Dasar-Dasar Pemrograman 2

Tutorial 5 Kelas C & F

OOP
Selasa, 12 Maret 2019 - 16:00 WIB

Object Oriented Programming

Dalam paradigma *Object Oriented Programming*, kita memodelkan objek dunia nyata sebagai sebuah *class* di dalam program. Objek tersebut memiliki *variable*, *constructor*, dan *method*. Sebagai contoh, kita akan membuat suatu *class* yang memodelkan objek Mobil di dunia nyata. Mobil dapat dilihat sebagai sebuah objek yang memiliki merk, model, warna, dan kelajuan (*Instance Variable*). Kelajuan mobil dapat dipercepat (akselerasi) ataupun diperlambat (deselerasi) (*Method*). Berikut *class* yang dibuat:

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

    public Mobil(String merk, String model, String warna) {
        this.merk = merk;
        this.model = model;
        this.warna = warna;
        this.kelajuan = 0;
    }

    public void akselerasi(double perubahan) {
        this.kelajuan += perubahan;
    }
```

```
    public void deselerasi(double perubahan) {
        if (kelajuan > 10) {
            this.kelajuan -= perubahan;
        } else {
            this.kelajuan = 0;
        }
    }
}
```

Catatan:
penggunaan *this* pada kode diatas (Java) memiliki ekuivalensi seperti penggunaan *self* pada kode Python.

Constructor

Constructor adalah method untuk membuat instansiasi objek dari class. Constructor adalah method khusus yang akan dieksekusi pada saat pembuatan objek (instance). Biasanya method ini berisi inisialisasi variable untuk objek.

```
public class Mobil {
    //...

    public Mobil(String merk, String model, String warna) {
        this.merk = merk;
        this.model = model;
        this.warna = warna;
        this.kelajuan = 0;
    }
}
```

Constructor di atas akan dijalankan jika objek mobil dibuat.

```
Mobil mobil = new Mobil("Yotayo", "Bus", "Blue")
```

Setter & Getter

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

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

Contoh pengaplikasian Setter & Getter :

```
public class Mobil {
    //...

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

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

Contoh penggunaan setter & getter :

```
// Masih menggunakan class Mobil diatas

Mobil tayo = new Mobil("Toyota", "Avanza", "Silver");
System.out.println(tayo.getKelajuan());
tayo.setKelajuan(40);
System.out.println(tayo.getKelajuan());
```

Menghasilkan output :

```
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 adalah nilai kode hash dari objek tersebut.

```
Mobil s = new Mobil("Ferrari", “458”, “merah”);
System.out.println(s);
```

```
Output:
Mobil@1eed786
```

Dengan method toString() kita dapat mengatur representasi string dari objek tersebut. Contohnya dengan menambahkan method toString() dalam class Mobil, seperti ini:

```
public class Mobil {
    //...

    public String toString() {
        return "Mobil " + merk + " " + model + " bewarna " + warna;
    }
}
```

Maka ketika melakukan print:

```
Mobil s = new Mobil("Ferrari", "458", "merah");
System.out.println(s);
```

```
Output:
Mobil Ferrari 458 berwarna merah
```

Arena of Blood

src : http://elementrpg.wdfiles.com/local--files/wiki:ritual-of-blood-and-sand/arena.jpg

“Cukup sudah dengan kenangan itu, aku sudah tidak memiliki cita cita sebagai arkeolog, mari kita bicarakan masa depan” - Dek Depi.

Dek Depi tidak ingin mengingat-ngingat lagi masa lalunya, mengingat-ngingat masa lalunya dengan kekasih tercinta, buku-buku arkeologi yang menemaninya setiap malam, pada akhirnya memberikan kesan pahit padanya karena Dek Depi tidak berhasil meraih apa yang ia inginkan waktu itu.

Sekarang, Dek Depi mengalihkan pikirannya dengan hal-hal lain yang dapat ia capai di masa depan. Belum lama ini, Dek Depi membaca sebuah buku cerita tentang gladiator. Dek Depi memutuskan, untuk pencapaiannya kali ini, dia akan membuat sebuah program berdasarkan buku yang ia baca tersebut. Program yang akan dibuat oleh Dek Depi merupakan program simulasi pertarungan gladiator, dimana beberapa gladiator akan dipertemukan di suatu arena, kemudian mereka akan saling bertarung. Seperti biasa, Dek Depi ingin kamu membantunya untuk membuat permainan tersebut, tentunya kamu mau membantunya kan ?

Spesifikasi Program :

*Kamu wajib menggunakan template yang disediakan, method dan attribut di bawah ini harus diimplementasikan pada template. Kamu juga harus mengimplementasikan constructor untuk setiap kelas. Selain itu, template sudah berisi method lainnya yang mungkin dapat kamu manfaatkan dalam menyelesaikan tugas ini.

CLASS:

- Gladiator
- Arena

SPESIFIKASI CLASS GLADIATOR:

- Setiap **Gladiator** memiliki attribut:
    - String name
    - int hp
    - int damage
    - String faction
- Setiap **Gladiator** memiliki method:
    - void attack (Gladiator gladiator)
        - Tidak terjadi apa-apa jika ***Gladiator*** yang melakukan attack kepada ***Gladiator*** lainnya berada di ***Faction*** yang sama.
        - Tidak dapat dilakukan oleh **Gladiator** yang memiliki hp = 0
        - Selain itu, ***Gladiator*** yang menjadi target **Attack** akan berkurang ***HP*** nya sebesar ***Damage Gladiator*** yang melakukan ***Attack.***
    - void setHP (int hp)
        - Method untuk mengatur hp dari Gladiator, perhatikan hp **tidak boleh negatif.**
    - String toString()
        - Mengembalikan String dengan format:
          Nama: <Gladiator_Name>
          HP: <Gladiator_HP>
          Damage: <Gladiator_Damage>
          Faction: <Gladiator_Faction>

SPESIFIKASI CLASS Arena:

- Setiap **Arena** memiliki attribut:
    - ArrayList<Gladiator> gladiators
    - int id
- Setiap **Arena** memiliki method:
    - void doFight()
        - Setiap **Gladiator** yang ada pada **Arena** tersebut akan melakukan **Attack** kepada semua **Gladiator** lain yang berbeda **Faction** pada **Arena** tersebut.
        - Urutan **Gladiator** yang melakukan **Attack** diurutkan berdasarkan urutan datangnya **Gladiator** ke **Arena** tersebut, dimulai dari **Gladiator** yang pertama kali tiba.
    - void add(Gladiator Gladiator)
        - Menambahkan **Gladiator** ke **Arena** pada urutan terakhir tiba.

Jalankan File Simulator.java yang disediakan pada Template untuk mengecek kebenaran program

Output :

*Selain membandingkan secara manual, Anda dapat menggunakan diffchecker untuk mengecek output, misalnya menggunakan https://www.diffchecker.com/diff

```
<<Arena 1>>
Name: Achilles
HP: 52
Damage: 50
Faction: Tihr

Name: Aegeus
HP: 0
Damage: 11
Faction: Uxhal

Name: Ajax
HP: 0
Damage: 19
Faction: Uxhal

----------------------------------------
<<Arena 2>>
Name: Aeneas
HP: 7
Damage: 32
Faction: Sargoth

Name: Aesop
HP: 0
Damage: 24
Faction: Dhirim

Name: Arastoo
HP: 4
Damage: 19
Faction: Uxhal

----------------------------------------
<<Arena 3>>
Name: Alastair
HP: 37
Damage: 50
Faction: Dhirim
```

```
Name: Alejandro
HP: 17
Damage: 13
Faction: Uxhal

Name: Antreas
HP: 0
Damage: 14
Faction: Uxhal

----------------------------------------
```

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi Class Gladiator	Melakukan implementasi pada bagian-bagian yang diminta dalam *class Gladiator*, dengan menggunakan paradigma *object oriented*	25%
Implementasi Class Arena	Melakukan implementasi pada bagian-bagian yang diminta dalam *class Arena*, dengan menggunakan paradigma *object oriented*	25%
Kebenaran Program	Kesesuaian Output program dengan hasil yang diinginkan	35%
Kerapian Kode	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, terstruktur, dan disertakan oleh dokumentasi secukupnya	15%

Deadline :Selasa, 12 Maret 2019

Pukul **17:40 WIB**

Format Pengumpulan :

Kumpulkan hasil zip template yang sudah diimplementasikan kode Anda di slot pengumpulan yang telah disediakan di SCeLE dengan format :

[Kode Asdos]_[Nama]_[Kelas]_[NPM]_Lab[X].zip

Contoh :

TES_DemoSuremo_C_1234567891_Lab4.zip

Acknowledged Lecturers :

- Fariz Darari, S.Kom, M.Sc., Ph.D.

Selamat Bekerja !