Dasar-Dasar Pemrograman 2

Tutorial 08 Kelas B, D, & E

Abstract Class & Interface

Kamis, 18 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.

```
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*.

```
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();
}

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

public class Bird extends 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*

The Avenged : 7 Kali lipat

[Major Spoiler alert !]

Pada pertempuran sebelumnya yaitu “pertempuran tanpa batas”, sang musuh utama *Thanoshii* berhasil mengalahkan para pahlawan yang tergabung dalam kesatuan The Avenged. Setelah melewati satu tahun masa berkabung, para pahlawan kembali berkumpul untuk membahas strategi berikutnya melawan musuh bebuyutan mereka. Dalam pertemuan tersebut para pahlawan sepakat untuk melakukan perekrutan baru demi membalas kekalahan mereka 7 kali lipat kepada *Thanoshii*.

Secara umum pahlawan yang akan direkrut oleh tim Avenged memiliki 3 kekuatan, yaitu kemampuan **terbang, kekuatan super,** dan kemampuan **menembak laser** dan setiap pahlawan yang mendaftar dapat memiliki 1 atau lebih kekuatan dalam dirinya. Untuk mendaftarkan dirinya, para pahlawan baru harus mendaftarkan **nama** beserta angka **kekuatan dirinya,** dan **daftar kekuatannya** kepada sistem pintar milik Avenged. Setelah semua data masuk, secara otomatis sistem akan mengurutkan setiap pahlawan yang mendaftar berdasarkan angka **kekuatan dirinya** dengan pahlawan yang memiliki angka **paling besar berada di urutan pertama**. Kemudian sang sistem pintar akan memanggil nama-nama seluruh pahlawan dimulai dari urutan pertama. Setiap pahlawan yang dipanggil namanya akan menunjukan **Identitasnya** dan menjawab dengan format “*Siap ! saya **[nama]** siap menjadi pahlawan dengan kekuatan **[nama kekuatan 1, nama kekuatan 2, …, nama kekuatan n]** !* ”.

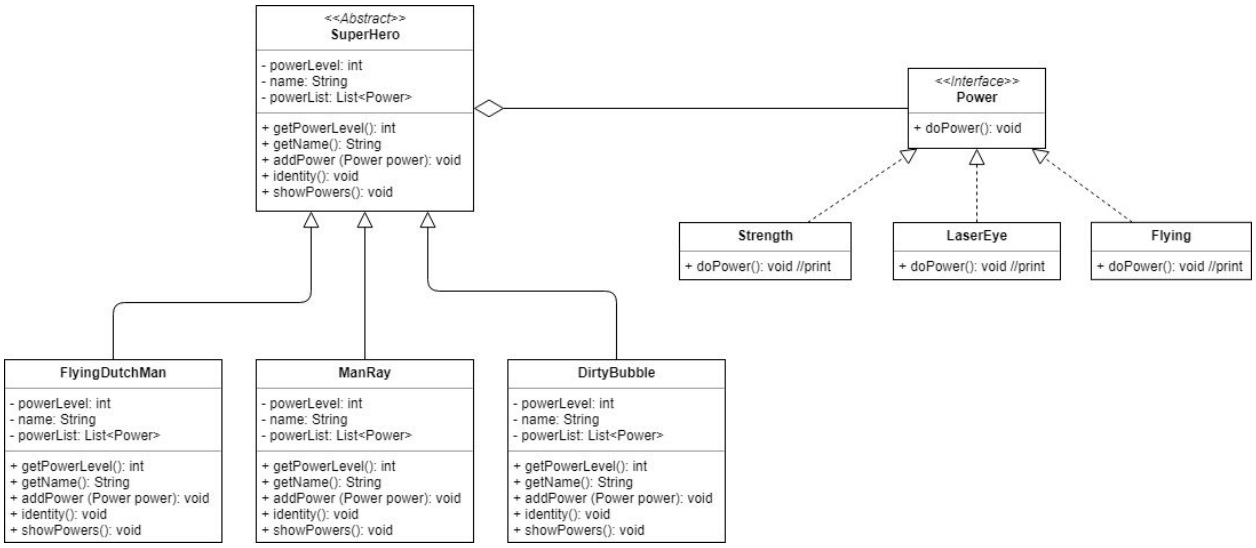
Setelah melihat kondisi yang ada, ternyata para pahlawan yang mendaftar tersebut dapat **digolongkan menjadi tiga.** Yaitu pahlawan **FlyingDutchMan, ManRay,** dan **DirtyBubble.** Setiap golongan pahlawan memiliki kekuatan yang berbeda-beda. Sebagai gambaran, **FlyingDutchMan** adalah seorang pahlawan yang bisa **Terbang** dan dapat **menembakan Laser** . Kemudian **ManRay** adalah seorang pahlawan yang dapat **menembak Laser** dan juga memiliki **kekuatan super.** Golongan terakhir **DirtyBubble** adalah jenis pahlawan yang dapat **Terbang** dan memiliki **Kekuatan Super**.

.

Waktu sudah kurang dari satu minggu hingga pertempuran terakhir, namun sistem pintar Avenged sedang mengalami masalah. Kamu sebagai *intern* yang sedang berada di Avenged kemudian diminta untuk membuat sistem penggantinya demi keberlangsungan hidup umat manusia.

Spesifikasi Program :

Class Diagram



Berikut adalah interface dan class-class yang harus kalian buat.

INTERFACE:

- **Power**

CLASS:

- **SuperHero (Abstract)**
- **FlyingDutchMan**
- **ManRay**
- **DirtyBubble**
- **Strength**
- **LaserEye**
- **Flying**

SPESIFIKASI INTERFACE **Power**:

- Method:
    - doPower()

SPESIFIKASI CLASS **Superhero**:

- Atribut:
    - powerLevel
    - Name
    - powerList

- Method:
    - getPowerLevel()
    - getName()
    - addPower(Power power)
    - identity()
    - showPowers()

Jangan lupa implementasikan Comparable di-class SuperHero agar dapat di *sort* di AvengedSimulator.java

Contoh Output dari AvengedSimulator.java:

```
==============================
It's Gennichiro, the ManRay! It has the power level of 0

.....HEED ME.....
FOR MY NAAAAAAAME IS GENNICHIRO
TIME TO SHOW YOU MY POWERS
SUPERIOR SIGHT, BEHOLD LASER EYE!
WEAKNESS DISGUST ME, BEHOLD SUPER STRENGTH!
==============================

==============================
It's Shirai, the FlyingDutchMan! It has the power level of 225

.....HEED ME.....
FOR MY NAAAAAAAME IS SHIRAI
TIME TO SHOW YOU MY POWERS
EAT DIRT MORTAL, BEHOLD THE POWER OF FLIGHT!
SUPERIOR SIGHT, BEHOLD LASER EYE!
==============================

==============================
It's Gyoubu Masataka Oniwa, the ManRay! It has the power level of 553

.....HEED ME.....
FOR MY NAAAAAAAME IS GYOUBU MASATAKA ONIWA
TIME TO SHOW YOU MY POWERS
SUPERIOR SIGHT, BEHOLD LASER EYE!
WEAKNESS DISGUST ME, BEHOLD SUPER STRENGTH!
==============================
```

```
==============================
It's Arnastria, the DirtyBubble! It has the power level of 666

.....HEED ME.....
FOR MY NAAAAAAAME IS ARNASTRIA
TIME TO SHOW YOU MY POWERS
WEAKNESS DISGUST ME, BEHOLD SUPER STRENGTH!
EAT DIRT MORTAL, BEHOLD THE POWER OF FLIGHT!
==============================

==============================
It's Tatenari, the FlyingDutchMan! It has the power level of 36556

.....HEED ME.....
FOR MY NAAAAAAAME IS TATENARI
TIME TO SHOW YOU MY POWERS
EAT DIRT MORTAL, BEHOLD THE POWER OF FLIGHT!
SUPERIOR SIGHT, BEHOLD LASER EYE!
==============================
```

Komponen Penilaian :

Komponen	Penjelasan	Bobot
Implementasi Abstract Class SuperHero dan Interface Power	Mengimplementasikan abstract class SuperHero dan Interface Power sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	30%
Implementasi Concrete Class Strength	Mengimplementasikan class Strength sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%
Implementasi Concrete Class LaserEye	Mengimplementasikan class LaserEye sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%
Implementasi Concrete Class Flying	Mengimplementasikan class Flying sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%

Implementasi Concrete Class FlyingDutchMan	Mengimplementasikan class FlyingDutchMan sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%
Implementasi Concrete Class ManRay	Mengimplementasikan class ManRay sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%
Implementasi Concrete Class DirtyBubble	Mengimplementasikan class DirtyBubble sesuai dengan ketentuan yang tertulis pada spesifikasi program dan prinsip-prinsip yang telah diajarkan.	10%
Kerapian	Penulisan program mengikuti kaidah dan konvensi yang telah diajarkan. Program ditulis dengan rapi, terstruktur, dan disertakan oleh dokumentasi secukupnya	10 %

Deadline :

Kamis, 18 April 2019
Pukul **17:40 WIB**

Format Pengumpulan :

Ekstrak semua class dan interface yang dibuat dan kumpulkan di slot pengumpulan yang telah disediakan di SCeLE dengan format :

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

Contoh :

JO_JonathanChristopherJakub_C_1706040151_Lab0.zip

Acknowledged Lecturers :

-

Authors :

-