**BAB VII**

**Object Oriented Programming II**

## 7.1. Tujuan

* 1. Praktikan dapat memahami konsep dasar *polymorphism, abstraction,* dan *encapsulation*.
  2. Praktikan dapat memahami apa itu UDT.
  3. Praktikan dapat memahami konsep *getter* dan *setter*.
  4. Praktikan dapat memahami jenis-jenis *modifier*.

## 7.2. Dasar Teori

1. **Polymorphism**

*Polymorphism* terbagi menjadi dua suku kata yaitu, *Poly*yang berartibanyak dan *Morfisme* yang berarti bentuk. *Polymorphism* dalam OOP merupakan sebuah konsep OOP di mana *class*memiliki banyak “bentuk” *method*yang berbeda, meskipun namanya sama. Maksud dari “bentuk” adalah isinya yang berbeda, namun tipe data dan parameternya berbeda.

*Polymorphism* juga dapat diartikan sebagai teknik *programming* yang mengarahkan programmer untuk memprogram secara general daripada secara spesifik. Contohnya kita memiliki tiga class yang berbeda yaitu: “Kelinci”, “Kucing”, dan “Sapi”. Di mana ketiga *class* tersebut merupakan turunan dari *class* “Hewan”.

**2**. **Abstraction**

*Abstraction* (Berasal dari bahasa Latin abs, yang berarti jauh dan trahere, yang berarti menggambar) adalah suatu proses dimana kita menghilangkan atau menghapus karakteristik dari sesuatu untuk mengurangi kompleksitasnya, menjadi seperangkat karakteristik penting.

Dalam pemrograman berorientasi objek, *Abstraction* adalah satu dari tiga prinsip sentral (Enkapsulasi dan Pewarisan). Melalui proses abstraksi, seorang programmer menyembunyikan semua data (informasi dari suatu object) tapi hanya data yang tidak relevan saja, jadi hanya menggunakan informasi yang bisa benar-benar menggambarkan suatu entitas (Objek). Hal itu bertujuan untuk mengurangi kompleksitas dan meningkatkan efisiensi.

Dengan cara yang sama bahwa *Abstraction* kadang-kadang bekerja seperti sebuah seni, dimana objek yang tersisa adalah representasi dari object aslinya, dengan syarat detail yang tidak menggambarkan atau berhubungan dengan object tertentu akan di hilangkan.

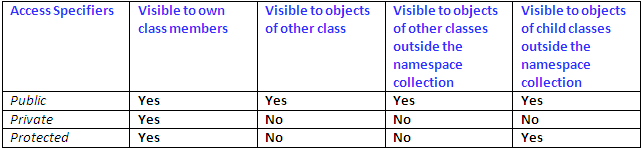
Objek yang dihasilkan itu sendiri dapat disebut sebagai *Abstraction*, yang berarti entitas yang terdiri dari atribut dan perilaku yang sudah terpilih secara spesifik akan di gunakan untuk membentuk entitas asal tertentu.

**3**. **Encapsulation**

*Encapsulation* merupakan implementasi penyembunyian informasi (information hiding). Tujuannya adalah untuk menyembunyikan informasi data objek sehingga tidak terlihat dari luar. Dengan demikian, informasi tersebut tidak dapat diakses sembarangan.

*Encapsulation* dapat dilakukan pada saat class, deklarasi atribut, ataupun pada method yang ada. Penerapan enkapsulasi dilakukan dengan memberikan hak akses pada class, atribut, dan method. Terdapat 3 jenis hak akses, yaitu public, protected, dan private.

* **Public** → ketika suatu atribut atau method diberi access modifier public, maka seluruh atribut atau method tersebut dapat diakses secara bebas oleh class lain.
* **Protected** → ketika suatu atribut atau method diberi access modifier protected, maka seluruh atribut atau method tersebut tidak dapat diakses class lain, hanya class turunannya saja yang bisa mengakses.
* **Private** → ketika suatu atribut atau method diberi access modifier private, maka seluruh atribut atau method tersebut tidak dapat diakses class lain, hanya dapat diakses oleh class tersebut.



**4. UDT (User Data Type)**

UDT (User Data Type) merupakan salah satu fitur dalam pemrograman berorientasi objek dimana programmer dapat membuat tipe data/objek yang berisi satu atau lebih variabel didalamnya. Manfaat menggunakan UDT adalah, ketika kita ingin mengirim beberapa variabel relevan secara bersamaan beberapa kali dari satu class ke class lain akan lebih mudah daripada harus mengirim variabel tersebut satu persatu.

**5. Setter**

*Setter* adalah sebuah *method* yang digunakan untuk memberikan nilai pada suatu attirubte, object, list, entity, dllnya. *Method setter* tidak memiliki pengembalian nilai, cirinya method ini method yang berisi kata void pada awal penulisan method.

**6. Getter**

*Method* *getter* adalah kebalikan dari *setter* yaitu pengambilan nilai dari suatu object atau attribute yang sudah berisi nilai.

Berikut adalah struktur yang digunakan untuk menggunakan UDT, Setter dan Getter

class Main

{

public static void main(String[] args) // metode main

{

String x; // deklarasi variabel

SetterGetter object = new SetterGetter(); // membuat objek baru

object.set("value"); // set nilai ke variabel objek

x = object.get(); // get nilai dari variabel objek

System.out.println(x); // cetak nilai variabel ke layar

}

}

class SetterGetter // kelas setter getter atau objek

{

private String variable; // variabel atau atribut

public void set(String value) // method buat set variabel

{

this.variable = value; // menetapkan nilai ke variabel

}

public String get() // method buat get variabel

{

return variable; // mengembalikan nilai variabel ke fungsi atau kelas yang memanggilnya

}

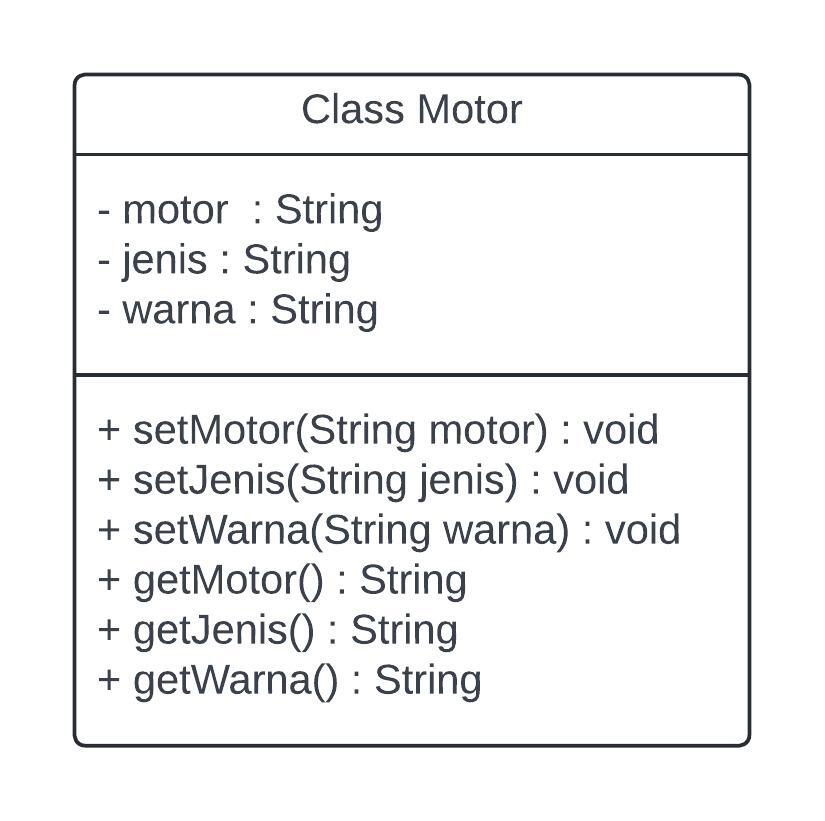
}

Pada kelas SetterGetter kita mendeklarasikan variabel yang menampung nilai dari variabel, jumlah variabel dalam kelas atau objek ini dapat berjumlah lebih dari satu sesuai kebutuhan. Setiap variabel/atribut dalam sebuah objek/UDT masing-masing memiliki 2 method yaitu method setter dan method getter. Pada method setter, method berjenis void dan memiliki atribut untuk menampung nilai dari fungsi yang memanggilnya. Method setter berisi penetapan nilai dari fungsi yang memanggil method setter ke variabel didalam objek tersebut. Sedangkan pada method getter, method berjenis non-void dan seuai dengan tipe data atribut objek tersebut. Method getter berisi perintah return yang akan mengembalikan nilai atribut yang telah ditetapkan oleh method setter sebelumnya ke fungsi yang memanggil method getter.

Pada kelas Main, kita dapat membuat objek baru dengan mendeklarasikan objek berjenis SetterGetter. Kemudian dari objek yang telah dibuat, kita dapat memanggil method setter dan membeberinya nilai untuk menentukan nilai dari atribut objek tersebut. Untuk mendapatkan nilai dari attribut objek tadi, kita bisa memanggil method get.

## 7.3. Percobaan

### 7.3.1. Java



1. Buka Intellij IDEA Community
2. Kemudian pilih *new project* (*file* → *new project*)
3. Pilih java kemudian *next*, *next*, beri nama *project* Modul6\_kelXX kemudian *Create*.

A screenshot of a computer

Description automatically generated

1. Setelah itu, pada bagian *project* (sebelah kiri) klik kanan pada folder src, kemudian *hover* pilihan *new*, kemudian pilih Java Class

A screenshot of a computer

Description automatically generated

1. Berikan nama Modul6\_KelXX kemudian *enter*
2. Masukan *sourcecode* yang berada di dalam Main berikut ini pada *Class* yang telah dibuat :

public class Modul6\_KelXX { //membuat class

//deklarasi atribut

private String motor;

private String jenis;

private String warna;

//membuat setter

public void setMotor(String motor) {

this.motor = motor;

}

public void setJenis(String jenis) {

this.jenis = jenis;

}

public void setWarna(String warna) {

this.warna = warna;

}

//membuat getter

public String getMotor(){

return motor;

}

public String getJenis(){

return jenis;

}

public String getWarna(){

return warna;

}

}

Selanjutnya, buatlah class baru Bernama Praktikum\_DKP untuk mengimplementasikan class yang telah dibuat tadi.

public class Praktikum\_DKP {

public static void main(String[] args) {

Modul6\_KelXX kiwkiw = new Modul6\_KelXX();

//mengatur nilai atribut

kiwkiw.setMotor("Aerox");

kiwkiw.setJenis("Yamaha");

kiwkiw.setWarna("Merah");

//mencetak nilai

System.out.print("Saya punya Motor " + kiwkiw.getMotor());

System.out.print(" dengan Merk "+ kiwkiw.getJenis());

System.out.print(", Berwarna "+ kiwkiw.getWarna());

}

}

Hasil yang di dapat adalah sebagai berikut :

A screenshot of a computer

Description automatically generated

### 7.3.2. C++

1. Buka IDE Code Blocks
2. Pilih *new,* pilih projects,lalu Console Application

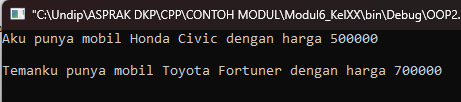
A screenshot of a computer program

Description automatically generated

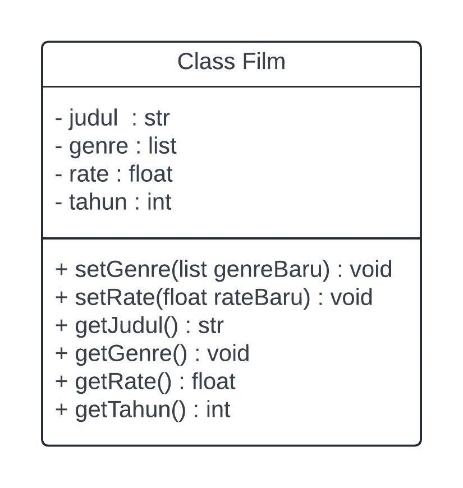
1. Klik *Next*, lalu pilih C++
2. Buat project deangan nama Modul6\_KelXX
3. Masukkan *source* *code* berikut:

|  |
| --- |
| #include <iostream>  using namespace std;  class Mobil {  //Encapsulation  private:  string merk, nama;  int harga;  public:  // Setter  void setMerk(string m) {  merk = m;  }  void setNama(string n) {  nama = n;  }  void setHarga(int h) {  harga = h;  }  // Getter  string getMerk() {  return merk;  }  string getNama() {  return nama;  }  int getHarga() {  return harga;  }  };  int main() {  Mobil rugidong;  rugidong.setMerk("Honda");  rugidong.setNama("Civic");  rugidong.setHarga(500000);  cout << "Aku punya mobil " + rugidong.getMerk() + " " + rugidong.getNama() + " dengan harga " ;  cout << rugidong.getHarga();  cout << endl; cout << endl;  rugidong.setMerk("Toyota");  rugidong.setNama("Fortuner");  rugidong.setHarga(700000);  cout << "Temanku punya mobil " + rugidong.getMerk() + " " + rugidong.getNama() + " dengan harga ";  cout << rugidong.getHarga();  cout << endl; cout << endl;  return 0;  } |

Hasil yang di dapat adalah sebagai berikut :



### Python



1. Buka Visual Studio
2. Pilih *create a new project*
3. Beri nama *project* Modul6\_KelXX kemudian klik *create*
4. Masukkan *source* *code* berikut:

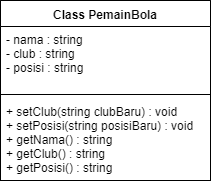
|  |
| --- |
| class film(object):  # description of class  \_\_judul = ""  \_\_genre = ["", ""]  \_\_rate = 0.0  \_\_tahun = 0  def \_\_init\_\_(self, judul, genre, rate, tahun):  self.\_\_judul = judul  self.\_\_genre = genre  self.\_\_rate = rate  self.\_\_tahun = tahun  pass  def setGenre(self, genreBaru):  self.\_\_genre = genreBaru  pass  def setRate(self, rateBaru):  self.\_\_rate = rateBaru  pass  def getJudul(self):  return self.\_\_judul  def getGenre(self):  for item in self.\_\_genre:  print(item)  pass  def getRate(self):  return self.\_\_rate  def getTahun(self):  return self.\_\_tahun  ygbeneraje = film("The Avengers", ["Action"], 7.9, 2012)  print("Kelompok xx")  print("-- Tahun 2023 --")  print("Judul : " + ygbeneraje.getJudul())  print("Genre : ")  ygbeneraje.getGenre()  print("Rate : " + str(ygbeneraje.getRate()))  print("Tahun : " + str(ygbeneraje.getTahun()))  ygbeneraje.setGenre(["Action", "Alien"])  ygbeneraje.setRate(9.9)  print("\n-- Tahun 2024 --")  print("Judul : " + ygbeneraje.getJudul())  print("Genre : ")  ygbeneraje.getGenre()  print("Rate : " + str(ygbeneraje.getRate()))  print("Tahun : " + str(ygbeneraje.getTahun())) |

Hasil yang di dapat adalah sebagai berikut :

A screen shot of a computer

Description automatically generated

### PHP



1. Buka IDE *Visual Studio Code*
2. Pilih *New File*
3. *Save as* *file* dengan nama Modul6\_kelompokxx.php di dalam folder yang bernama Modul6\_kelompokxx pada folder htdocs xampp.
4. Kemudian tuliskan *source code* berikut

|  |
| --- |
| <?php  class PemainBola {  private $nama;  private $club;  private $posisi;  public function \_\_construct($nama, $club, $posisi){  $this->nama = $nama;  $this->club = $club;  $this->posisi = $posisi;  }  public function setClub($clubBaru){  $this->club = $clubBaru;  }  public function setPosisi($posisiBaru){  $this->posisi = $posisiBaru;  }  public function getNama(){  return $this->nama;  }  public function getClub(){  return $this->club;  }  public function getPosisi(){  return $this->posisi;  }  }  $mangeak = new PemainBola("Cristian Ronaldo Siuuuuuu", "Manchester United", "Striker");  echo "-- Tahun 2023 -- <br>";  echo $mangeak->getNama() . "<br>";  echo $mangeak->getClub() . "<br>";  echo $mangeak->getPosisi() . "<br>";  $mangeak->setClub("Al-Nasr");  $mangeak->setPosisi("Kiper");  echo "-- Tahun 2024 -- <br>";  echo $mangeak->getNama() . "<br>";  echo $mangeak->getClub() . "<br>";  echo $mangeak->getPosisi() . "<br>";  ?> |

1. Untuk menjalankan *listing* di atas, buka XAMPP *Control Panel*
2. Start Apache dan MySQL
3. Kemudian buka browser ketikkan http://localhost/Modul6\_kelompokxx/Modul6\_kelompokxx.php

*Output* :

A screenshot of a computer

Description automatically generated