Dasar-Dasar Pemrograman 2

Lab 05

Object-Oriented Programming (OOP)

Object Oriented Programming

Object Oriented Programming (OOP) adalah sebuah paradigma pemrograman yang berdasarkan dari konsep objek. Dalam paradigma ini, kita memodelkan suatu objek dalam dunia nyata menjadi suatu *class* dalam program. Objek tersebut memiliki *attributes*, constructor dan methods.

Sebagai contoh, kita akan memodelkan objek **Mobil** menjadi sebuah class. Mobil yang kita modelkan memiliki merek, model, warna, kelajuan. Mobil dapat dipercepat (akselerasi) kelajuannya dan juga dapat diperlambat (deselerasi). Anggap juga terdapat counter jumlah mobil yang telah dibuat. Berikut implementasi dari objek Mobil tersebut.

```
public class Mobil{
   private String merek;
   private String model;
   private String warna;
   private double kelajuan;

    //class' Constructor
   public Mobil(String merek, String model, String warna){
       this.merek = merek;
       this.model = model;
       this.warna = warna;
       this.kelajuan = 0;
       jumlah += 1;
   }
```

```
    //class's method
    public akselerasi(double perubahan){
        this.kelajuan += perubahan;
    }

    //class's method
    public deselarasi(double perubahan){
        this.kelajuan += perubahan;
    }
}
```

Attributes

Attributes adalah variabel yang dimiliki oleh sebuah objek. Berikut cara menulis *attribute* dalam class.

```
public class Mobil{
    private String merek;
    private String model;
    private String warna;
    private double kelajuan;

    // Constructor dan method-method lainnya
}
```

Setiap attribute memiliki 3 komponen, yaitu access modifier, non-access modifier (modifier selain access_modifier) dan *data type attribute*, dan *nama attribute*. Berikut gambaran dari komponen-komponen attribute.

```
accessModifier nonAccessModifier type_data name;
```

Mungkin kita tidak perlu membahas lagi *data type* dan *name* karena sudah dibahas di materi-materi sebelumnya.

- ## Access Modifier

  Access modifier adalah tingkat akses dari sebuah variabel atau fungsi. Di materi ini kita akan membahas 4 access modifier, yaitu **public**, **private**, **default**, dan **protected**.

  Modifier **public** adalah penanda bahwa sebuah variabel atau fungsi dapat diakses oleh **semua class**.

  Modifier **private** adalah sebuah adalah penanda bahwa sebuah variabel atau fungsi hanya dapat diakses oleh **class itu sendiri.**

Modifier **default** adalah sebuah adalah penanda bahwa sebuah variabel atau fungsi hanya dapat diakses oleh **package yang sama. Anda tidak perlu menuliskan default pada access modifier variabel atau fungsi, dikosongkan saja.**

Modifier **protected** adalah sebuah adalah penanda bahwa sebuah variabel atau fungsi hanya dapat diakses oleh **subclass dan package yang sama** (Akan dibahas lebih lanjut di materi Inheritance).

Kenapa kita memerlukan **access modifier**? Dengan **access modifier** kita dapat mengontrol class / package mana yang dapat mengakses sebuah variabel atau fungsi tersebut. Hal ini yang berkaitan dengan salah satu konsep utama dari OOP, yaitu **encapsulation**.

Encapsulation atau pengkapsulan adalah konsep tentang pembungkusan data atau metode yang berbeda yang disatukan atau "dikapsulkan" menjadi satu unit data. Encapsulation dapat mempermudah dalam pembacaan code karena informasi yang disajikan tidak perlu dibaca secara rinci dan sudah merupakan satu kesatuan. (source : https://www.jagoanhosting.com/blog/).

Berikut tabel dari access Modifier.

	default	private	protected	public
Same Class	Yes	Yes	Yes	Yes
Same package subclass	Yes	No	Yes	Yes
Same package non-subclass	Yes	No	Yes	Yes
Different package subclass	No	No	Yes	Yes
Different package non-subclass	No	No	No	Yes

Constructor

Constructor adalah *method* untuk membuat instansiasi objek dari *class*. Constructor adalah *method* khusus yang akan dieksekusi pada saat pembuatan objek (*instance*). Pada umumnya, *method* ini berisi inisialisasi variabel untuk objek.

```
public class Mobil{

   //...inisialisasi variabel
```

```
    //class' Constructor
    public Mobil(String merek, String model, String warna){
        this.merek = merek;
        this.model = model;
        this.warna = warna;
        this.kelajuan = 0;
    }
}
```

Constructor di atas akan dijalankan jika objek mobil dibuat

```
Mobil mobil = new Mobil("Daihatsu", "Xenia", "Putih");
```

Setter & Getter

Setter adalah *method* yang dipakai suatu objek untuk mengubah *value* dari variabel yang dimiliki sebuah objek, sedangkan ***getter*** adalah *method* yang dipakai untuk mengambil *value* dari sebuah variabel yang dimiliki objek.

Setter dan *getter* digunakan bersamaan terhadap variabel yang disembunyikan oleh objek, biasanya dengan memakai tipe **private** pada atribut tersebut (Contohnya semua variabel pada kelas **Mobil** di atas). Hal ini dilakukan untuk menghindari akses secara langsung dari kelas lain yang bisa menyebabkan kebocoran dan perubahan terhadap data yang disimpan oleh suatu objek

Contoh pengaplikasian *setter* dan *getter*:

```
    //setter method
    public void setKelajuan(int laju){
        this.kelajuan = laju;
    }

    //getter method
    public double getKelajuan(){
        return this.kelajuan;
    }
```

Contoh penggunaan *setter* dan *getter*:

```
Mobil mobil = new Mobil("Daihatsu", "Xenia", "Putih");
System.out.println(mobil.getKelajuan());
mobil.setKelajuan(40);
System.out.println(mobil.getKelajuan());
```

Output yang dihasilkan:

```
0
40
```

toString()

toString adalah *method* yang digunakan untuk mengatur representasi suatu objek ke dalam String. Jika dalam suatu *class* tidak terdapat method **toString()**, maka saat melakukan print, yang muncul di output program adalah nilai kode hash dari objek tersebut

```
Mobil mobil = new Mobil("Daihatsu", "Xenia", "Putih");
System.out.println(mobil);
```

Output:

```
Mobil@d716361
```

Dengan *method* **toString()** kita dapat mengukur representasi String dari objek tersebut. Contohnya dengan menambahkan *method* **toString()** dalam *class* **Mobil**, seperti di bawah ini:

```
class Mobil{

   //......

   //toString method
   public String toString(){
      return "Mobil " + merek + " " + model + " berwarna " + warna;
   }
}
```

Output ketika kita melakukan print:

```
Mobil Daihatsu Xenia berwarna Putih
```

Static Modifier

Static adalah salah satu non-access modifier dimana sebuah variabel atau fungsi tidak terikat dengan objek, melainkan class itu sendiri.

Method Static & non-Static

Method **static** merupakan suatu method yang tidak perlu instansiasi objek untuk menggunakannya. Fungsi ini melekat ke suatu class sehingga untuk memanggilnya bisa dilakukan dengan **<NamaClass>.namaMethod()**. Sedangkan fungsi **non-static** adalah fungsi

yang membutuhkan suatu objek dari suatu class yang memiliki fungsi tersebut untuk menggunakannya.

```
public class Mobil{
	public static void main(String[] args){
		Mobil mobil = new Mobil();
		System.out.println(mobil.maju()); \\Maju
		System.out.println(Mobil.jalan()); \\Jalan
	}
	public static String jalan(){
		return "Jalan";
	}
	public String maju(){
		return "Maju";
	}
}
```

Variable Static dan non-Static

Sama halnya seperti method, variabel static melekat pada suatu class bukan melekat pada suatu objek pada class tersebut sehingga untuk mengetahui value dari suatu variabel static dapat dilakukan dengan **<NamaClass>.namaVariabel**. Sedangkan variabel non-static akan melekat ke suatu objek dari class tersebut sehingga perlu dibuat objek dari class tersebut untuk menggunakannya.

```
public class Mobil{
	public static String maju = "maju";
	public String jalan = "Jalan";
	public static void main(String[] args){
		Mobil mobil = new Mobil();
		System.out.println(mobil.jalan); \\Jalan
		System.out.println(Mobil.maju); \\maju
	}
}
```

Diagram Kelas

Diagram kelas merupakan salah satu diagram UML (Unified Modelling Language) yang digunakan untuk memodelkan desain program yang berbasis OOP (*Object Oriented Programming*). Diagram tersebut memiliki kapabilitas untuk menggambarkan susunan struktur

suatu class di suatu program. Diagram UML tersusun dari sekumpulan tabel yang berisi keterangan dari suatu kelas dan dihubungkan dengan suatu notasi.

Struktur tabel diagram kelas

NamaClass
Variabel
Fungsi

Contoh diagram kelas

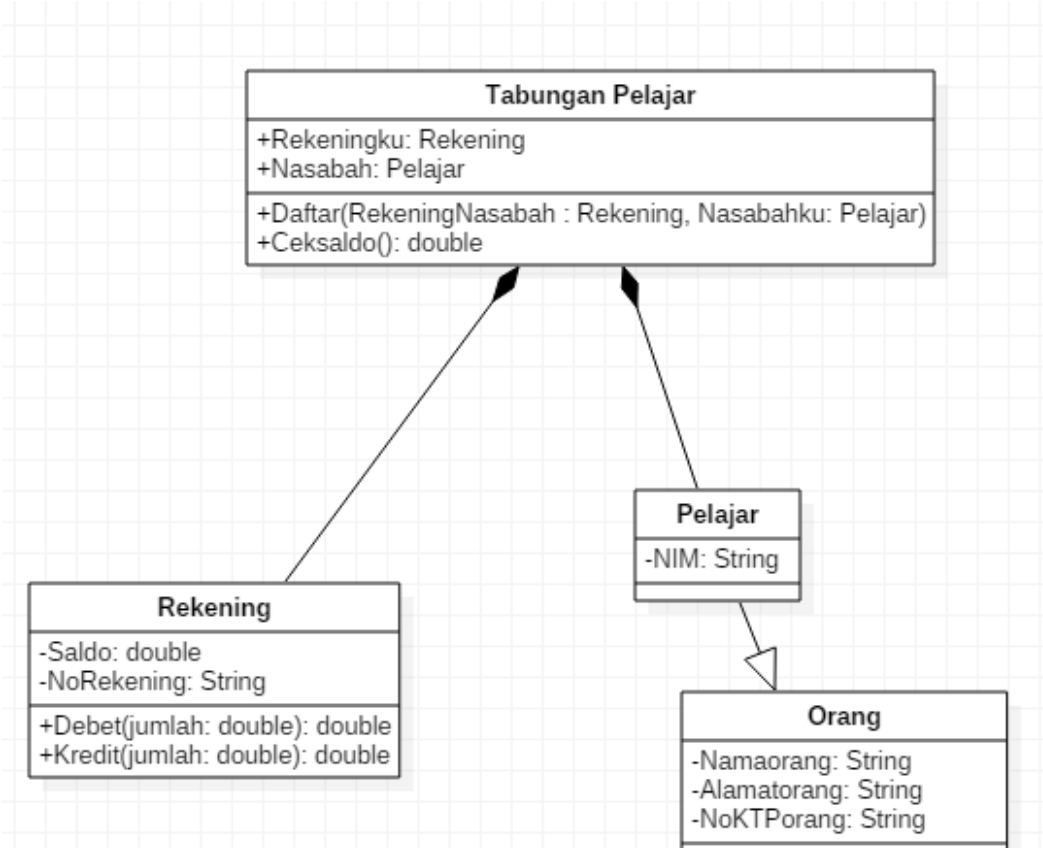

Diagram tersebut menjelaskan bahwa Tabungan Pelajar tidak mungkin ada jika tidak ada Rekening dan Pelajar, Pelajar merupakan turunan dari Orang. Variabel Saldo dan NoRekening di rekening merupakan variabel private. Sementara Rekeningku dan Nasabah di Tabungan Pelajar merupakan variabel public.

Soal Lab 05

Pacilmon Go

Angewomon telah berhasil memecahkan teka -teki Etemon berkat bantuan kalian! Angewomon yang telah bosan menonton Dig*mon Adv*nture memutuskan untuk bermain-main dengan game yang baru ia *download* yaitu Pacilmon Go.

Tetapi, karena ini adalah kali pertama Angewomon memainkan Pacilmon Go, ia tidak terlalu mengerti tentang game tersebut. Angewomon kesulitan untuk mencari nama Pokemon, Pokemon terkuat, dan rata-rata kekuatan Pokemon secara keseluruhan.

Angewomon kemudian mencari Dek Depe untuk meminta diajari Pacilmon Go. Akan tetapi ternyata Dek Depe sedang sibuk mengerjakan hal lain. Mau tidak mau dengan berat hati Angewomon kembali memintamu untuk membantunya memahami game Pacilmon Go.

Yuk bantu Angewomon untuk bermain Pacilmon Go!

“Helping one person might not change the whole world, but it could change the world for one person”
- not Angewomon

AMI, AWP, MAR, REL | S1

SPESIFIKASI

Akan ada tiga template untuk setiap class, yaitu:

1. **Pokemon**

Kelas **Pokemon** memiliki 4 atribut, yaitu **name (String), trainer (Trainer), level (int),** dan **defaultPower (int).** Tugas Anda adalah melengkapi TO DO dalam class ini yang termasuk membuat *constructor* dan *method setter/getter* apabila diperlukan. Constructor akan menerima parameter name, trainer, level dan defaultPower. (**NOTE: Atribut class harus private. Jangan diubah!**)

Class **Pokemon** memiliki beberapa method, yaitu:

- int getBattlePower()

  Method ini akan mengembalikan Battle Power suatu pokemon yaitu dengan **mengalikan level dengan defaultPowernya**

2. **Trainer**

Kelas **Trainer** memiliki 5 atribut, yaitu **allPokemon (Pokemon[ ]), trainersPokemon (Pokemon[ ]), name (String), numOfAllPokemon (int), dan numOfTrainersPokemon (int).** Tugas Anda adalah melengkapi TO DO dalam method di class ini yang termasuk membuat *constructor* dan *method setter/getter* apabila diperlukan. Constructor akan menerima parameter name. Perlu diperhatikan bahwa jumlah seluruh Pokemon tidak akan lebih dari 30 dan seorang trainer tidak boleh memiliki lebih dari 5 Pokemon. (**NOTE: Atribut harus private**)

Class **Trainer** memiliki beberapa method, yaitu :

- void addPokemon(Pokemon pokemon)

  Method ini akan menambahkan pokemon ke daftar allPokemon dan trainersPokemon.

- static Seller getSpecificSeller(String name)

  Method ini merupakan method static yang akan mengembalikan pokemon berdasarkan nama yang sudah terdaftar dengan mengecek isi daftar allPokemon. Jika Pokemon tidak ditemukan, akan memberikan return null.

- static Pokemon getStrongestPokemon()

  Method ini merupakan method static yang akan mengembalikan pokemon terkuat dari seluruh pokemon berdasarkan **battle powernya**. Jika ada satu atau lebih Pokemon yang memiliki battle power terkuat yang sama, kembalikan **salah satu** dari Pokemon tersebut.

- static int getTotalBattlePower()

  Method ini merupakan method static yang akan **mengembalikan total battle power seluruh Pokemon** yang ada.

- static double getAverageBattlePower()

  Method ini merupakan method static yang akan **mengembalikan rata-rata battle power seluruh Pokemon (Total Batttle Power / Jumlah Pokemon).**

NOTE : Tidak diperbolehkan mengubah atau menghilangkan “static” dari static attribute dan static method.

3. Simulator

Untuk memeriksa kebenaran program Anda, silahkan jalankan class Simulator. Namun sebelum itu lengkapi dulu //TO DO yang ada. Tersedia template untuk menerima input user. Format output harus sama dengan contoh.

Contoh input dan output program ketika dijalankan:

```
Daftar pokemon milik Ash:
Sceptile saat ini berada di level 1 memiliki battle power sebanyak
100 dan dimiliki oleh Ash
Pikachu saat ini berada di level 2 memiliki battle power sebanyak
400 dan dimiliki oleh Ash
Charizard saat ini berada di level 3 memiliki battle power
sebanyak 450 dan dimiliki oleh Ash

Daftar pokemon milik May:
Eeve saat ini berada di level 1 memiliki battle power sebanyak 75
dan dimiliki oleh May
Squirtle saat ini berada di level 1 memiliki battle power sebanyak
90 dan dimiliki oleh May
Skitty saat ini berada di level 1 memiliki battle power sebanyak
85 dan dimiliki oleh May
Venusaur saat ini berada di level 3 memiliki battle power sebanyak
240 dan dimiliki oleh May

Daftar pokemon milik James:
Numel saat ini berada di level 1 memiliki battle power sebanyak 90
dan dimiliki oleh James
Victreebell saat ini berada di level 2 memiliki battle power
sebanyak 180 dan dimiliki oleh James

---------------------------------------
---------------------------------------
Selamat datang di database Pacilmon

1. Cari Pokemon berdasarkan nama
2. Cari Pokemon terkuat
3. Lihat rata-rata kekuatan Pokemon
4. Exit
Silahkan masukkan perintah yang ingin dijalankan: 1
Masukkan nama pokemon yang ingin dicari: Pikachu
Pikachu saat ini berada di level 2 memiliki battle power sebanyak
400 dan dimiliki oleh Ash

1. Cari Pokemon berdasarkan nama
2. Cari Pokemon terkuat
3. Lihat rata-rata kekuatan Pokemon
```

```
4. Exit
Silahkan masukkan perintah yang ingin dijalankan: 1
Masukkan nama pokemon yang ingin dicari: Paijomon
Pokemon tidak ditemukan

1. Cari Pokemon berdasarkan nama
2. Cari Pokemon terkuat
3. Lihat rata-rata kekuatan Pokemon
4. Exit
Silahkan masukkan perintah yang ingin dijalankan: 2
Charizard saat ini berada di level 3 memiliki battle power
sebanyak 450 dan dimiliki oleh Ash

1. Cari Pokemon berdasarkan nama
2. Cari Pokemon terkuat
3. Lihat rata-rata kekuatan Pokemon
4. Exit
Silahkan masukkan perintah yang ingin dijalankan: 3
Rata-rata battle power para pokemon adalah 190.0

1. Cari Pokemon berdasarkan nama
2. Cari Pokemon terkuat
3. Lihat rata-rata kekuatan Pokemon
4. Exit
Silahkan masukkan perintah yang ingin dijalankan: 4
Terima kasih sudah menggunakan database Pacilmon
---------------------------------------
---------------------------------------
```

Catatan:

1. Jangan lupa berdoa sebelum mengerjakan.
2. Sangat disarankan menggunakan template 😊
3. Mengubah hal yang sudah diperingatkan adalah kesalahan yang **fatal**, akan ada pengurangan nilai yang signifikan untuk hal itu. (Mengubah private attribute menjadi tidak private, mengubah atau menghilangkan static attribute dan static method)
4. Mencari Pokemon, menghitung rata-rata battle power, dan mencari pokemon terkuat **HARUS** dipanggil dari **static method** di dalam class **Trainer.**
5. Terdapat typo pada template constructor Pokemon. Seharusnya parameternya berupa defaultPower, bukan battlePower

Komponen Penilaian

- 20% Implementasi *class* Pokemon
- 40% Implementasi *class* Trainer
- 30% Implementasi *class* Simulator
- 10% Kerapian kode dan dokumentasi

Keterangan Revisi

a. Revisi 1
   i. … defaultBattlePower → ...defaultPower
   ii. … 5. Terdapat typo pada template
   iii. … Lab04 → Lab05

Kumpulkan berkas .java yang telah di-zip dengan format penamaan seperti berikut.
`Lab05_[Kelas]_[KodeAsdos]_[NPM]_[NamaLengkap].zip`
Contoh:
`Lab05_A_LN_1234567890_DekDepe.zip`

Selamat mengerjakan! 😊