**BAB VII**

**UDT, SETTER DAN GETTER**

### **Tujuan**

1. Mengenal dan mengetahui perbedaan penggunaan beberapa *IDE* atau aplikasi pemrograman dan beberapa jenis bahasa pemrograman.
2. Memahami pengertian dan konsep dari *UDT, Setter* dan *Getter*.
3. Mengetahui penggunaan *UDT, Setter* dan *Getter* dengan bahasa pemrograman Python, C#, Java, dan PHP.
4. Mengetahui perbedaan *UDT, Setter* dan *Getter* dalam bahasa pemrograman Python, C#, Java, dan PHP.
5. Mampu mengimplementasikan *UDT, Setter* dan *Getter* dalam bahasa pemrograman Python, C#, Java, dan PHP.

### **Dasar Teori**

* + 1. **Enkapsulasi**

Enkapsulasi adalah metode untuk mengatur struktur pada *class* dengan tujuan untuk menentukan hak akses pada *property* atau *method*. Hal ini bertujuan untuk menyembunyikan informasi dari *method* dan *property* dengan alasan keamanan. Enkapsulasi bisa juga diartikan sebagai pembungkusan data dalam satu buah unit. Cara lain untuk berpikir mengenai enkapsulasi adalah upaya untuk melindungi (perisai proteksi) yang mencegah data untuk diakses oleh pihak luar.

Keunggulan dari enkapsulasi adalah sebagai berikut:

1. Data tersembunyi (*data hiding*): pengguna tidak mengetahui implementasi dasar dari *class*. Hal yang diketahui oleh *user* adalah nilai telah diatur melalui *setter method* dan variabel telah diinisialisasi dengan nilai tertentu.
2. Meningkatkan fleksibilitas: *programmer* dapat membuat variabel dari *class* sebagai *read-only* atau *write-only* tergantung pada kebutuhan.
3. *Reusability*: enkapsulasi meningkatkan *reusability* dan kemudahan perubahan menggunakan *new requirement*.
4. Kemudahan pengujian kode program: enkapsulasi kode memudahkan proses pengetesan program untuk unit-unit programnya masing-masing.

Untuk memberikan hak akses pada *method* dan *property* pada sebuah *class*, terdapat 3 teknik, yaitu:

* + - 1. Hak Akses *Public*

Jika sebuah *method* dan *property* menggunakan *public*, maka *method* dan *property* tersebut dapat diakses dari luar ataupun dari dalam *class* tersebut. Jika tidak ditetapkan hak aksesnya, maka hak akses pada *method* atau *property* tersebut secara *default* menjadi *public*. Contoh penerapannya:

|  |
| --- |
| <?php  //class manusia  class manusia{  //prop  public $nama;  public $warna;    //method manusia  function tampilkan\_nama(){  return "Nama saya malasngoding <br/>";  }    }  //instansiasi class manusia  $manusia = new manusia();  //memanggil method tampilkan\_nama dari class manusia  echo $manusia->tampilkan\_nama(); |

* + - 1. Hak Akses *Private*

*Private* adalah hak akses yang melarang *method* atau *property* yang menggunakannya dilarang diakses dari luar *class*. Jadi hak akses hanya bisa diakses dari dalam *class* itu sendiri. Contoh penerapannya:

|  |
| --- |
| <?php  //class manusia  class manusia{  //menentukan property dengan private  private $nama = "malas ngoding";    //method public  public function tampilkan\_nama(){  return "Nama saya " .$this->nama;  }      }  //instansiasi class manusia  $manusia = new manusia();  //memanggil method public tampilkan\_nama dari class manusia  echo $manusia->tampilkan\_nama(); |

* + - 1. Hak Akses *Protected*

Jika *property* dan *method* ditetapkan sebagai *protected*, berarti *property* dan *method* tersebut tidak bisa diakses dari luar *class*, tetapi bisa diakses dari dalam *class* itu sendiri. Contoh penerapannya:

|  |
| --- |
| <?php  //class manusia  class manusia{  //menentukan property dengan protected  protected $nama = "malas ngoding";    //method protected  protected function nama(){  return "Nama saya " .$this->nama;  }    public function tampilkan\_nama(){  return $this->tampilkan\_nama;  }    }  //instansiasi class manusia  $manusia = new manusia();  //memanggil method public tampilkan\_nama dari class manusia  echo $manusia->tampilkan\_nama(); |

*(Sumber: https://www.malasngoding.com/php-oop-part-4-pengertian-enkapsulasi-public-private-protected/)*

* + 1. **User Data Type (UDT)**

*User-Defined Type* adalah sebuah tipe data yang terdiri dari struktur-struktur data dan membentuk sebuah tipe data tersendiri sesuai dengan yang didefinisikan, Misalkan akan membuat sebuah tipe data Identitas *Customer*, yang mana data tersebut terdiri dari Kode *Customer*, Nama *Customer*, Alamat dan PlafonKredit dengan menggunakan nama tipe udtIdentitas*Customer*.

*(Sumber: https://www.kodiing.com/2015/06/user-defined-type-dan-enumerasi-vb6.html)*

* + 1. **Setter dan Getter**

*Setter* adalah sebuah aksi saat kita memasukkan sebuah nilai/*values* ke dalam suatu variable/*object*, sedangkan *Getter* adalah sebuah aksi saat kita mengambil sebuah nilai/*values* dari suatu variabel/*object*.

Sebelum menggunakan *Setter* dan *Getter* kita harus membuat sebuah *class/object* terlebih dahulu sehingga kita bisa mengimplementasikan *setter* dan *getter*. Contohnya penerapannya:

|  |
| --- |
| package setterandgetter.domain;  public class Student {  private String nim;  private String name;  private String address;  private String phoneNo;    public void setNim(String nim){  this.nim = nim;  }    public String getNim(){  return nim;  }    public void setName(String name){  this.name = name;  }    public String getName(){  return name;  }    public void setAddress(String address){  this.address = address;  }    public String getAddress(){  return address;  }    public void setPhoneNo(String phoneNo){  this.phoneNo = phoneNo;  }    public String getPhoneNo(){  return phoneNo;  }   } |

*(Sumber: https://medium.com/java-and-other-java-framework/setter-dan-getter-pada-java-46820df13b3)*

### **Hasil Percobaan dan Analisis**

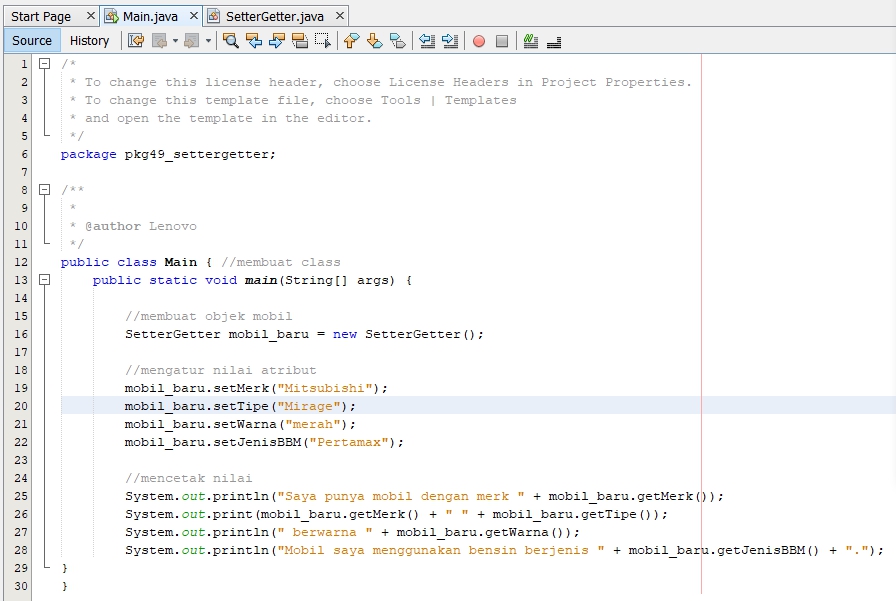
* + 1. **Bahasa Pemrograman Java**

Main.java

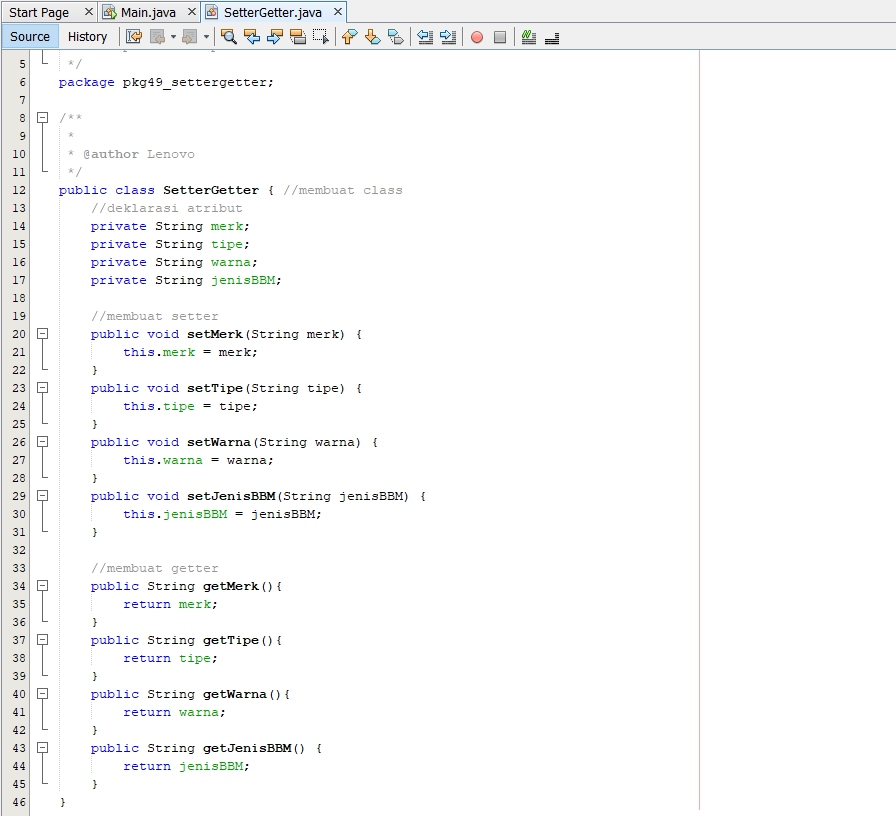
|  |
| --- |
| package pkg49\_settergetter;  /\*\*  \*  \* @author Lenovo  \*/  public class Main { //membuat class  public static void main(String[] args) {    //membuat objek mobil  SetterGetter mobil\_baru = new SetterGetter();    //mengatur nilai atribut  mobil\_baru.setMerk("Mitsubishi");  mobil\_baru.setTipe("Mirage");  mobil\_baru.setWarna("merah");  mobil\_baru.setJenisBBM("Pertamax");    //mencetak nilai  System.out.println("Saya punya mobil dengan merk " + mobil\_baru.getMerk());  System.out.print(mobil\_baru.getMerk() + " " + mobil\_baru.getTipe());  System.out.println(" berwarna " + mobil\_baru.getWarna());  System.out.println("Mobil saya menggunakan bensin berjenis " + mobil\_baru.getJenisBBM() + ".");  }  } |

SetterGetter.php

|  |
| --- |
| package pkg49\_settergetter;  /\*\*  \*  \* @author Lenovo  \*/  public class SetterGetter { //membuat class  //deklarasi atribut  private String merk;  private String tipe;  private String warna;  private String jenisBBM;  //membuat setter  public void setMerk(String merk) {  this.merk = merk;  }  public void setTipe(String tipe) {  this.tipe = tipe;  }  public void setWarna(String warna) {  this.warna = warna;  }  public void setJenisBBM(String jenisBBM) {  this.jenisBBM = jenisBBM;  }    //membuat getter  public String getMerk(){  return merk;  }  public String getTipe(){  return tipe;  }  public String getWarna(){  return warna;  }  public String getJenisBBM() {  return jenisBBM;  }  } |



Gambar 7. 1. Source Code Setter Getter dalam Java (1)



Gambar 7. 2. Source Code Setter Getter dalam Java (2)

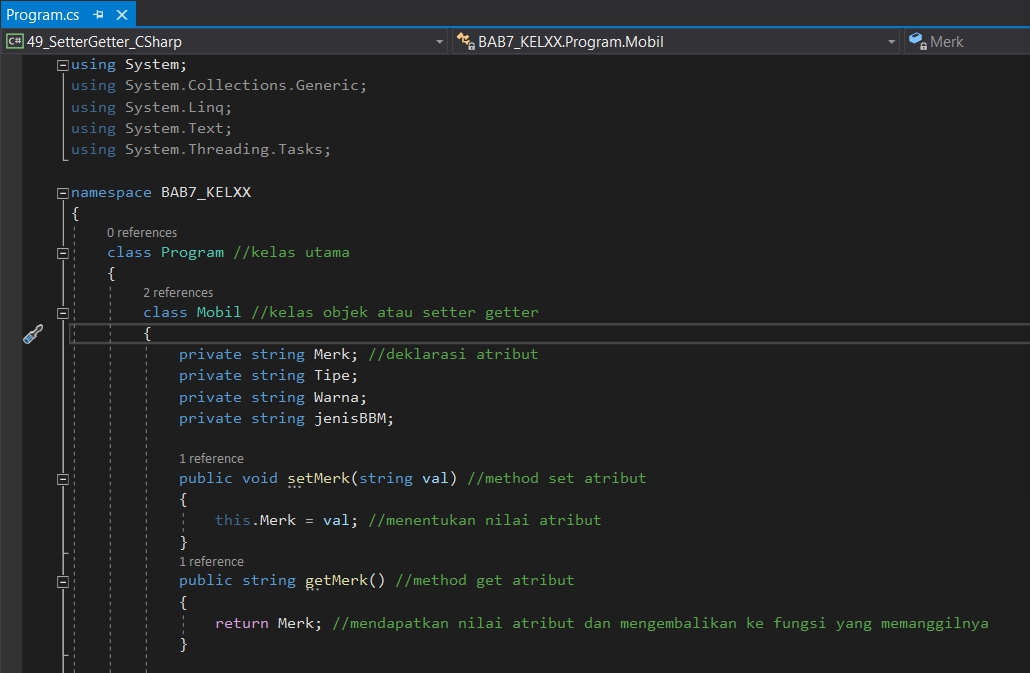


Gambar 7. 3. Output Setter Getter dalam Java

Setelah kita membuat *file* Main.java, Pertama-tama kita buat *file* dengan nama SetterGetter.java dan buat kelas SetterGetter lalu kita definisikan variabel-variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* maka jenis *method* ini adalah *void* karena fungsinya hanya menyimpan nilai yang masuk dan disimpan ke *directory* sementara dan jangan lupa kita buat variabel yang perlu dimasukkan supaya nilai dapat tersimpan. Setelah itu pada *file* Main.java dengan kelas Main. Kita panggil kelas SetterGetter lalu kita tetapkan variabel-variabel untuk disimpan ke kelas SetterGetter dengan menggunakan *method setter* yang telah kita buat pada kelas SetterGetter. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas SetterGetter dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

* + 1. **Bahasa Pemrograman C#**

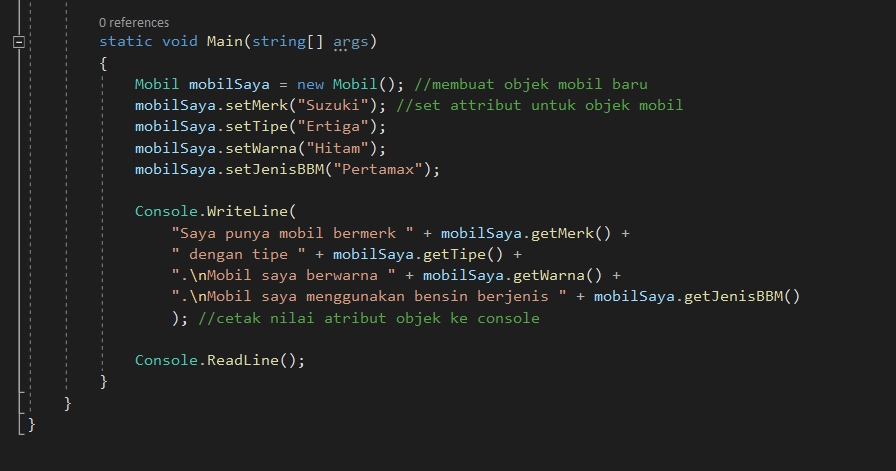
|  |
| --- |
| using System;  using System.Collections.Generic;  using System.Linq;  using System.Text;  using System.Threading.Tasks;  namespace BAB7\_KEL49  {  class Program //kelas utama  {  class Mobil //kelas objek atau setter getter  {  private string Merk; //deklarasi atribut  private string Tipe;  private string Warna;  private string jenisBBM;  public void setMerk(string val) //method set atribut  {  this.Merk = val; //menentukan nilai atribut  }  public string getMerk() //method get atribut  {  return Merk; //mendapatkan nilai atribut dan mengembalikan ke fungsi yang memanggilnya  }  public void setTipe(string val)  {  this.Tipe = val;  }  public string getTipe()  {  return Tipe;  }  public void setWarna(string val)  {  this.Warna = val;  }  public string getWarna()  {  return Warna;  }  public void setJenisBBM(string val)  {  this.jenisBBM = val;  }  public string getJenisBBM()  {  return jenisBBM;  }  }  static void Main(string[] args)  {  Mobil mobilSaya = new Mobil(); //membuat objek mobil baru  mobilSaya.setMerk("Suzuki"); //set attribut untuk objek mobil  mobilSaya.setTipe("Ertiga");  mobilSaya.setWarna("Hitam");  mobilSaya.setJenisBBM("Pertamax");  Console.WriteLine(  "Saya punya mobil bermerk " + mobilSaya.getMerk() +  " dengan tipe " + mobilSaya.getTipe() +  ".\nMobil saya berwarna " + mobilSaya.getWarna() +  ".\nMobil saya menggunakan bensin berjenis " + mobilSaya.getJenisBBM()  ); //cetak nilai atribut objek ke console  Console.ReadLine();  }  }  } |



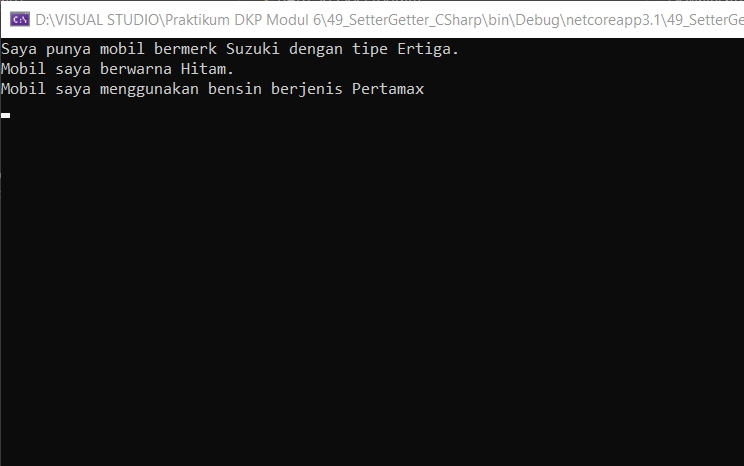
Gambar 7. 4. Source Code Setter Getter dalam C# (1)



Gambar 7. 5. Source Code Setter Getter dalam C# (2)



Gambar 7. 6. Source Code Setter Getter dalam C# (3)

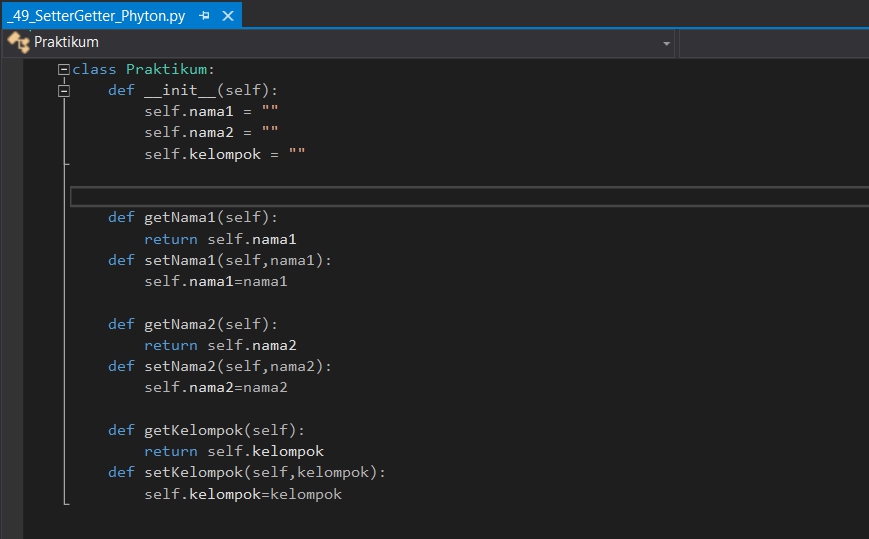
****

Gambar 7. 7. Output Setter Getter dalam C#

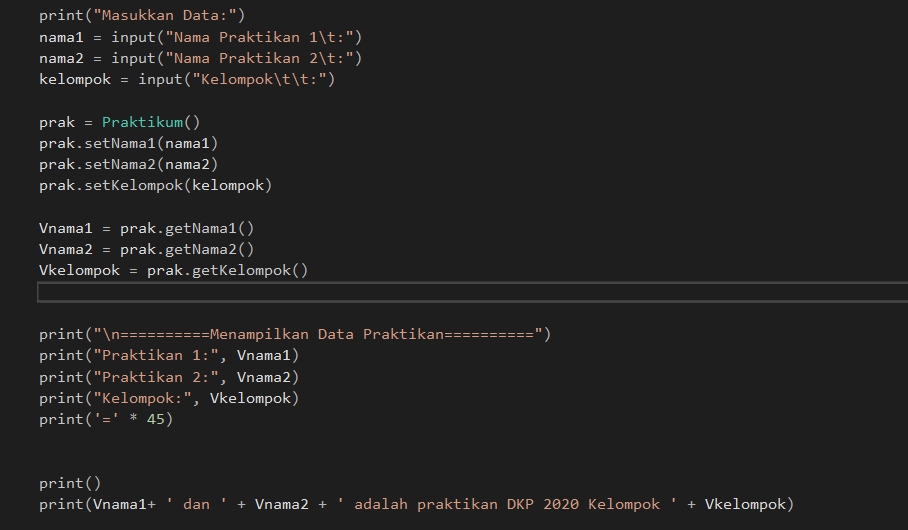
Setelah kita membuat *file* Program.cs, Pertama-tama kita buat kelas dengan nama mobil lalu kita definisikan variabel variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* maka jenis *method* ini adalah *void* karena fungsinya hanya menyimpan nilai yang masuk dan disimpan ke *directory* sementara dan jangan lupa kita buat variabel yang perlu dimasukkan supaya nilai dapat tersimpan. Setelah itu pada kelas utama kita panggil kelas mobil lalu kita tetapkan variabel-variabel untuk disimpan ke kelas mobil dengan menggunakan *method setter* yang telah kita buat pada kelas mobil. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas mobil dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

* + 1. **Bahasa Pemrograman Python**

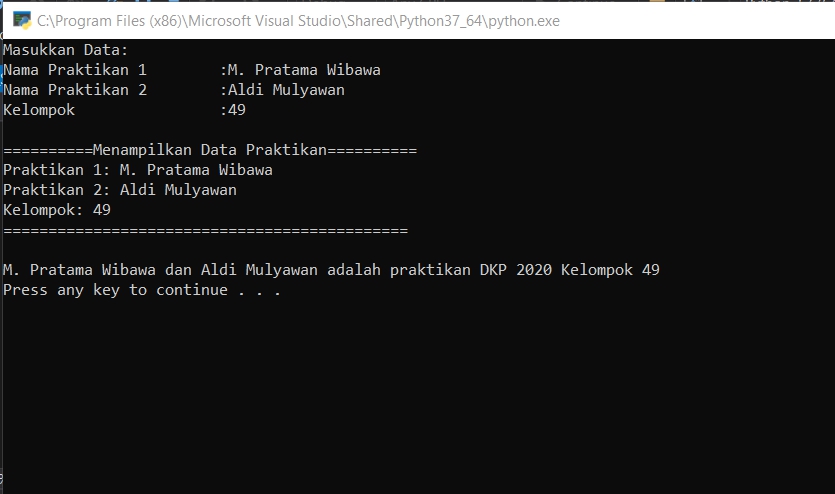
|  |
| --- |
| class Praktikum:  def \_\_init\_\_(self):  self.nama1 = ""  self.nama2 = ""  self.kelompok = ""  def getNama1(self):  return self.nama1  def setNama1(self,nama1):  self.nama1=nama1  def getNama2(self):  return self.nama2  def setNama2(self,nama2):  self.nama2=nama2  def getKelompok(self):  return self.kelompok  def setKelompok(self,kelompok):  self.kelompok=kelompok  print("Masukkan Data:")  nama1 = input("Nama Praktikan 1\t:")  nama2 = input("Nama Praktikan 2\t:")  kelompok = input("Kelompok\t\t:")  prak = Praktikum()  prak.setNama1(nama1)  prak.setNama2(nama2)  prak.setKelompok(kelompok)  Vnama1 = prak.getNama1()  Vnama2 = prak.getNama2()  Vkelompok = prak.getKelompok()    print("\n==========Menampilkan Data Praktikan==========")  print("Praktikan 1:", Vnama1)  print("Praktikan 2:", Vnama2)  print("Kelompok:", Vkelompok)  print('=' \* 45)  print()  print(Vnama1+ ' dan ' + Vnama2 + ' adalah praktikan DKP 2020 Kelompok ' + Vkelompok) |



Gambar 7. 8. Source Code Setter Getter dalam Python (1)



Gambar 7. 9. Source Code Setter Getter dalam Python (2)

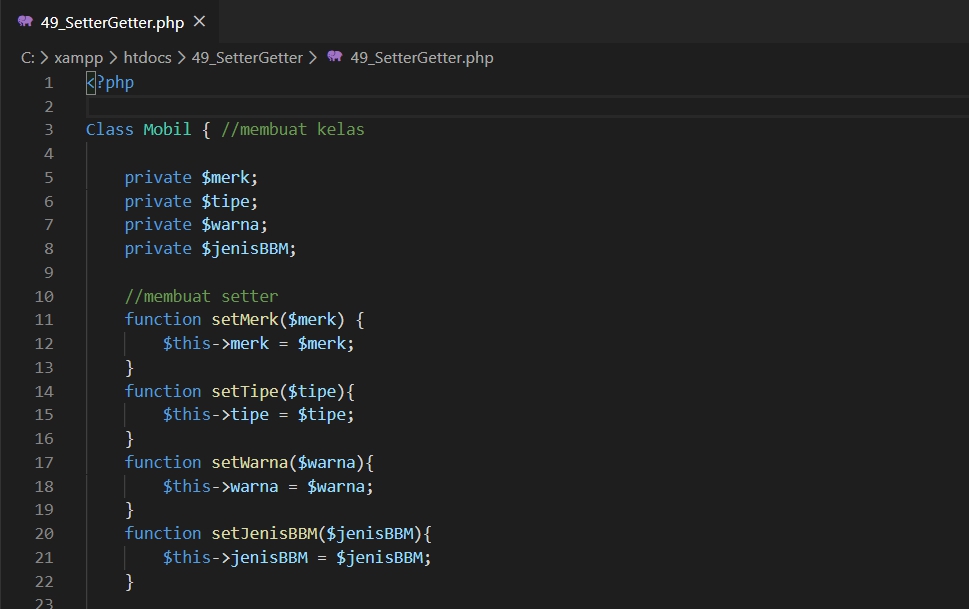


Gambar 7. 10. Output Setter Getter dalam Python

Setelah kita membuat *file* 49\_Setter\_Getter\_Phyton.py, Pertama-tama kita buat kelas dengan nama praktikum lalu kita buat *method inisialisasi*. kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* yang memiliki variabel yang perlu dimasukkan bila *method* ini dipanggil dan jangan lupa kita masukkan *syntax* untuk menyimpan nilai ke kelas praktikum. Setelah itu pada kelas utama kita panggil kelas praktikum lalu kita tetapkan variabel-variabel untuk disimpan ke kelas praktikum dengan menggunakan *method setter* yang telah kita buat pada kelas praktikum. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas praktikum dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

* + 1. **Bahasa Pemrograman PHP**

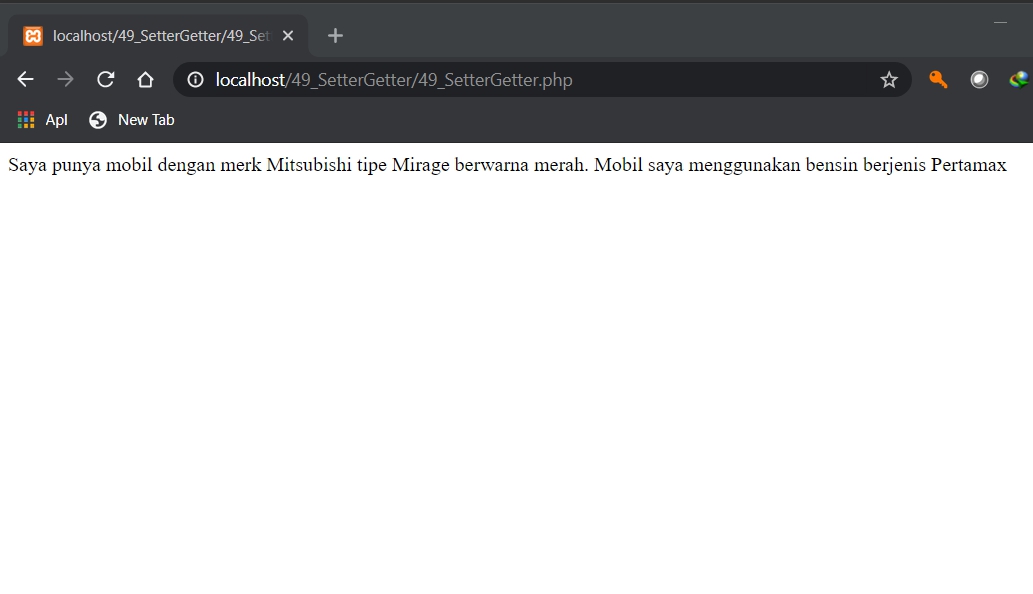
|  |
| --- |
| <?php  Class Mobil { //membuat kelas  private $merk;  private $tipe;  private $warna;  private $jenisBBM;    //membuat setter  function setMerk($merk) {  $this->merk = $merk;  }  function setTipe($tipe){  $this->tipe = $tipe;  }  function setWarna($warna){  $this->warna = $warna;  }  function setJenisBBM($jenisBBM){  $this->jenisBBM = $jenisBBM;  }    //membuat getter  function getMerk() {  return $this->merk;  }  function getTipe() {  return $this->tipe;  }  function getWarna() {  return $this->warna;  }  function getJenisBBM() {  return $this->jenisBBM;  }  } // akhir kelas  //membuat objek  $mobil\_baru = new Mobil();  //mengatur nilai  $mobil\_baru->setMerk("Mitsubishi");  $mobil\_baru->setTipe("Mirage");  $mobil\_baru->setWarna("merah");  $mobil\_baru->setJenisBBM("Pertamax");  //menampilkan data  echo "Saya punya mobil dengan merk ".$mobil\_baru->getMerk()." tipe ".$mobil\_baru->getTipe();  echo " berwarna ".$mobil\_baru->getWarna().". Mobil saya menggunakan bensin berjenis ".$mobil\_baru->getJenisBBM();    ?> |



Gambar 7. 11. Source Code Setter Getter dalam PHP (1)



Gambar 7. 12. Source Code Setter Getter dalam PHP (2)



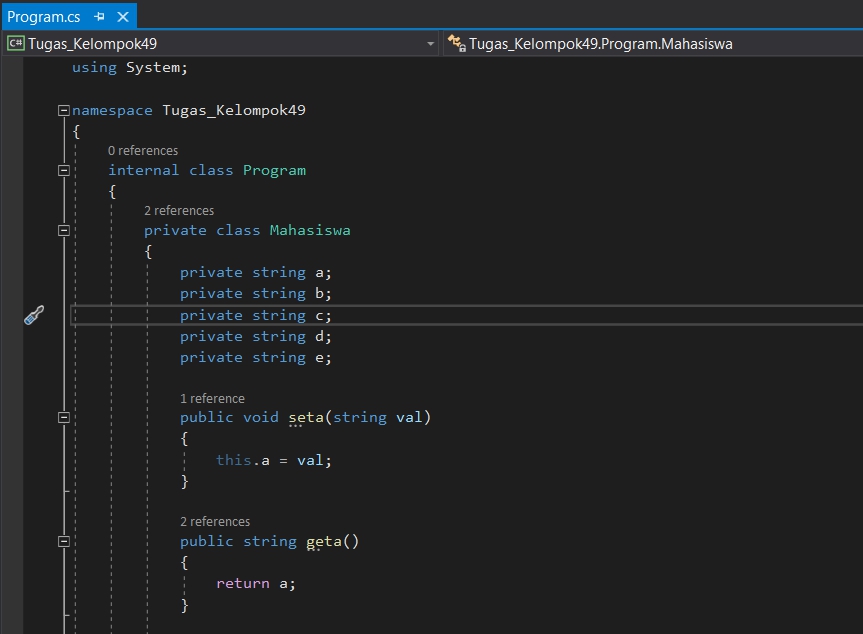
Gambar 7. 13. Output Setter Getter dalam PHP

Setelah kita membuat *file* 49\_SetterGetter.php, Pertama-tama kita buat kelas dengan nama mobil lalu kita definisikan variabel-variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method* lalu kita membuat *method setter* yang memiliki variabel yang perlu dimasukkan bila *method* ini dipanggil dan jangan lupa kita masukkan *syntax* untuk menyimpan nilai ke kelas praktikum. Setelah itu pada kelas utama kita panggil kelas mobil lalu kita tetapkan variabel-variabel untuk disimpan ke kelas mobil dengan menggunakan *method setter* yang telah kita buat pada kelas mobil. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas mobil dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

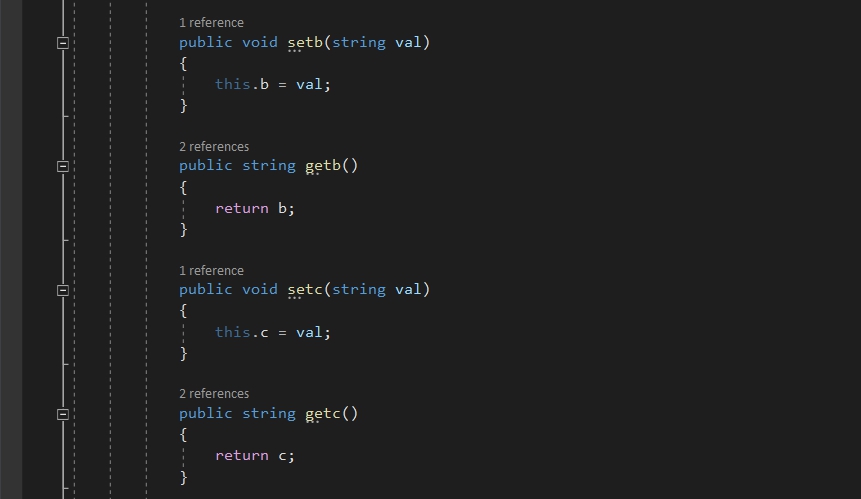
### **Tugas**

* + 1. **Bahasa Pemrograman C#**

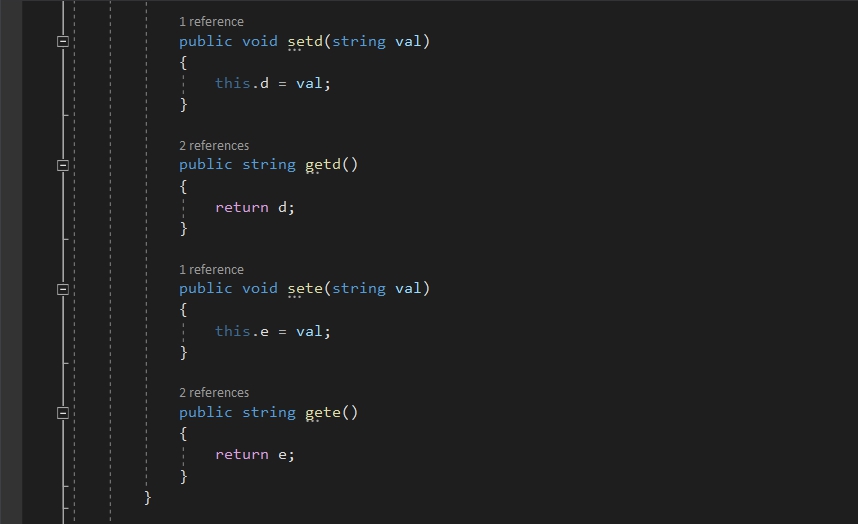
|  |
| --- |
| using System;  namespace Tugas\_Kelompok49  {  internal class Program  {  private class Mahasiswa  {  private string a;  private string b;  private string c;  private string d;  private string e;  public void seta(string val)  {  this.a = val;  }  public string geta()  {  return a;  }  public void setb(string val)  {  this.b = val;  }  public string getb()  {  return b;  }  public void setc(string val)  {  this.c = val;  }  public string getc()  {  return c;  }  public void setd(string val)  {  this.d = val;  }  public string getd()  {  return d;  }  public void sete(string val)  {  this.e = val;  }  public string gete()  {  return e;  }  }  private static void Main(string[] args)  {  Mahasiswa call = new Mahasiswa();  Console.Write("Nama : ");  string a = Console.ReadLine();  Console.Write("NIM : ");  string b = Console.ReadLine();  Console.Write("Kelompok : ");  string c = Console.ReadLine();  Console.Write("Hobi : ");  string d = Console.ReadLine();  Console.Write("Asal : ");  string e = Console.ReadLine();  Console.WriteLine();  call.seta(a);  call.setb(b);  call.setc(c);  call.setd(d);  call.sete(e);  Console.WriteLine(  "--------------------Output Data---------------------" +  "\nNama : " + call.geta() +  "\nNIM : " + call.getb() +  "\nKelompok : " + call.getc() +  "\nHobi : " + call.getd() +  "\nAsal : " + call.gete() +  "\n------------------------------------------------\n" +  "Nama mahasiswa adalah " + call.geta() +  " dengan NIM " + call.getb() +  " dari Kelompok " + call.getc() +  " dan berasal dari " + call.gete() +  " memiliki Hobi " + call.getd()  );  Console.ReadLine();  }  }  } |



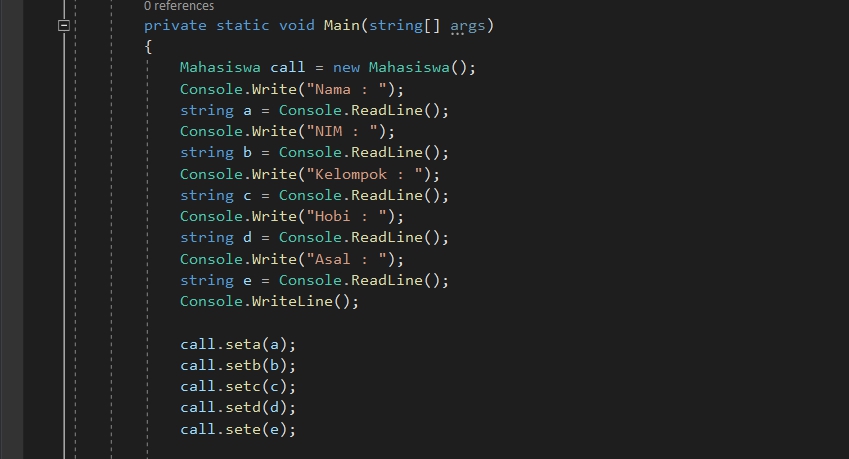
Gambar 7. 14. Source Code Tugas dalam C# (1)



Gambar 7. 15. Source Code Tugas dalam C# (2)



Gambar 7. 16. Source Code Tugas dalam C# (3)

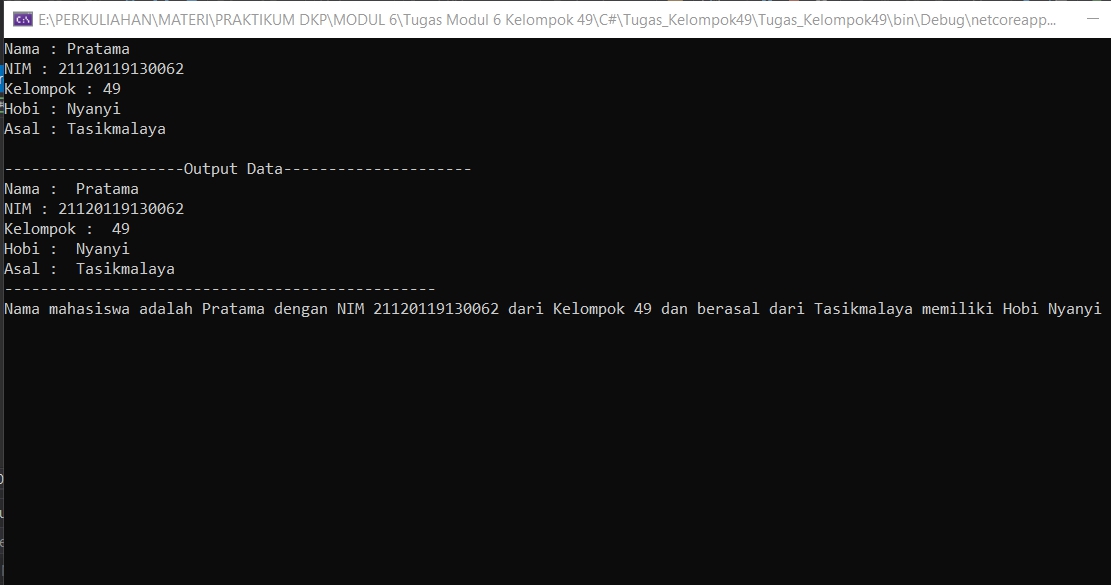
**

Gambar 7. 17. Source Code Tugas dalam C# (4)

**

Gambar 7. 18. Source Code Tugas dalam C# (5)

*Output* Hasil:



Gambar 7. 19. Output Tugas dalam C#

Penjelasan *Output* :

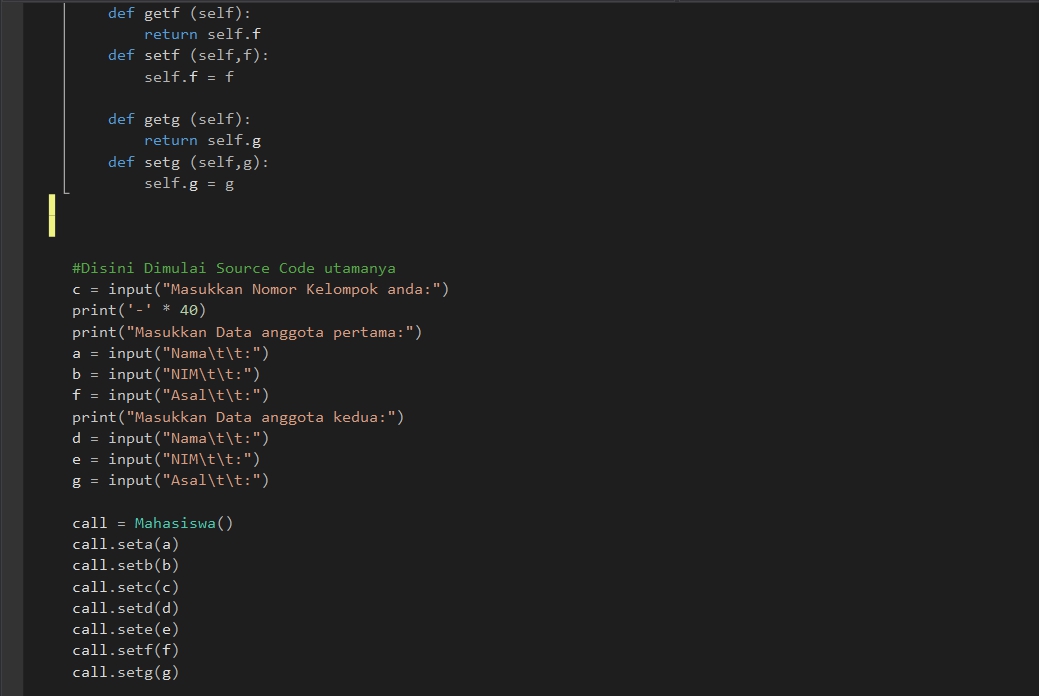
Setelah kita membuat *file* Tugas\_Kelompok49.cs, Pertama-tama kita buat kelas dengan nama Mahasiswa lalu kita definisikan variabel-variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter*, bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* maka jenis *method* ini adalah *void* karena fungsinya hanya menyimpan nilai yang masuk dan disimpan ke *directory* sementara dan jangan lupa kita buat variabel yang perlu dimasukkan supaya nilai dapat tersimpan. Setelah itu pada kelas utama kita panggil kelas Mahasiswa lalu kita tetapkan variabel-variabel untuk disimpan ke kelas Mahasiswa dengan menggunakan *method setter* yang telah kita buat pada kelas Mahasiswa. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas Mahasiswa dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

* + 1. **Bahasa Pemrograman Python**

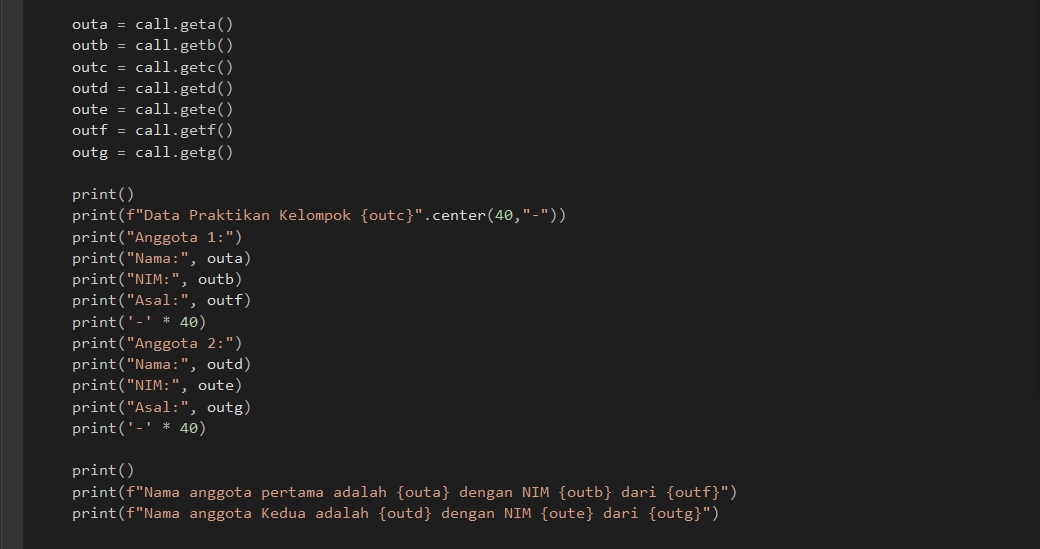
|  |
| --- |
| class Mahasiswa:  def \_\_init\_\_ (self):  self.a = ""  self.b = ""  self.c = ""  self.d = ""  self.e = ""  self.f = ""  def geta (self):  return self.a  def seta (self,a):  self.a = a  def getb (self):  return self.b  def setb (self,b):  self.b = b  def getc (self):  return self.c  def setc (self,c):  self.c = c  def getd (self):  return self.d  def setd (self,d):  self.d = d  def gete (self):  return self.e  def sete (self,e):  self.e = e  def getf (self):  return self.f  def setf (self,f):  self.f = f  def getg (self):  return self.g  def setg (self,g):  self.g = g  #Disini Dimulai Source Code utamanya  c = input("Masukkan Nomor Kelompok anda:")  print('-' \* 40)  print("Masukkan Data anggota pertama:")  a = input("Nama\t\t:")  b = input("NIM\t\t:")  f = input("Asal\t\t:")  print("Masukkan Data anggota kedua:")  d = input("Nama\t\t:")  e = input("NIM\t\t:")  g = input("Asal\t\t:")  call = Mahasiswa()  call.seta(a)  call.setb(b)  call.setc(c)  call.setd(d)  call.sete(e)  call.setf(f)  call.setg(g)  outa = call.geta()  outb = call.getb()  outc = call.getc()  outd = call.getd()  oute = call.gete()  outf = call.getf()  outg = call.getg()    print()  print(f"Data Praktikan Kelompok {outc}".center(40,"-"))  print("Anggota 1:")  print("Nama:", outa)  print("NIM:", outb)  print("Asal:", outf)  print('-' \* 40)  print("Anggota 2:")  print("Nama:", outd)  print("NIM:", oute)  print("Asal:", outg)  print('-' \* 40)  print()  print(f"Nama anggota pertama adalah {outa} dengan NIM {outb} dari {outf}")  print(f"Nama anggota Kedua adalah {outd} dengan NIM {oute} dari {outg}") |



Gambar 7. 20. Source Code Tugas dalam Python (1)



Gambar 7. 21. Source Code Tugas dalam Python (2)

Gambar 7. 22. Source Code Tugas dalam Python (3)

*Output* Hasil:



Gambar 7. 23. Output Tugas dalam Python

Penjelasan *Output* :

Setelah kita membuat *file* Tugas\_Kelompok49.py, Pertama-tama kita buat kelas dengan nama praktikum lalu kita buat *method inisialisasi*. kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* yang memiliki variabel yang perlu dimasukkan bila *method* ini dipanggil dan jangan lupa kita masukkan *syntax* untuk menyimpan nilai ke kelas praktikum. Setelah itu pada kelas utama kita panggil kelas praktikum lalu kita tetapkan variabel-variabel untuk disimpan ke kelas praktikum dengan menggunakan *method setter* yang telah kita buat pada kelas praktikum. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas praktikum dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

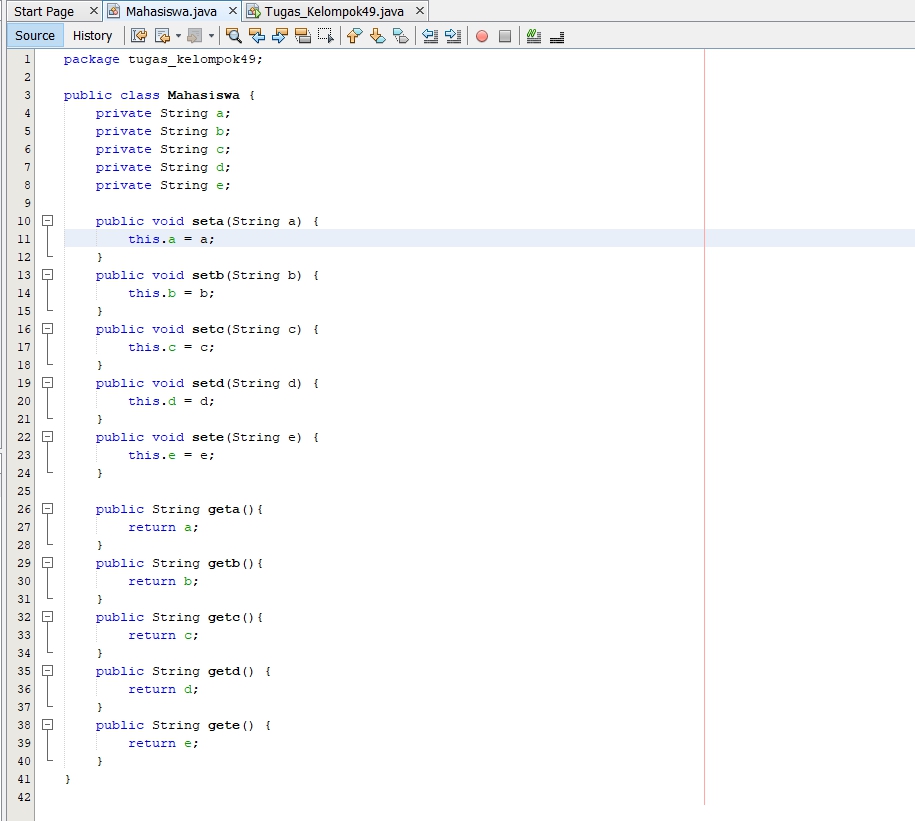
* + 1. **Bahasa Pemrograman Java**

Mahasiswa.java

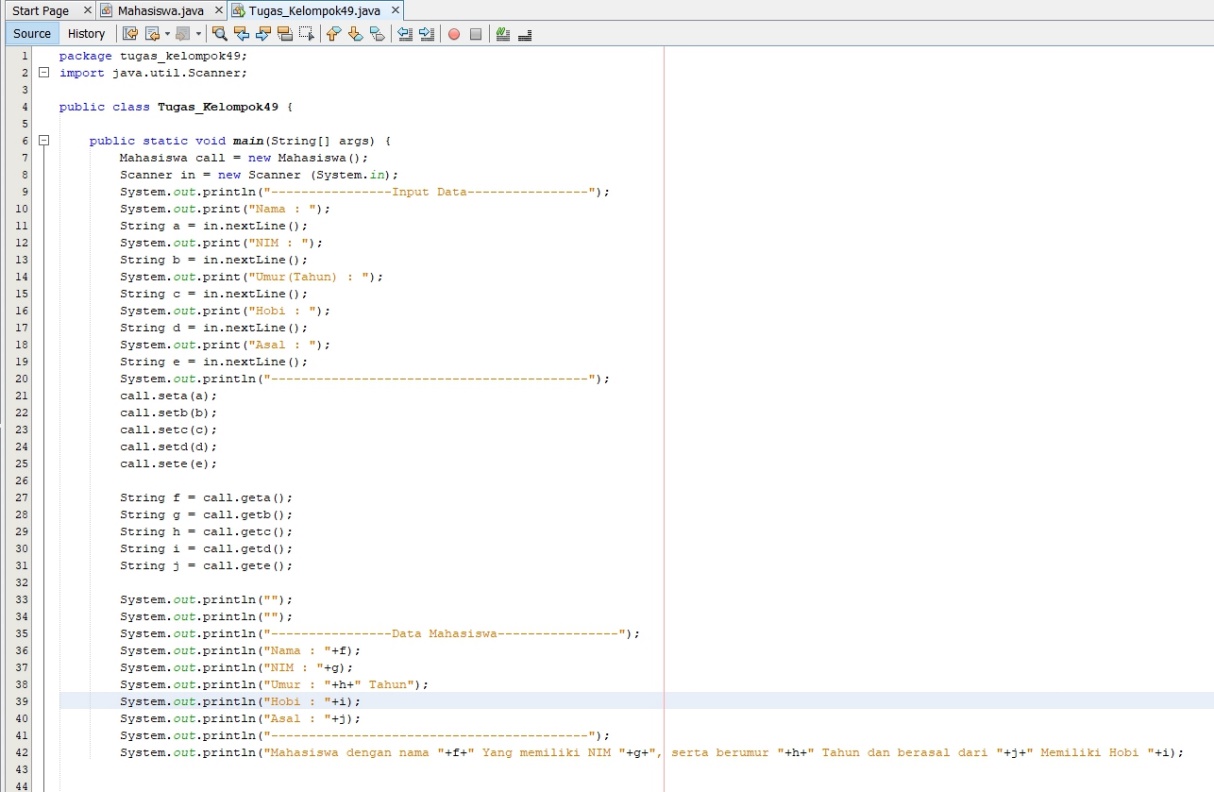
|  |
| --- |
| package tugas\_kelompok49;  public class Mahasiswa {  private String a;  private String b;  private String c;  private String d;  private String e;  public void seta(String a) {  this.a = a;  }  public void setb(String b) {  this.b = b;  }  public void setc(String c) {  this.c = c;  }  public void setd(String d) {  this.d = d;  }  public void sete(String e) {  this.e = e;  }    public String geta(){  return a;  }  public String getb(){  return b;  }  public String getc(){  return c;  }  public String getd() {  return d;  }  public String gete() {  return e;  }  } |

Tugas\_Kelompok49.java

|  |
| --- |
| package tugas\_kelompok49;  import java.util.Scanner;  public class Tugas\_Kelompok49 {  public static void main(String[] args) {  Mahasiswa call = new Mahasiswa();  Scanner in = new Scanner (System.in);  System.out.println("----------------Input Data----------------");  System.out.print("Nama : ");  String a = in.nextLine();  System.out.print("NIM : ");  String b = in.nextLine();  System.out.print("Umur(Tahun) : ");  String c = in.nextLine();  System.out.print("Hobi : ");  String d = in.nextLine();  System.out.print("Asal : ");  String e = in.nextLine();  System.out.println("------------------------------------------");  call.seta(a);  call.setb(b);  call.setc(c);  call.setd(d);  call.sete(e);    String f = call.geta();  String g = call.getb();  String h = call.getc();  String i = call.getd();  String j = call.gete();    System.out.println("");  System.out.println("");  System.out.println("----------------Data Mahasiswa----------------");  System.out.println("Nama : "+f);  System.out.println("NIM : "+g);  System.out.println("Umur : "+h+" Tahun");  System.out.println("Hobi : "+i);  System.out.println("Asal : "+j);  System.out.println("------------------------------------------");  System.out.println("Mahasiswa dengan nama "+f+" Yang memiliki NIM "+g+", serta berumur "+h+" Tahun dan berasal dari "+j+" Memiliki Hobi "+i);          }    } |

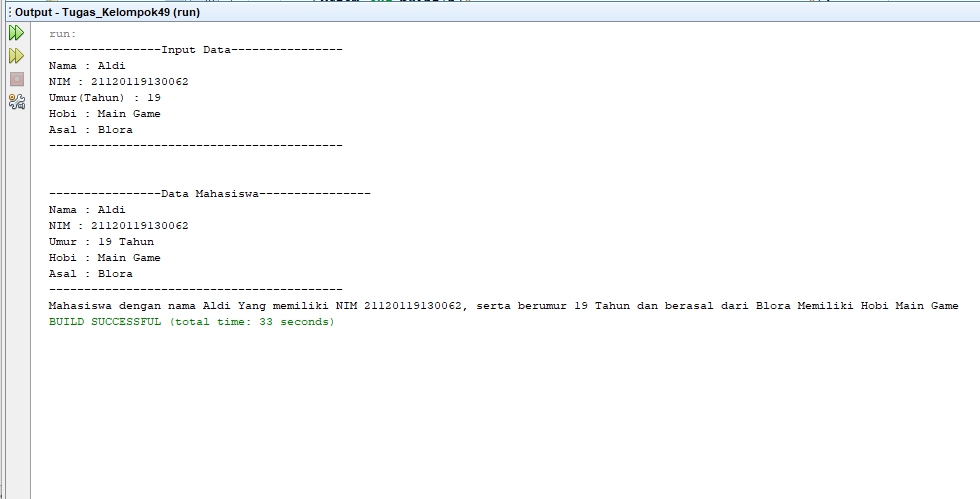


Gambar 7. 24. Source Code Tugas dalam Java (1)



Gambar 7. 25. Source Code Tugas dalam Java (2)

*Output* Hasil:



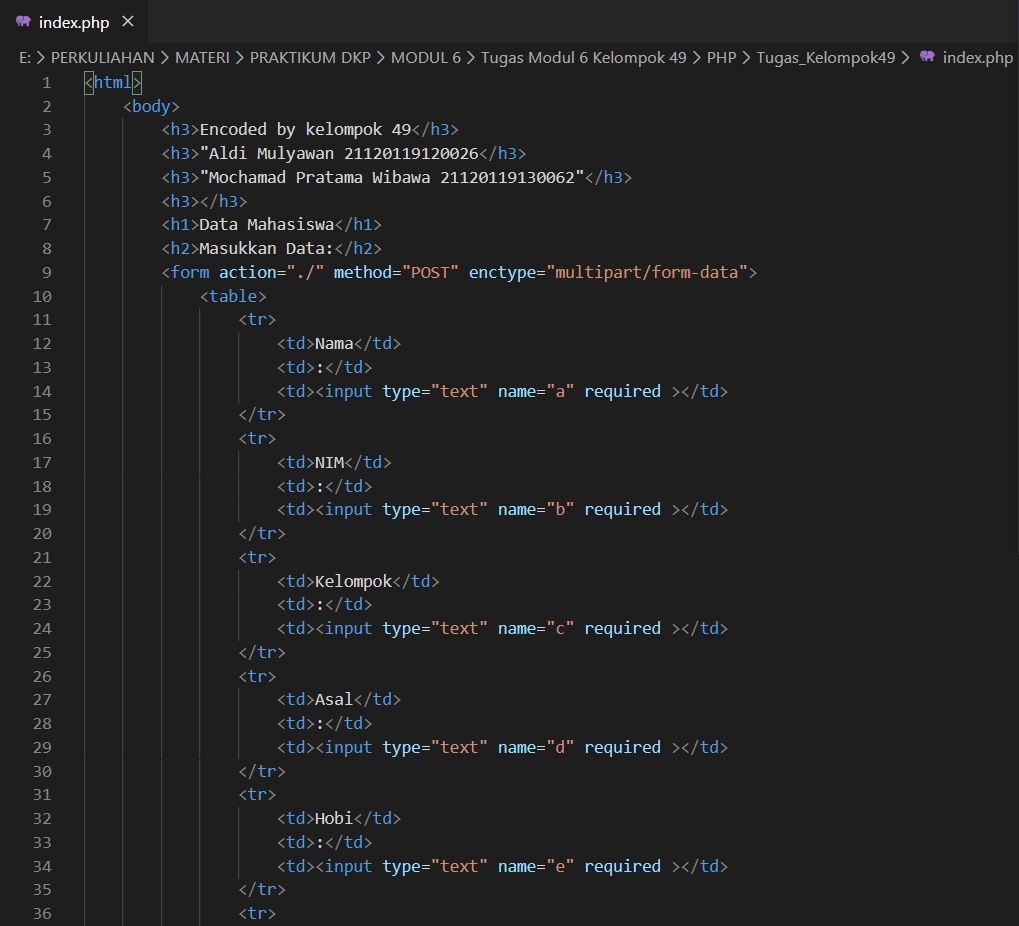
Gambar 7. 26. Output Tugas dalam Java

Penjelasan *Output* :

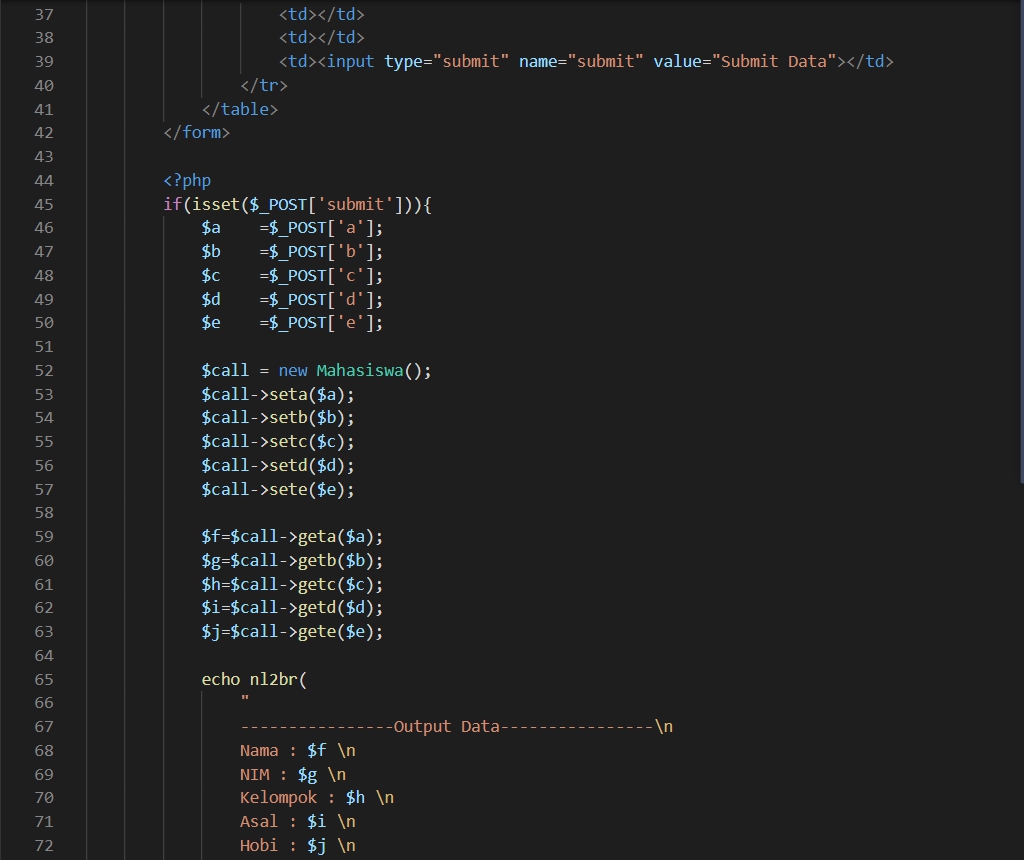
Setelah kita membuat *file* Tugas\_Kelompok49.java, Pertama-tama kita buat file dengan nama Mahasiswa.java dan buat kelas Mahasiswa lalu kita definisikan variabel-variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter*, bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* di akhir *method*, lalu kita membuat *method setter* maka jenis *method* ini adalah *void* karena fungsinya hanya menyimpan nilai yang masuk dan disimpan ke directory sementara dan jangan lupa kita buat variabel yang perlu dimasukkan supaya nilai dapat tersimpan. Setelah itu pada file Tugas\_Kelompok49.java dengan kelas Tugas\_Kelompok49. kita panggil kelas Mahasiswa lalu kita tetapkan variabel-variabel untuk disimpan ke kelas Mahasiswa dengan menggunakan *method setter* yang telah kita buat pada kelas Mahasiswa. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas Mahasiswa dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

* + 1. **Bahasa Pemrograman PHP**

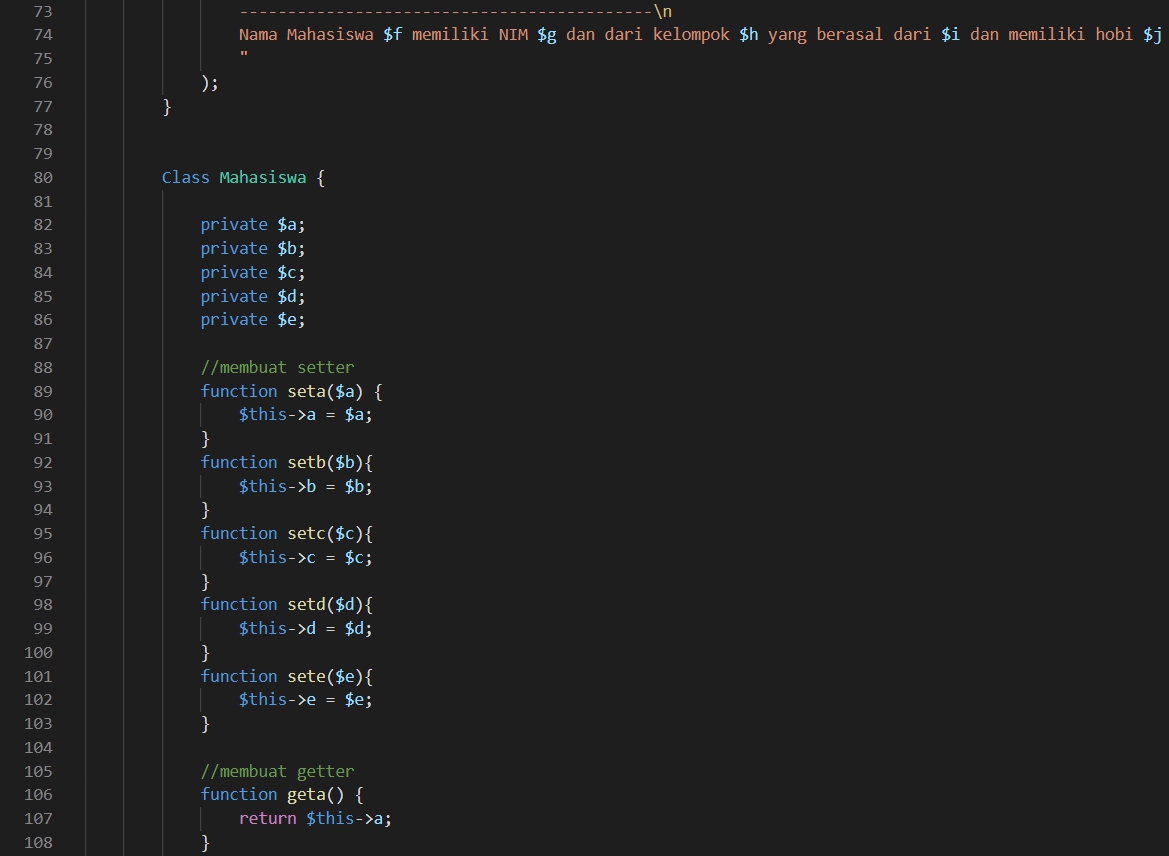
|  |
| --- |
| <html>  <body>  <h3>Encoded by kelompok 49</h3>  <h3>"Aldi Mulyawan 21120119120026</h3>  <h3>"Mochamad Pratama Wibawa 21120119130062"</h3>  <h3></h3>  <h1>Data Mahasiswa</h1>  <h2>Masukkan Data:</h2>  <form action="./" method="POST" enctype="multipart/form-data">  <table>  <tr>  <td>Nama</td>  <td>:</td>  <td><input type="text" name="a" required ></td>  </tr>  <tr>  <td>NIM</td>  <td>:</td>  <td><input type="text" name="b" required ></td>  </tr>  <tr>  <td>Kelompok</td>  <td>:</td>  <td><input type="text" name="c" required ></td>  </tr>  <tr>  <td>Asal</td>  <td>:</td>  <td><input type="text" name="d" required ></td>  </tr>  <tr>  <td>Hobi</td>  <td>:</td>  <td><input type="text" name="e" required ></td>  </tr>  <tr>  <td></td>  <td></td>  <td><input type="submit" name="submit" value="Submit Data"></td>  </tr>  </table>  </form>  <?php  if(isset($\_POST['submit'])){  $a =$\_POST['a'];  $b =$\_POST['b'];  $c =$\_POST['c'];  $d =$\_POST['d'];  $e =$\_POST['e'];  $call = new Mahasiswa();  $call->seta($a);  $call->setb($b);  $call->setc($c);  $call->setd($d);  $call->sete($e);    $f=$call->geta($a);  $g=$call->getb($b);  $h=$call->getc($c);  $i=$call->getd($d);  $j=$call->gete($e);    echo nl2br(  "  ----------------Output Data----------------\n  Nama : $f \n  NIM : $g \n  Kelompok : $h \n  Asal : $i \n  Hobi : $j \n  -------------------------------------------\n  Nama Mahasiswa $f memiliki NIM $g dan dari kelompok $h yang berasal dari $i dan memiliki hobi $j  "  );  }  Class Mahasiswa {  private $a;  private $b;  private $c;  private $d;  private $e;    //membuat setter  function seta($a) {  $this->a = $a;  }  function setb($b){  $this->b = $b;  }  function setc($c){  $this->c = $c;  }  function setd($d){  $this->d = $d;  }  function sete($e){  $this->e = $e;  }    //membuat getter  function geta() {  return $this->a;  }  function getb() {  return $this->b;  }  function getc() {  return $this->c;  }  function getd() {  return $this->d;  }  function gete() {  return $this->e;  }  }  ?> |



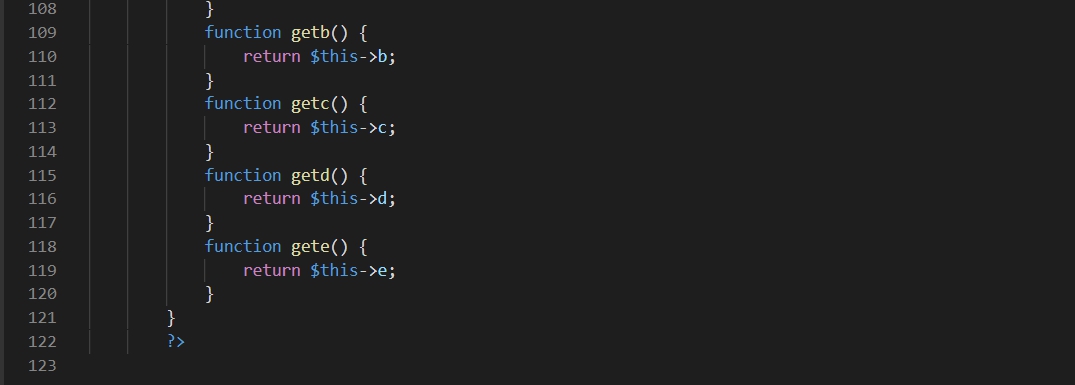
Gambar 7. 27. Source Code Tugas dalam PHP (1)



Gambar 7. 28. Source Code Tugas dalam PHP (2)

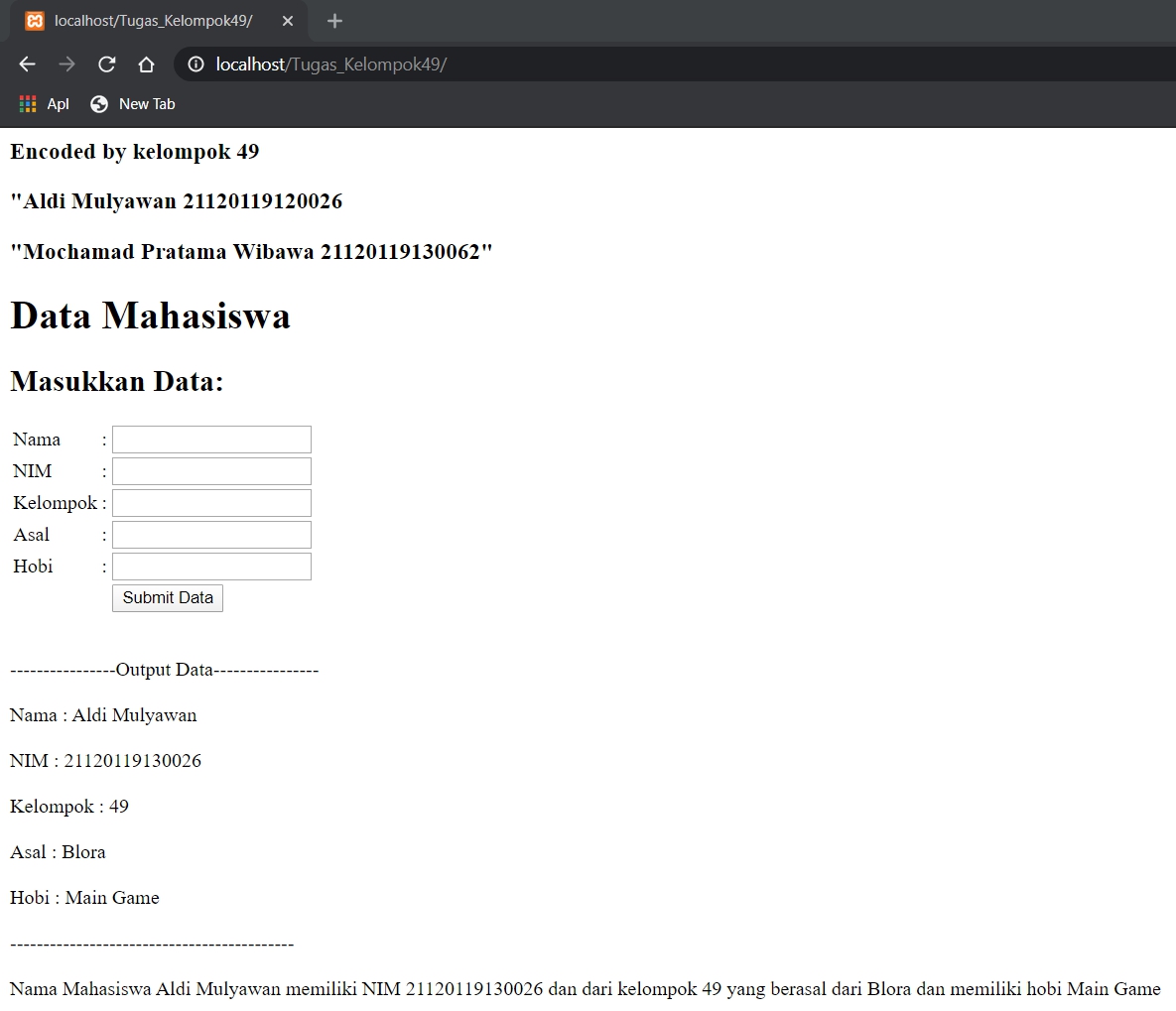


Gambar 7. 29. Source Code Tugas dalam PHP (3)



Gambar 7. 30. Source Code Tugas dalam PHP (4)

*Output* Hasil:



Gambar 7. 31. Output Tugas dalam PHP

Penjelasan *Output* :

Setelah kita membuat *file* Tugas\_Kelompok49.php Pertama-tama kita buat kelas dengan nama Mahasiswa lalu kita definisikan variabel-variabel yang akan digunakan. Lalu kita buat *method setter* dan *getter* , bila kita membuat *method getter* maka *method* kita harus dapat mengembalikan nilai dan memiliki *return* diakhir *method* lalu kita membuat *method setter* yang memiliki variabel yang perlu dimasukkan bila *method* ini dipanggil dan jangan lupa kita masukkan *syntax* untuk menyimpan nilai ke kelas praktikum. Setelah itu pada kelas utama kita panggil kelas Mahasiswa lalu kita tetapkan variabel-variabel untuk disimpan ke kelas Mahasiswa dengan menggunakan *method setter* yang telah kita buat pada kelas Mahasiswa. Selanjutnya kita panggil kembali nilai yang telah kita simpan di kelas Mahasiswa dengan menggunakan *method getter* dan selanjutnya nilai yang telah kita ambil dapat langsung kita keluarkan dan mengeluarkan hasil yang kita inginkan.

### **Kesimpulan**

* + 1. *UDT, Setter* dan *Getter* memiliki hubungan yang saling berkaitan.
    2. Penggunaan *Setter* dan *Getter* bertujuan agar menghemat ukuran program, dapat dipakai berkali-kali dan memudahkan dalam mengembangkannya.
    3. Penggunaan *Setter* atau *Getter* dipilih berdasarkan kebutuhan dan urgensinya masing-masing.

1. Penggunaan *Setter* digunakan disaat kita ingin memasukkan data ke kelas lain.
2. Penggunaan *Getter* digunakan disaat kita ingin mengambil data dari kelas lain.
3. Data yang disimpan menggunakan *Setter* cenderung lebih aman dan memiliki fleksibilitas lebih tinggi.
4. Kita dapat menentukan siapa yang dapat mengambil data yang telah tersimpan dengan menggunakan *Setter Getter*.
5. *UDT* biasanya dipakai di awal kelas untuk mendefinisikan variabel-variabel yang akan dipakai.
6. Supaya kita dapat memanggil *Setter* kita harus memasukkan nilai/variabel tertentu, sedangkan untuk *Getter* kita tidak memerlukan nilai/variabel tertentu untuk memanggil *Getter*.