tDasar-Dasar Pemrograman 2

Tutorial 08 Kelas C & F

Abstract Class & Interface
Selasa, 16 April 2019 - 16:00 WIB

Pada tutorial/lab sebelumnya, kalian telah mengetahui bagaimana cara memodelkan sebuah masalah *object-oriented* ke dalam bentuk *hierarchical classes* dengan konsep *inheritance* dan variabel yang dapat mereferensikan dirinya ke berbagai objek lain dengan konsep *polymorphism*. Tutorial/lab ini akan membahas pemodelan masalah *object-oriented* dengan *abstract class* dan *interface*.

Abstract Class dapat didefinisikan sebagai sebuah *class* yang dideklarasikan *abstract*. *Abstract class* dapat memiliki variabel non-*static* dan non-*final.* Sebuah *abstract class* dapat memiliki *public*, *protected*, dan *private method(s)* yang telah diimplementasikan. Sebuah *abstract class* dapat memiliki minimal satu *abstract method* (atau tidak sama sekali).

```
abstract class CaffeineBeverage {
    // Instance variable 1
    // Instance variable 2

    // Abstract method 1
    // dst...
}
```

Gambar 8.1: Contoh abstract class

Abstract method sendiri adalah sebuah *method* yang dideklarasikan tanpa implementasi (tanpa kurung kurawal, dan diakhiri dengan titik koma).

```
protected abstract double cost();
```

Gambar 8.2: Contoh abstract method

Abstract class sendiri tidak dapat diinstansiasi, tapi dapat dibuat *subclass*-nya. Jika sebuah *abstract class* dibuat *subclass*, maka *subclass* dapat memberikan implementasi dari semua *abstract method* yang ada.

```java
abstract class CaffeineBeverage {
    protected String description;

    protected abstract double cost();
}

class Coffee extends CaffeineBeverage {
    public Coffee() {
        this.description = "Coffee";
    }

    @Override
    public double cost() {
        return 2.0;
    }

}

class Tea extends CaffeineBeverage {
    public Tea() {
        this.description = "Tea";
    }

    @Override
    public double cost() {
        return 1.0;
    }

}
```

Gambar 8.3: Contoh lengkap *abstract class* dengan *subclass* yang mengimplementasikan *abstract method*

Catatan: jika subclass dari abstract class tidak mengimplementasikan semua *abstract method* yang ada di abstract class, maka subclass tersebut harus dideklarasikan abstract juga.

Interface dapat didefinisikan sebagai sebuah *reference type*, mirip seperti sebuah *class*, tapi hanya dapat memiliki *constant* dan *method signatures*. *Constants* dalam interface, tanpa perlu dideklarasikan, akan secara otomatis menjadi *public*, *static*, dan *final*. Sedangkan method signatures, tanpa perlu dideklarasikan, akan secara otomatis menjadi *public* dan *abstract*.

```java
interface Flyable {
    // Declare public static final variables

    // Declare public and abstract methods
}
```

Gambar 8.4: Contoh sebuah *interface*

Sebuah *interface* tidak dapat diinstansiasi, tapi hanya dapat di-*implement* oleh class atau di-*extend* dengan *interface* lain. *Interface* juga dapat dikatakan sebagai sebuah kontrak, dengan kata lain, jika sebuah *class* meng-*implement* sebuah *interface*, maka *class* tersebut harus mengimplementasikan method yang ada di *interface* tersebut.

```
interface Flyable {
    void fly();
}

class Airplane implements Flyable {
    public void fly() {
        System.out.println("Fly high in the air with dem powerful turbines.");
    }
}

class Bird implements Flyable {
    public void fly() {
        System.out.println("Fly high in the air with dem powerful wings.");
    }
}
```

Gambar 8.5: Contoh lengkap *interface* dengan *class* yang mengimplementasikan *method* yang ada di *interface*

Javari Park Season 2

Javari Park merupakan sebuah kebun binatang yang berada di Earth-2019 di mana semua binatang yang berada di sana dapat melakukan atraksi pada siang hari dan menulis kode pada malam hari.

Spesifikasi Program :

Secara umum, setiap binatang yang terdapat di Javari Park memiliki nama (**name**) dan poin (**point**) yang menandakan seberapa baik performa mereka. Mereka dapat mengeluarkan suara, spesifik terhadap spesies mereka. **Lion** dapat mengeluarkan auman **"Hahummm…"**, **Eagle** dapat mengeluarkan suara **"Kwakkk…"**, sementara **Snake** dapat mendesis **"Sssss…."**.

Setiap harinya, para binatang di Javari Park diharapkan untuk melaksanakan atraksi dengan baik. Jika mereka berhasil, maka **poin** mereka akan **bertambah**, namun karena setiap jenis binatang mendapat atraksi yang berbeda, banyak poin yang ditambahkan juga ikut berbeda. **Lion** akan mendapat pertambahan **20** poin, **Eagle** sebanyak **15** poin, dan **Snake** sebanyak **10** poin. Setiap kali ada binatang yang melakukan atraksi, sistem akan mencetak sebuah info dengan format "***[Nama]*** *melakukan atraksi! Poin bertambah!*"

Terkadang, binatang yang merasa bosan dengan melakukan atraksi setiap harinya akan bermalas-malasan *(jangan ditiru!)*. Setiap kali mereka bermalas-malasan, mereka akan kehilangan sebanyak poin yang mereka dapatkan seandainya mereka melakukan atraksi dengan baik. Setiap kali ada binatang yang bermalas-masalan, sistem akan mencetak sebuah info dengan format *"**[Nama]** bermalas-malasan! Poin berkurang!"*. Pihak manajemen Javari Park telah menetapkan bahwa setiap binatang **tidak dapat memiliki poin yang minus dan paling sedikit 0 poin**.

Setiap akhir pekan, sistem akan melakukan pengurutan (*ranking*) setiap binatang yang terdaftar di Javari Park berdasarkan **poin** yang mereka miliki karena pada awalnya, belum ada standar yang ditetapkan dalam perbandingan antar dua binatang. Jika ada dua binatang yang dibandingkan, maka binatang yang poinnya lebih besar akan berada di posisi "lebih atas" dibanding binatang kedua.

Kamu sebagai *intern* di Javari Park merasa tertantang untuk memodelkan situasi-situasi di atas dalam pemrograman berbasis *object*. Bantulah pihak Javari Park!

INTERFACE:

- **Attraction**

CLASS:

- **ParkAnimal (Abstract)**
- **Lion**
- **Eagle**
- **Snake**

SPESIFIKASI INTERFACE **Attraction**:

- Method:
    - doAttraction()

SPESIFIKASI CLASS **ParkAnimal**:

- Attribute:
  - name
  - point
- Method:
  - getName()
  - getPoint()
  - setPoint()
  - makeSound()
  - slackOff()
  - compareTo()

Class **Lion**, **Eagle**, dan **Snake** mewarisi semua attribute dan method yang didefinisikan pada class **ParkAnimal**.

Note: gunakan jenis method (abstract atau non-abstract) dan *access modifier* yang sesuai dalam mendefinisikan method-method di atas sesuai dengan prinsip abstraksi dan interface.
Kamu juga bebas mengimplementasikan class-class dan interface di atas (mengimplementasikan lebih dari yang diminta) **asalkan** sesuai dengan ketentuan-ketentuan yang telah tertulis di atas dan **tidak mengubah** isi dari class **JavariParkSimulator**.

Contoh Output JavariParkSimulator.java yang Diharapkan:

```
Hahummm...
Sssss....
Kwakkk...
[Leo] melakukan atraksi! Poin bertambah sebanyak 20!
[Solid Snake] melakukan atraksi! Poin bertambah sebanyak 10!
[Taka] melakukan atraksi! Poin bertambah sebanyak 15!

[SNAKE] Solid Snake - 10
[EAGLE] Taka - 15
[LION] Leo - 20
[Taka] melakukan atraksi! Poin bertambah sebanyak 15!
[Taka] bermalas-malasan! Poin berkurang sebanyak 15!
[Leo] bermalas-malasan! Poin berkurang sebanyak 20!
[Leo] bermalas-malasan! Poin berkurang sebanyak 20!
[Taka] melakukan atraksi! Poin bertambah sebanyak 15!
[Leo] melakukan atraksi! Poin bertambah sebanyak 20!
[Solid Snake] bermalas-malasan! Poin berkurang sebanyak 10!
[Taka] bermalas-malasan! Poin berkurang sebanyak 15!
[Leo] melakukan atraksi! Poin bertambah sebanyak 20!
[Taka] melakukan atraksi! Poin bertambah sebanyak 15!
[Leo] bermalas-malasan! Poin berkurang sebanyak 20!

[SNAKE] Solid Snake - 0
[LION] Leo - 20
[EAGLE] Taka - 30
```

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi Abstract Class ParkAnimal dan Interface Attraction	Mengimplementasikan abstract class ParkAnimal dan Interface Attraction sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	20 %
Implementasi Concrete Class Lion	Mengimplementasikan class Lion sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	15 %
Implementasi Concrete Class Eagle	Mengimplementasikan class Eagle sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	15 %
Implementasi Concrete Class Snake	Mengimplementasikan class Eagle sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	15 %
Ketepatan Program	Ketepatan output ketika JavariParkSimulator dijalankan (pemenuhan requirement client).	25 %
Kerapian	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, terstruktur, dan disertakan oleh dokumentasi secukupnya (bila diperlukan).	10 %

Deadline :

Selasa, 16 April 2019
Pukul **17:40 WIB**

Format Pengumpulan :

Satukan semua class-class yang dibutuhkan pada zip archive dan kumpulkan di slot pengumpulan yang telah disediakan di SCeLE dengan format :

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

Contoh :

TES_DemoSuremo_X_1706040151_Lab8.zip

Acknowledged Lecturers :

- Fariz Darari, S.Kom, M.Sc., Ph.D.
- Raja Oktovin Parhasian Damanik S.Kom., M.Sc.

Authors :

- FAA
- LA
- MDN
- WH