek (OOP) adalah paradigma pemrograman yang berfokus pada “objek” yang berisi data, atribut, dan code dalam bentuk fungsi yang lebih dikenal dengan nama *method*. *Object oriented programming* (OOP) merupakan konsep pembuatan program dengan memecah permasalahan program menggunakan objek. Objek dapat diartikan dengan fungsi khusus yang dapat berdiri sendiri dan saling bertukar guna mencapai hasil akhir.

Ada beberapa keunggulan yang bisa kita dapatkan saat menggunakan metode OOP ini. Diantaranya:

1. *Improved Software Development Productivity*: Sistem program dapat dimodifikasi tanpa melibatkan banyak modul dimana hanya objek saja yang terlibat. Program dapat dikembangkan sampai skala paling kompleks.
2. *Improved Software Maintainability*: Bagian dari *software* dapat dengan mudah di *maintenance* jika ada perubahan meskipun dalam skala yang cukup besar.
3. *Faster Development*: Metode ini didukung oleh banyak *library* objek, sehingga mempercepat penyelesaian program dan juga projek berikutnya.
4. *Lower Cost of Development*: *Faster development* tentu akan mengurangi biaya pengembangan pembuatan program
5. *Higher Quality Software*: *Faster development* akan memberikan lebih banyak waktu dan sumberdaya untuk proses verifikasi *software*.

Adapun kelemahan dari OOP yaitu:

1. Untuk beberapa programmer butuh waktu untuk terbiasa dengan OOP.
2. Ukuran program yang dibuat dengan metode ini cukup besar.
3. *Runtime* program lebih lambat.
4. Tidak semua masalah program bisa diselesaikan dengan OOP.

Istilah-Istilah Dalam Pemrograman Berorientasi Objek:

1. Enkapsulasi (Pembungkusan): Merupakan pelindung program dan data yang sedang diolah. Enkapsulasi mendefinisikan perilaku melindungi program dan data agar tidak diakses secara sembarangan oleh orang lain.
2. Inheritansi: Objek-objek yang ada disekitar kita adalah objek-objek yang saling terhubung secara hirarki.
3. *Class*: Cetakan dari *object*. Sebuah *class* berisi kode-kode yang menjelaskan sebuah *object* akan berperilaku dan berinteraksi satu sama lain.
4. *Property*: Variabel yang dideklarasikan di dalam sebuah *class*, tetapi tidak berada di dalam fungsi atau *method* dari suatu *class*.
5. *Method*: Fungsi yang ada di dalam *class*. *Method* dapat diakses dengan tiga jenis *user* atau *modifier*. Dalam pemrograman objek *method* dapat menyimpan *state* dalam variabel dan mengimplementasikan *behavior*nya menggunakan *method*.
6. *Object*: Bagian dari sebuah program. Di dalamnya dihubungkan beberapa variabel dan *method* yang saling berkaitan satu sama lain
7. *Overloading*: Pemisalan dalam sebuah *class*.

Sumber: badoystudio.com

* + 1. **Class**

*Class* adalah “Cetak Biru” dari objek. *Class* digunakan hanya untuk membuat kerangka dasar. Yang akan dipakai nantinya adalah hasil dari cetakan dari *class*, yakni objek. *Class* di dalam OOP disa dikatakan sebagai *library*. *Class* berisi *property* dan *method*. Ibaratnya, *class* adalah sebuah wadah menyimpan *property* dan *method*, dan objek yang dihasilkan biasanya berdasarkan isi dari *class*.

Berikut bentuk umum dari penggunaan *class*:

|  |
| --- |
| [modifier] class <nama\_class> {  <class body>  } |

Keterangan :

1. *Modifier,* ini hampir sama seperti enkapsulasi pada sebuah method/function. Namun jarak jangkauan aksesnya berada pada package. Umunya ada 3 jenis Modifier:

* Kosong / *default* / *not specified*: Dapat diakses oleh class lain tapi dalam satu *package*.
* *Public*: Dapat dipakai di *class* manapun atau *package* lain.
* *Private*: Tidak bisa dipakai oleh oleh *class* manapun.

1. <*class\_name*>, bisa diganti dengan nama class yang diinginkan.
2. <*class\_body*>, ini biasanya berisi variabel, *method, function, constructor* maupun data dari *class* lain.

Adapun cara untuk mengakses *class* di *class* yang lain (*object*) dapat menggunakan format berikut:

|  |
| --- |
| <nama\_kelas> <nama\_objek> = new <nama\_kelas>(<parameter>); |

Keterangan:

1. <nama\_kelas> ini merupakan nama dari *class* lain yang mempunyai m*odifier public* maupun *default*.
2. <nama\_objek> ini sebenarnya seperti variabel bebas yang memakili *class* yang akan dipakai.
3. <parameter> yang ada dalam *new* <nama\_class> ini bersifat opsional. Dan parameter ini nanti akan berhubungan dengan sebuah *contructor*.

Sumber: malasngoding.com, duniailkom.com, Modul 5

* + 1. **Constructor**

*Constructor* adalah *method* khusus yang akan dijalankan secara otomatis pada saat sebuah objek dibuat (instansiasi), yakni ketika perintah “*new*” dijalankan. *Constructor* bisa disebut sebagi *method* yang namanya sama dengan nama *class*nya.

*Constructor* biasa digunakan untuk membuat proses awal dalam mempersiapkan objek, seperti memberi nilai awal kepada *property*, memanggil *method* internal dan beberapa proses lain yang digunakan untuk mempersiapkan objek. *Constructor* bisa memiliki parameter bisa juga tidak. Dalam *Constructor*, tidak boleh pakai *keyword void*.

Cara panggil *constructor* adalah dengan menambah *keyword* “*new*”. *Keyword new* dalam deklarasi ini artinya kita mengalokasikan pada *memory* sekian blok *memory* untuk menampung objek yang baru kita buat.

|  |
| --- |
| <modifier> <nama\_class> (<parameter>) {  <body\_constructor>  } |

Keterangan :

1. <*modifier*> ini bisa menggunakan salah satu dari 3 *modifier* yang ada.
2. <nama\_*class*> merupakan bagian nama sebuah *class* dipakai karena nanti saat dipanggil dari *class* lain akan langsung otomatis dijalankan.
3. <parameter> disini menyesuaikan dari *object class* tersebut dipakai oleh di *class* lain.

Sumber: jagokoding.com, duniailkom.com, Modul 5

### **Hasil Percobaan dan Analisis**

* + 1. **Class pada PHP**

**Index.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 Input Email dan password  <form action="./server.php" method="POST">  <label >System Login Kelompok 49</label><br>  <input type="email" name="email" placeholder="email">  <input type="password" name="password" placeholder="password">  <button type="submit">Submit</button>  </form>  </body> #End Input Email dan Password  </html> |

**Server.php**

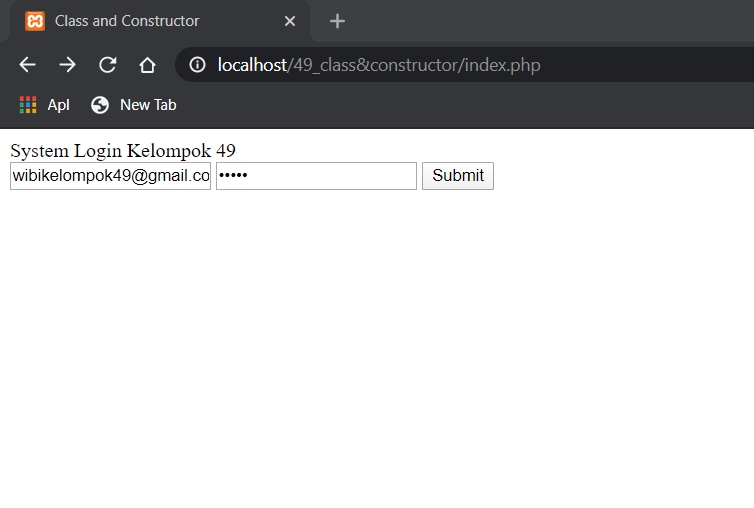
|  |
| --- |
| <?php  include("userService.php"); #pemrosesan inputan pada userService  $user = new userService($\_POST['email'], $\_POST['password']);  if($get\_user = $user->login()) { # output Bila email,pass benar  echo 'Welcome '.$user→getRole();  echo ', Logged it as user email:'.$get\_user;  } else { #output bila email,pass salah  echo 'Invalid Login';  }  ?> |

**UserService.php**

|  |
| --- |
| <?php  class userService  {  #Start Deklarasi Variabel yang digunakan  protected $email; // using protected so they can be accessed  protected $password; // and overidden if necessary  protected $dataUser; // dummy data  protected $getRole; // stores the role data  #End Deklarasi Variabel yang digunakan  #Method inisialisasi,  public function \_\_construct($email, $password)  {  ##Deklarasi email,pass,dan role yang berpasangan  $this->\_dataUser = [  (object) [  'email' => "aldikelompok49@gmail.com",  'password' => "12345",  'role' => "superadmin"  ],  (object) [  'email' => "wibikelompok49@gmail.com",  'password' => "12345",  'role' => "user"  ]  ];  $this->email = $email; #Pengambilan email  $this->password = $password; #Pengambilan password  }  public function login() #Pengecekan kebenaran email,pass  {  $user = $this->checkCredentials();  if($user) {  $this->getRole = $user->role;  return $get\_data = $user->email;  } else {  return false;  }  }  protected function checkCredentials()  {  foreach($this->\_dataUser as $key => $value) {  if($value->email == $this->email && $value->password == $this->password) {  return $value;  }  }  return false;  } #End Pengecekan kebenaran email,pass  public function getRole() #Pengembalian role  {  return $this->getRole;  }  }  ?> |

Pertama tama kita buat 3 *file* yaitu *index, server,* dan *UserService* . disini kita membuat *file userService* untuk pengecekan apakah pengguna memasukkan *data* yang benar atau salah. Hal ini kita lakukan dengan membuat kelas setelah itu pendefinisian variabel variabel yang kita gunakan, setelah itu kita buat 3 *method* untuk inisialisasi, Pengecekan *data* dan mengembalikan nilai. Lalu pada *file server* kita gunakan sebagai halaman lanjutan bila pengguna telah memasukkan *data*, disini keluaran dari *file server* bergantung dari pengecekan *data* di *file userService*. Lalu pada *File index* kita memberikan *form* supaya pengguna dapat memasukkan *data* dan selanjutnya akan keluar halaman yang memuat *file server* bila *data* telah dimasukkan dan tombol telah dipencet sehingga hasil akhir akan dikeluarkan.

*Output* Hasil:



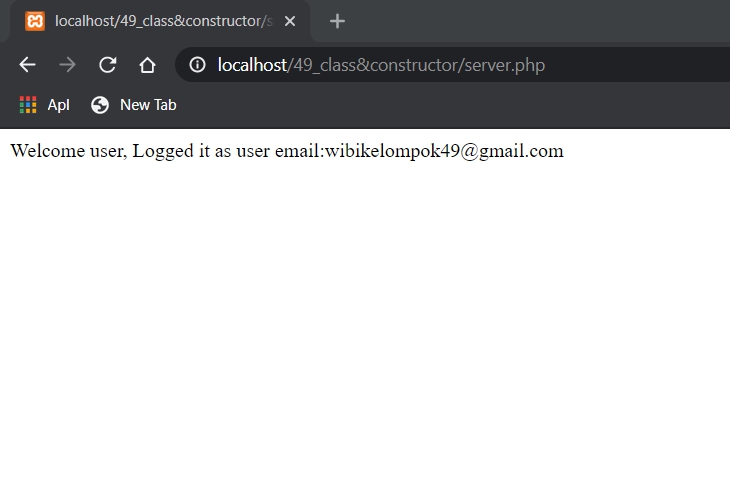
Gambar 6. 1. Output Class dalam PHP

* + 1. **Constructor pada PHP**

|  |
| --- |
| public function \_\_construct($email, $password)  {  #Deklarasi email,pass,dan role yang benar  $this->\_dataUser = [  (object) [  'email' => "aldikelompok49@gmail.com",  'password' => "12345",  'role' => "superadmin"  ],  (object) [  'email' => "wibikelompok49@gmail.com",  'password' => "12345",  'role' => "user"  ]  ];  $this->email = $email; #Pengambilan email  $this->password = $password; #Pengambilan Password  } |

Pada *constructor* kita menjelaskan variabel variabel yang akan dipakai, disini kita akan memakai variabel *dataUser* untuk pengumpulan *data* yang tepat serta variabel *email* dan *password* untuk mengambil *data* dari luar.

*Output* Hasil:

****

Gambar 6. 2. Output Constructor dalam PHP

* + 1. **Class pada Phyton**

**49\_ClassConstructor.py**

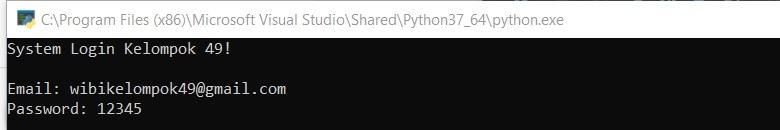
|  |
| --- |
| from userService import userService  #input email pass  print("System Login Kelompok 49!\n")  email = input("Email: ")  password = input("Password: ")  #Pemanggilan userService  get\_class = userService(email,password)  get\_class.login() |

**userService.py**

|  |
| --- |
| class userService(object):  """description of class"""  def \_\_init\_\_(self, email, password): #Inisialisasi  self.email = email #Pemanggilan email  self.password = password #Pemanggilan Pass  #email,password,dan role yang benar  self.data = {  "aldikelompok49@gmail.com" : {  "email" : "aldikelompok49@gmail.com",  "password" : "12345",  "role" : "superadmin"  },  "wibikelompok49@gmail.com" : {  "email" : "wibikelompok49@gmail.com",  "password" : "12345",  "role" : "user"  }  }  def login(self): #output setelah pengecekan  get\_data = self.checkCredentials()  if get\_data: #bila pengecekan benar  print("\nWelcome ",get\_data['role'])  print("Logged it as user email: ",get\_data['email'])  else: #bila pengecekan salah  print("\nInvalid Login!\n")  def checkCredentials(self): #pengecekan email dan pass  for value in self.data:  if value == self.email:  get\_data\_user = self.data[value]  if self.password == get\_data\_user['password']:  return get\_data\_user  else:  return False |

Pertama tama kita buat 2 *file* yaitu *classConstructor* dan *UserService*. Disini kita membuat *file userService* untuk pengecekan apakah pengguna memasukkan *data* yang benar atau salah. Hal ini kita lakukan dengan membuat kelas setelah itu pendefinisian variabel variabel yang kita gunakan, setelah itu kita buat 3 *method* untuk inisialisasi, Pengecekan *data* dan pengeluaran hasil. Lalu pada *classConstructor*  kita memberikan *inputan* supaya pengguna dapat memasukkan *data* dan selanjutnya akan lanjut ke *file userService*  untuk pengecekan apakah *data* yang dimasukkan benar atau salah dan hasil keluaran.

*Output* Hasil:

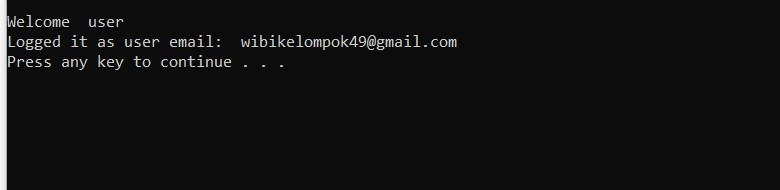
****

Gambar 6. 3. Output Class dalam Phyton

* + 1. **Constructor pada Phyton**

|  |
| --- |
| def \_\_init\_\_(self, email, password): #Inisialisasi  self.email = email #Pemanggilan email  self.password = password #Pemanggilan Pass  #email,password,dan role yang benar  self.data = {  "aldikelompok49@gmail.com" : {  "email" : "aldikelompok49@gmail.com",  "password" : "12345",  "role" : "superadmin"  },  "wibikelompok49@gmail.com" : {  "email" : "wibikelompok49@gmail.com",  "password" : "12345",  "role" : "user"  }  } |

Pada *constructor* kita menjelaskan variabel variabel yang akan dipakai, disini kita akan memakai variabel *data* untuk pengumpulan *data* yang tepat serta variabel *email* dan *password* untuk mengambil *data* dari luar.

*Output* Hasil:

Gambar 6. 4. Output Constructor dalam Phyton

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

**Program.cs**

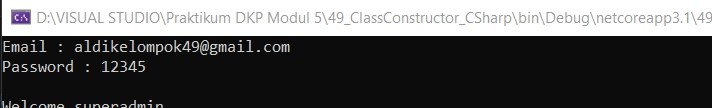
|  |
| --- |
| using System;  namespace \_49\_ClassConstructor\_CSharp  {  class Program  {  static void Main(string[] args)  {  string email, password; #Deklarasi email,pass  //inputan  Console.Write("Email : ");  email = Console.ReadLine();  Console.Write("Password : ");  password = Console.ReadLine();  //Pemanggilan userService  userService data = new userService(email, password);  data.login();  Console.ReadKey();  }  }  } |

**userService.cs**

|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Text;  namespace \_49\_ClassConstructor\_CSharp  {  class userService  {  //deklarasi variabel  private string[,] data;  private string email, password, roles = "";  public userService(string emails, string passwords)  {  email = emails; //pengambilan email  password = passwords; //pengambilan password  //email,password,dan role yang benar  data = new string[2, 3] {  {"aldikelompok49@gmail.com", "12345", "superadmin" },  {"wibikelompok49@gmail.com", "12345", "user" }  };  }  public void login() //output hasil pengecekan  {  var (status, role) = checkCredentials();  if (status == true) //bila pengecekan benar  {  Console.WriteLine("\nWelcome " + role);  Console.WriteLine("Logged it as user email: " + email);  }  else //bila pengecekan salah  {  Console.WriteLine("\nInvalid Login");  }  }  private (bool, string) checkCredentials() //Pengecekan  {  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);  }  }  } |

Pertama tama kita buat 2 *file* yaitu *Program* dan *UserService* . disini kita membuat *file userService* untuk pengecekan apakah pengguna memasukkan *data* yang benar atau salah. Hal ini kita lakukan dengan membuat kelas setelah itu pendefinisian variabel variabel yang kita gunakan, setelah itu kita buat 3 *method* untuk inisialisasi, Pengecekan *data* dan pengeluaran hasil. Lalu pada *Program*  kita memberikan *inputan* supaya pengguna dapat memasukkan *data* dan selanjutnya akan lanjut ke *file userService*  untuk pengecekan apakah *data* yang dimasukkan benar atau salah dan hasil keluaran.

*Output* Hasil:

****

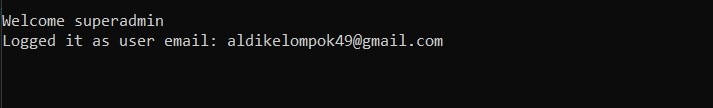
Gambar 6. 5. Output Class dalam C#

* + 1. **Constructor dalam C#**

|  |
| --- |
| class userService  {  //deklarasi variabel  private string[,] data;  private string email, password, roles = "";  public userService(string emails, string passwords)  {  email = emails; //pengambilan email  password = passwords; //pengambilan password  //email,password,dan role yang benar  data = new string[2, 3] {  {"aldikelompok49@gmail.com", "12345", "superadmin" },  {"wibikelompok49@gmail.com", "12345", "user" }  };  } |

Pada *constructor* kita menjelaskan variabel variabel yang akan dipakai, disini kita akan memakai variabel *data* untuk pengumpulan *data* yang tepat serta variabel *email* dan *password* untuk mengambil *data* dari luar serta variabel *roles* untuk pengeluaran variabel *roles*.

*Output* Hasil:



Gambar 6. 6. Output Constructor dalam C#

* + 1. **Class pada Java**

**Main.java**

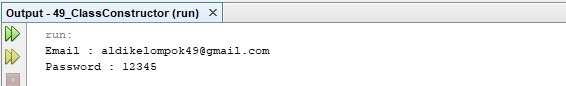
|  |
| --- |
| package pkg49\_classconstructor;  import java.util.Scanner; //Import code untuk input  public class Main {  public static void main(String[] args) {  String email, password; //deklarasi variabel    //Input email,dan password  Scanner input = new Scanner(System.in);  System.out.print("Email : ");  email = input.nextLine();  System.out.print("Password : ");  password = input.nextLine();    //Pemanggilan userService  userService satu = new userService( email, password );  satu.login();  }    } |

**userService.java**

|  |
| --- |
| package pkg49\_classconstructor;  class userService {  private String[][] data = new String[2][3];  private String email, password, roles = ""; //deklarasi    //ini namanya constructor yang akan dijalankan setiap class diinisialisasikan  public userService(String emails, String passwords)  {  email = emails; //pengambilan email  password = passwords; //pemanggilan password  //email,password,dan role yang benar  String[][] data =  {  {"aldikelompok49@gmail.com", "12345", "supedAdmin"},  {"wibikelompok49@gmail.com", "12345", "user"}  };  this.data = data;  }    private boolean checkCredential() //Pengecekan email,pass  {  for(int i = 0; i < data.length; i++ )  {  if(data[i][0].equals(email))  {  if(data[i][1].equals(password))  {  roles = data[i][2];  return true;  }  }  }  return false;  }    public void login() //Pengeluaran hasil pengecekan  {  boolean status = checkCredential();  if(status == true) //bila pengecekan benar  {  System.out.println("\nWelcome " + roles);  System.out.println("Logged it as user email " + email);  }  else //Bila pengecekan salah  {  System.out.println("\nInvalid Login ");  }  }  } |

Pertama tama kita buat 2 *file* yaitu *Main* dan *UserService* . disini kita membuat *file userService* untuk pengecekan apakah pengguna memasukkan *data* yang benar atau salah. Hal ini kita lakukan dengan membuat kelas setelah itu pendefinisian variabel variabel yang kita gunakan, setelah itu kita buat 3 *method* untuk inisialisasi, Pengecekan *data* dan pengeluaran hasil. Lalu pada *Main*  kita memberikan *inputan* supaya pengguna dapat memasukkan *data* dan selanjutnya akan lanjut ke *file userService*  untuk pengecekan apakah *data* yang dimasukkan benar atau salah dan hasil keluaran.

*Output* Hasil:

****

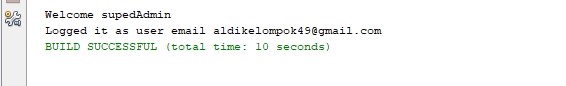
Gambar 6. 7. Output Class dalam Java

* + 1. **Constructor pada Java**

|  |
| --- |
| private String email, password, roles = ""; //deklarasi    //ini namanya constructor yang akan dijalankan setiap class diinisialisasikan  public userService(String emails, String passwords)  {  email = emails; //pengambilan email  password = passwords; //pemanggilan password  //email,password,dan role yang benar  String[][] data =  {  {"aldikelompok49@gmail.com", "12345", "supedAdmin"},  {"wibikelompok49@gmail.com", "12345", "user"}  };  this.data = data;  } |

Pada *constructor* kita menjelaskan variabel variabel yang akan dipakai, disini kita akan memakai variabel *data* untuk pengumpulan *data* yang tepat serta variabel *email* dan *password* untuk mengambil *data* dari luar serta variabel *roles* untuk pengeluaran variabel *roles*.

*Output* Hasil:

**

Gambar 6. 8. Output Constructor dalam Java

### **Tugas**

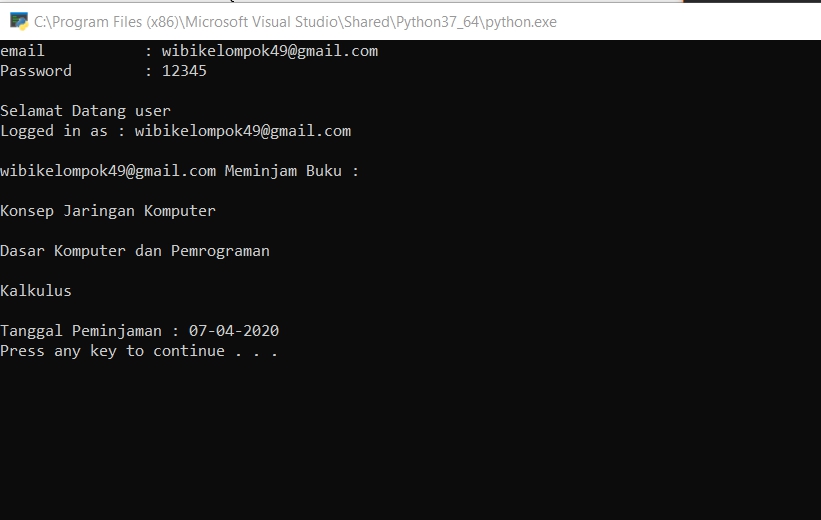
**Tugas\_Modul5\_Kelompok49.py**

|  |
| --- |
| import data\_user #Pengambilan data\_user  #input  email = input("email\t\t: ")  password = input("Password\t: ")  print()  #Pemanggilan data  login = data\_user.data(email,password)  login.out\_login() |

**data\_user.py**

|  |
| --- |
| class data :  def \_\_init\_\_ (self,email,password): #inisialisasi  self.email = email #Pemanggilan email  self.password = password #Pemanggilan password  #Email,Password,dan role yang benar  self.data = {  "aldikelompok49@gmail.com" : {  "email" : "aldikelompok49@gmail.com",  "password" : "12345",  "role" : "superadmin"  },  "wibikelompok49@gmail.com" : {  "email" : "wibikelompok49@gmail.com",  "password" : "12345",  "role" : "user"  }  }  #Riwayat Peminjaman  self.history = {  "aldikelompok49@gmail.com":{  "peminjaman\_buku":{"Fisika Dasar","Dasar Komputer dan Pemrograman"},  "tanggal\_peminjaman" : "10-04-2020"  },  "wibikelompok49@gmail.com":{  "peminjaman\_buku":{"Kalkulus","Dasar Komputer dan Pemrograman","Konsep Jaringan Komputer"},  "tanggal\_peminjaman" : "07-04-2020"  }  }    def cek\_login (self) : #Pengecekan email,password  for value in self.data:  if value == self.email:  call\_data = self.data[value]  if self.password == call\_data["password"]:  return self.email  else :  return False  def out\_login (self) : #Hasil Pengecekan email password  a = self.cek\_login()  if a: #bila pengecekan benar  login = self.data[a]  history = self.history[a]  if login:  print(f'Selamat Datang {login["role"]}')  print(f'Logged in as : {login["email"]}')  print()  print(f'{login["email"]} Meminjam Buku :')  print()  for x in history["peminjaman\_buku"]:  print(x)  print()  print(f'Tanggal Peminjaman : {history["tanggal\_peminjaman"]}')  else : #bila pengecekan salah  print("Invalid Login") |

*Output* Hasil:



Penjelasan *Output* :

Pertama tama kita buat 2 *file*  yaitu Tugas\_Modul5\_Kelompok49 dan *data\_user*. Pada *file* *data\_user* kita membuat kelas dengan 3 buah *method* didalamnya. *Method* init berfungsi sebagai inisialisasi dan penyebutan variabel variabel yang digunakan. disini kami menggunakan variabel *email* dan *password* untuk menerima masukan dari luar sehingga dapat diolah, *data* berfungsi sebagai penampung *email,password,* dan *role* yang benar serta *history* yang berfunsi sebagai *history* peminjaman yang dilakukan oleh *user*. Selanjutnya pada *method* *cek\_login* berfungsi sebagai pengecekan apakah *email* dan *password* cocok atau tidak, jika tidak cocok maka akan mengembalikan nilai false. Selanjutnya pada *method out\_login* akan mengambil nilai dari *cek­­\_login* dan mengeluarkan hasil program. Selanjutnya pada Tugas\_Modul5\_Kelompok49 kita menyediakan *input* supaya *user* dapat memasukkan *email* dan *password* dan selanjutnya *input* akan diolah di *data­\_user*.

Link Tugas : <https://github.com/SMAMHTN/Praktikum_DKP_Kel49/tree/master/Tugas%20Modul%205%20Kelompok%2049/Tugas_Modul5_Kelompok49>

### **Kesimpulan**

* + 1. *Class* dan *Constructor* memiliki hubungan yang saling berkaitan.
    2. Penggunaan *Class* dan *Constructor* betujuan agar menghemat ukuran program, dapat dipakai berkali-kali dan memudahkan dalam mengembangkannya.
    3. Penggunaan *Class* atau *Constructor* dipilih berdasarkan kebutuhan dan urgensinya masing-masing.
    4. Bahasa pemrograman yang keluar dalam waktu dekat, akan memiliki kemampuan yang lebih cepat dan *simple* dari pendahulunya. Seperti phyton lebih *simple* dari C#.
    5. Pemograman mempunyai tiga inti dasar yaitu input, proses, dan output.
    6. Antara bahasa pemrograman satu dengan yang lainnya kadang terdapat perbedaan-perbendaan tertentu, karena memiliki aturannya masing-masing.
    7. Bahasa pemrogaman tidak dapat dibanding-bandingkan, setiap bahasa pemograman bisa dipakai sesuai kebutuhannya masing-masing.
    8. Diperlukan ketelitian dalam mebuat sebuah program, karena kesalahan sedikit akan membuat program tidak berjalan dengan baik.