Dasar-Dasar Pemrograman 2

Tutorial 5 Kelas B & E

OOP
Kamis, 14 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
```

Race of Champion

https://www.playstation.com/en-us/games/motogp-psp/

Hari ini adalah hari yang paling ditunggu-tunggu oleh Dek Depi karena hari ini akan ada babak kualifikasi MotoGP ! Dek Depi sangat mengagumi Marc Marquez yang selalu saja tampil memukau di ajang balapan MotoGP. Saat ini waktu menunjukkan pukul empat sore, dan perlombaan baru akan dimulai pada pukul sembilan malam, namun Dek Depi sudah sangat tidak sabar untuk mengetahui hasil dari pertandingan MotoGP kali ini.

“Duh, aku sangat berharap Marc Marquez dapat memenangkan pertandingan kali ini.”

Aha! Dek Depi memiliki ide yang cemerlang. Dek Depi ingin mencoba untuk memprediksi hasil pertandingan MotoGP kali ini dengan mencari nama pembalap yang akan bertanding pada malam hari ini dan melihat spesifikasi motor masing-masing pembalap. Dengan data yang ia kumpulkan tersebut, Dek Depi ingin memprediksi juara-juara MotoGP pada malam hari ini. Seperti biasa, Dek Depi ingin kamu untuk membantunya membuat suatu program yang dapat mengeluarkan output berupa prediksi nama-nama pemenang untuk lomba MotoGP pada malam hari ini. Tentunya kamu juga penasaran kan dengan hasil lomba pada malam hari ini?

Spesifikasi Program :

Kamu wajib menggunakan template yang disediakan, method dan atribut 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:

- Motorcycle
- Circuit

SPESIFIKASI CLASS MOTORCYCLE:

- Setiap **Motorcycle** memiliki atribut:
    - String name
    - int speed
    - int fuel
    - int maxFuel
- Setiap **Motorcycle** memiliki method:
    - void setFuel (int fuel)
        - Method untuk mengatur bensin dari Motorcycle dengan syarat bensin suatu **Motorcycle** tidak boleh negatif maupun melebihi maxFuel.
          Hint : Jika bensin suatu **Motorcycle** bernilai negatif, maka nilai bensin harus diubah menjadi 0. Begitu pula jika bensin melebihi batas, akan dibatasi sebanyak maxFuel.
    - void toString()
        - Mengembalikan String dengan format:
          Name: <Motorcycle_Name>
          Speed: <Motorcycle_Speed>
          Fuel: <Motorcycle_Fuel>
          Max Fuel: <Motorcycle_maxFuel>

SPESIFIKASI CLASS CIRCUIT:

- Setiap **Circuit** memiliki atribut:
    - ArrayList<Motorcycle> motor
    - String circuitName
    - int circuitLength
    - Int raceCount -> suatu atribut yang menghitung sudah berapa kali pertandingan dilakukan.
- Setiap **Circuit** memiliki method:
    - void refuel(int amount)
        - Setiap **Motorcycle** yang ada pada **Circuit** tersebut melakukan pengisian bensin sesuai dengan jumlah bensin yang menjadi parameter.
    - void addMotorcycle(Motorcycle motorcycle)
        - Menambahkan **Motorcycle** ke **Circuit**.
    - void doRace()
        - Semua **Motorcycle** yang ada pada **Circuit** tersebut bertanding secara sekaligus.

- Setiap kali doRace dilakukan, bensin setiap motor akan berkurang sebesar panjang **Circuit**.
- **Motorcycle** yang kehabisan bensin saat pertandingan tidak dapat mengakhiri pertandingan, yang berarti tidak dapat menjadi juara walaupun memiliki kecepatan yang paling tinggi.
- Mencetak pembalap tercepat yang dapat mengakhiri pertandingan, dengan urutan **Motorcycle** yang tiba di garis finish diurutkan berdasarkan kecepatan. Apabila ada **Motorcycle** yang seri, maka program akan mencetak pembalap yang pertama kali dimasukkan ke **Circuit**.
  Hint: Untuk mencetak pemenang, gunakan method yang sudah ada pada **Motorcycle**

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

Jangan lupa untuk mengompilasi ulang semua file setiap ada file yang diubah. Untuk mengompilasi seluruh file, dapat menggunakan `javac *.java`

Output :

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

```
<<Race 1 | Losail International Circuit>>
[ Winner ]
Name: Andrea Dovizioso
Speed: 30
Fuel: 15
Max Fuel: 1000

---------------------------------------
<<Race 2 | Losail International Circuit>>
[ Winner ]
Name: Marc Márquez
Speed: 25
Fuel: 0
Max Fuel: 1000

---------------------------------------
<<Race 1 | TT Circuit Assen>>
[ Winner ]
Name: Maverick Viñales
Speed: 40
Fuel: 50
Max Fuel: 150

---------------------------------------
<<Race 2 | TT Circuit Assen>>
```

```
[ Winner ]
Name: Joan Mir
Speed: 25
Fuel: 25
Max Fuel: 225

---------------------------------------
<<Race 3 | TT Circuit Assen>>
[ Winner ]
No one can complete this race

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

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi Class Motorcycle	Melakukan implementasi pada bagian-bagian yang diminta dalam *class Motorcycle*, dengan menggunakan paradigma *object oriented*	25%
Implementasi Class Circuit	Melakukan implementasi pada bagian-bagian yang diminta dalam *class Circuit*, 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 :Kamis, 14 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_Lab5.zip

Acknowledged Lecturers :

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

Selamat Bekerja !