Modul 2 : Class & Object – Class Diagram

2.1 Tujuan

Setelah mengikuti praktikum ini mahasiswa diharapkan dapat:

1. Memahami konsep class & object.
2. Dapat mengimplementasikan konsep class & object pada kasus sederhana.
3. Mampu mentranslasikan class diagram menjadi kode program.

2.2 Dasar Teori

2.2.1 Class & Object

Pada pembuatan perangkat lunak berbasis Object Oriented, terdapat mekanisme untuk merancang perangkat lunak tersebut menggunakan UML (Unified Modeling Language). Salah satu bentuk perancangan dari UML adalah **Class Diagram**. Contoh class diagram diperlihatkan pada gambar di bawah ini.

Mahasiswa
-nim : string -nama : string -tinggi : int -pindahan : bool
+setNim(in nim : string) : void +getNim() : string +setNama(in nama : string) : void +getNama() : string +setTinggi(in tinggi : int) : void +getTinggi() : int +setPindahan(in pindahan : bool) : void +isPindahan() : bool +Mahasiswa(in nim : string, in nama : string, in tinggi : int) +Mahasiswa(in nim : string, in nama : string, in tinggi : int, in pindahan : bool)

Gambar 1: Class Diagram dari Class Mahasiswa

Berdasarkan gambar di atas, terlihat bahwa class diagram merupakan sebuah bujur sangkar dengan 3 tingkatan, dengan konten di masing-masing tingkat sebagai berikut:

a) Tingkat pertama: nama Class

b) Tingkat kedua: nama atribut beserta tipe datanya

c) Tingkat ketiga: nama method beserta keterangannya

2.2.2 Penulisan Atribut

Atribut dituliskan dengan aturan berikut:

visibility name: type-expression= initial-value{ property-string}

Keterangan:

a. Visibility: private (-), protected(#), dan public (+)
b. Name: nama atribut
c. Type-expression: tipe data dari atribut
d. Initial-value: nilai awal atribut (jika ada)
e. Property-string: aturan perancangan lainnya (jika ada)

Jika sebuah atribut bersifat final (merupakan konstanta), maka pada class diagram ditulis dengan huruf kapital. Jika atribut bersifat static, berikan *underline*.

2.3.3 Penulisan Method

Method dituliskan dengan aturan berikut:

visibility name(parameter-list) : return-type-expression{ property-string}

Keterangan:

a. Visibility: private (-), protected(#), dan public (+)
b. Name: nama method
c. Parameter-list: penulisan parameter yang dilewatkan oleh method. Terdapat aturan lain untuk ini.
d. Return -ype-expression: tipe data yang dikembalikan oleh method
e. Property-string: aturan perancangan lainnya (jika ada)

Aturan penulisan parameter-list:

kind name: type-expression= default-value

Keterangan:

a. Kind: jenis parameter dari method. Terdapat 3 jenis parameter: in, out, inout. Khusus Bahasa pemrograman Java, hanya ada tipe “in”.
b. Name: nama parameter yang dilewatkan
c. Type-expression: tipe data parameter yang dilewatkan
d. Default-value: nilai awal parameter (jika ada).

Class diagram “Mahasiswa” di atas memiliki 2 konstruktor. Dengan kata lain, overloading constructor terjadi pada class “Mahasiswa”. Sebuah class dapat memiliki lebih dari 1 konstruktor. Bahkan, sebuah class dapat tidak dituliskan konstruktornya. Secara default konstruktor kosong akan terbentuk jika tidak dituliskan konstruktor secara eksplisit pada sebuah class. Bentuk dari konstruktor kosong:

Penulisan Constructor di jAVA

```
public NamaClass(...){

}
```

Penulisan Constructor di PHP

```
public function __construct(...){

}
```

Listing Program 2: Class Mahasiswa.java

```
public class Mahasiswa {
    private String nim;
    private String nama;
    private int tinggi;
    private boolean pindahan;

    public String getNim() {
        return nim;
    }

    public void setNim(String nim) {
        this.nim = nim;
    }

    public String getNama() {
        return nama;
    }

    public void setNama(String nama) {
        this.nama = nama;
    }

    public int getTinggi() {
        return tinggi;
    }
```

```
    public void setTinggi(int tinggi) {
        this.tinggi = tinggi;
    }

    public boolean isPindahan() {
        return pindahan;
    }

    public void setPindahan(boolean pindahan) {
        this.pindahan = pindahan;
    }

    public Mahasiswa(String nim, String nama, int tinggi) {
        this.nim = nim;
        this.nama = nama;
        this.tinggi = tinggi;
    }

public Mahasiswa(String nim, String nama, int tinggi, boolean pindahan) {
        this(nim,nama,tinggi); //memanggil konstruktor 3 parameter
        this.pindahan = pindahan;
    }
}
```

Sebagai catatan, bentuk isPindahan merupakan hal yang sama dengan bentuk get lainnya. Hanya saja, untuk pengembalian tipe Boolean, “get” akan berubah menjadi “is”. Hal ini hanya bentuk kesepakatan.

Untuk membentuk objek dari class di atas dan menampilkan nilai objek, dapat dibentuk sebuah class baru dengan kandungan public static void main(String args[]) sebagai berikut:

```
public class MainMhs {
    public static void main(String[] args) {
        Mahasiswa m1 = new Mahasiswa("6701148000", "Angga", 166);
        m1.setPindahan(false);

        //menampilkan data
        System.out.println("Data Mahasiswa 1");
        System.out.println("NIM: "+m1.getNim());
        System.out.println("Nama: "+m1.getNama());
        System.out.println("Tinggi Badan: "+m1.getTinggi());
        if(m1.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }
    }
}
```

Class di atas membentuk sebuah objek dari class Mahasiswa bernama m1 melalui konstruktor dengan 3 parameter. Karena class Mahasiswa memiliki 4 atribut, sedangkan konstruktor hanya memberikan fasilitas 3 parameter (nim, nama, dan tinggi), maka data pindahan/tidak diberikan melalui set-nya. Contoh lain dari pembentukan mahasiswa (m2) yang dibentuk melalui konstruktor 4 parameter dituliskan sebagai berikut.

```
public class MainMhs {
    public static void main(String[] args) {
        Mahasiswa m1 = new Mahasiswa("6701148000", "Angga", 166);
        m1.setPindahan(false);

        //menampilkan data
        System.out.println("Data Mahasiswa 1");
        System.out.println("NIM: "+m1.getNim());
        System.out.println("Nama: "+m1.getNama());
        System.out.println("Tinggi Badan: "+m1.getTinggi());
        if(m1.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }

        Mahasiswa m2 = new Mahasiswa("6701148001", "Rena", 154, true);

        //menampilkan data
        System.out.println();
        System.out.println("Data Mahasiswa 2");
        System.out.println("NIM: "+m2.getNim());
        System.out.println("Nama: "+m2.getNama());
        System.out.println("Tinggi Badan: "+m2.getTinggi());
        if(m2.isPindahan()){
            System.out.println("Mahasiswa pindahan");
        }else{
            System.out.println("Mahasiswa reguler");
        }
    }
}
```

2.2 Latihan

Latihan 1

Seorang developer ingin membuat sebuah simulasi fighting game menggunakan konsep PBO.

Setelah menganalisis kebutuhan, class diagram yang dihasilkan adalah sebagai berikut:

GameCharacter
-name : string -lifePoint : int -attackHitPoint : int -attackKickPoint : int
+hit(in karB : GameCharacter) : void +kick(in karB : GameCharacter) : void +GameCharacter(in name : string, in attackHitPoint : int, in attackKickPoint : int) +getLifePoint() : int +getName() : string

Keterangan:

1. Atribut name merupakan identitas karakter
2. Atribut lifePoint merupakan atribut yang merepresentasikan tenaga dari tiap karakter. Nilai default dari lifePoint adalah 100.
3. Atribut attackHitPoint merupakan kekuatan yang diberikan ketika melakukan hit.
4. Atribut attackKickPoint merupakan kekuatan yang diberikan ketika melakukan kick
5. Method "hit" merupakan method untuk menendang lawan. Masukan dari method ini adalah objek GameCharacter lain. Efek dari method ini mengurangi lifePoint lawan sebanyak attackHitPoint yang dimiliki.
6. Method "kick" merupakan method untuk menendang lawan. Masukan dari method ini adalah objek GameCharacter lain. Efek dari method ini mengurangi lifePoint lawan sebanyak attackKickPoint yang dimiliki.
7. Konstruktor digunakan untuk menge-set nama, attackHitPoint, dan attackKickPoint sesuai parameter masukan. Selain itu, konstruktor juga melakukan set lifePoint sebesar nilai default.
8. Method getLifePoint dan getName merupakan method get bagi kedua atribut.
9. Tidak terdapat konstruktor kosong.

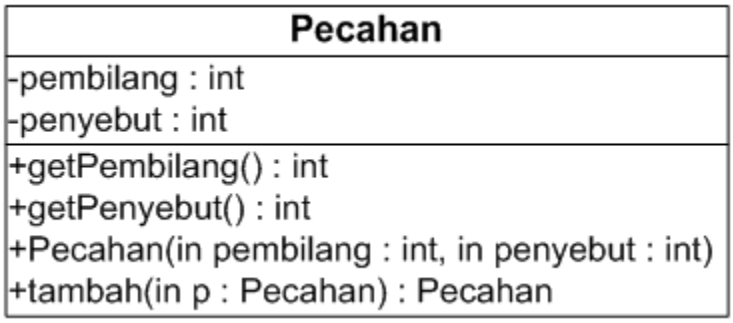
Berdasarkan skenario di atas, bantulah developer untuk mengkodekan class di atas. Lalu, pada class Main, berikan scenario berikut:

1. Buat sebuah objek dengan name: "Raiden", attackHitPoint: 10, attackKickPoint: 20.
2. Buat sebuah objek dengan name: "Sub-Zero", attackHitPoint: 5, attackKickPoint: 25
3. Objek Raiden memulai pertarungan dengan melakukan tendangan pada objek Sub-Zero.
4. Objek Sub-Zero melakukan perlawanan dengan menendang balik objek Raiden.
5. Objek Sub-Zero menyerang Raiden dengan pukulan berturut-turut sebanyak 3x (gunakan perulangan).
6. Pertarungan diakhiri oleh Raiden dengan melakukan tendangan beruntun 4x pada objek Sub-Zero (gunakan perulangan).
7. Tampilkan lifePoint dari objek Raiden dan objek Sub-Zero. Bandingkan kedua lifePoint tersebut.

8. Tampilkan status kemenangan dari pertarungan kedua objek. Objek yang menang adalah objek yang memiliki lifePoint tertinggi di akhir pertarungan.

2.3 Latihan 2

Latihan 2



Diketahui sebuah class seperti disamping. Konstruktor memiliki 2 paramter untuk menge-set kedua atribut dari class Pecahan. Method tambah merupakan method untuk menambahkan sebuah objek Pecahan dengan objek Pecahan lainnya. Method ini merupakan function dengan nilai yang dikembalikan merupakan hasil penambahan kedua objek Pecahan dengan tipe data class Pecahan tersebut. Berdasarkan informasi tersebut,

1. Kodekanlah class di samping beserta logika dari setiap method
2. Bentuklah sebuah Main class yang membentuk 2 objek sebagai berikut:

   Objek 1: 3/2
   Objek 2: 5/6