PERTEMUAN KELIMA

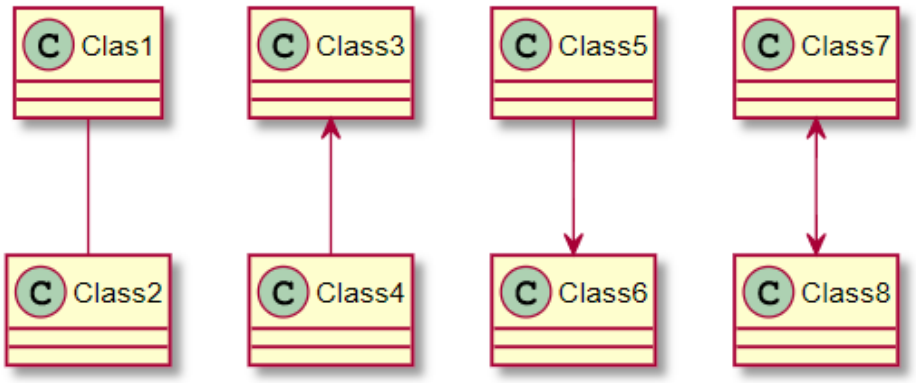
Association, Aggregation, and Composition

Prepared by Adi Wahyu Pribadi

Pengenalan

Pada konsep pemrograman berbasis objek, hubungan antar kelas terdapat empat yaitu pewarisan, asosiasi, agregasi, dan komposisi. Pewarisan telah dipelajari pada pertemuan sebelumnya. Materi kali ini akan fokus pada asosiasi, agregasi, dan komposisi.

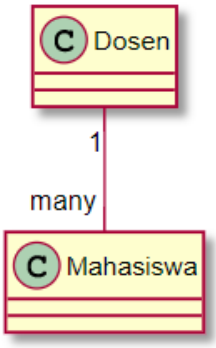
Association/Asosiasi



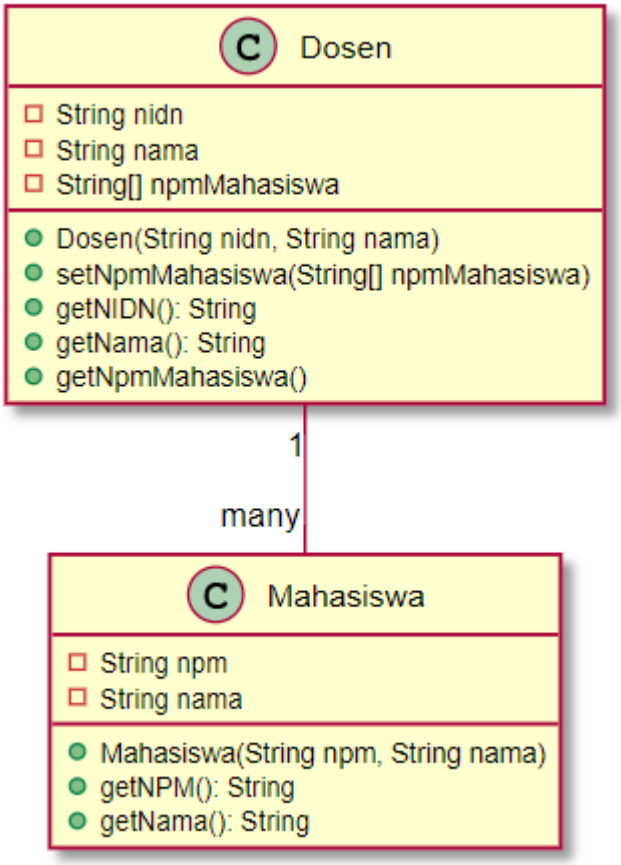
Pada gambar di atas, terlihat hubungan asosiasi antar Clas. Relasi assosiasi biasa disebut is “a” relationship. Assosiasi berarti bahwa sebuah object “menggunakan” object yang lain.

Assosiasi adalah sebuah relasi dimana semua object memiliki lifecycle nya sendiri dan tidak ada yang bertindak sebagai owner.

Pada gambar class diagram di samping, terdapat hubungan asosiasi antara Class Dosen dan Class Mahasiswa. Hubungan keduanya terlihat dari garis lurus tersambung antara Class Dosen dan Class Mahasiswa. Terdapat keterangan kardinalitas yang menyatakan bahwa Class Dosen dapat memiliki banyak Class Mahasiswa.



Penjelasan Class Diagram di samping terlihat pada Source Code berikut.



Mahasiswa.java

```
class Mahasiswa {
    private String npm;
```

```
    private String nama;

    public Mahasiswa(String npm, String nama) {
        this.npm = npm;
        this.nama = nama;
    }

    public String getNPM() { return this.npm; }

    public String getNama() { return this.nama; }
}
```

Dosen.java

```
class Dosen {
    private String nidn;
    private String nama;
    private String[] npmMahasiswa;

    public Dosen(String nidn, String nama) {
        this.nidn = nidn;
        this.nama = nama;
    }

    public void setNpmMahasiswa (String[] npmMahasiswa) {
        this.npmMahasiswa = npmMahasiswa;
    }

    // menampilkan npm-npm mahasiswa-nya Dosen
    public void getNpmMahasiswa() {
        // mendapatkan panjang Array
        int n = this.npmMahasiswa.length;
        int i = 0;  // counter
        while (i < n) {
            System.out.println("Peserta ke-" + (i+1) +": " +
this.npmMahasiswa[i]);
            i++;
        }
    }

    public String getNama() {
```

```
        return this.nama;
    }

    public String getNIDN() {
        return this.nidn;
    }
}
```

Implementasi App.java

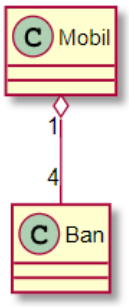
```
public class App {
    public static void main(String[] args) throws Exception {
        Dosen amir = new Dosen("12345", "Amir Murtako");
        Dosen bambang = new Dosen("12346", "Bambang Riono");

        Mahasiswa rina = new Mahasiswa("123", "Rina");
        Mahasiswa rano = new Mahasiswa("124", "Rano");
        Mahasiswa doel = new Mahasiswa("125", "Doel");
        Mahasiswa jajang = new Mahasiswa("126", "Jajang");

        String[] PAPakAmir = {doel.getNPM(), jajang.getNPM()};
        String[] PAMasBambang = {rina.getNPM(), rano.getNPM()};

        amir.setNpmMahasiswa(PAPakAmir);
        bambang.setNpmMahasiswa(PAMasBambang);

        System.out.println("Pak " + amir.getNama() + " dengan NIDN" +
amir.getNIDN());
        amir.getNpmMahasiswa();
        System.out.println("Mas " + bambang.getNama() + " dengan NIDN" +
bambang.getNIDN());
        bambang.getNpmMahasiswa();
    }
}
```

Output

```
Pak Amir Murtako dengan NIDN12345
Peserta ke-1: 125
Peserta ke-2: 126
Mas Bambang Riono dengan NIDN12346
Peserta ke-1: 123
Peserta ke-2: 124
```

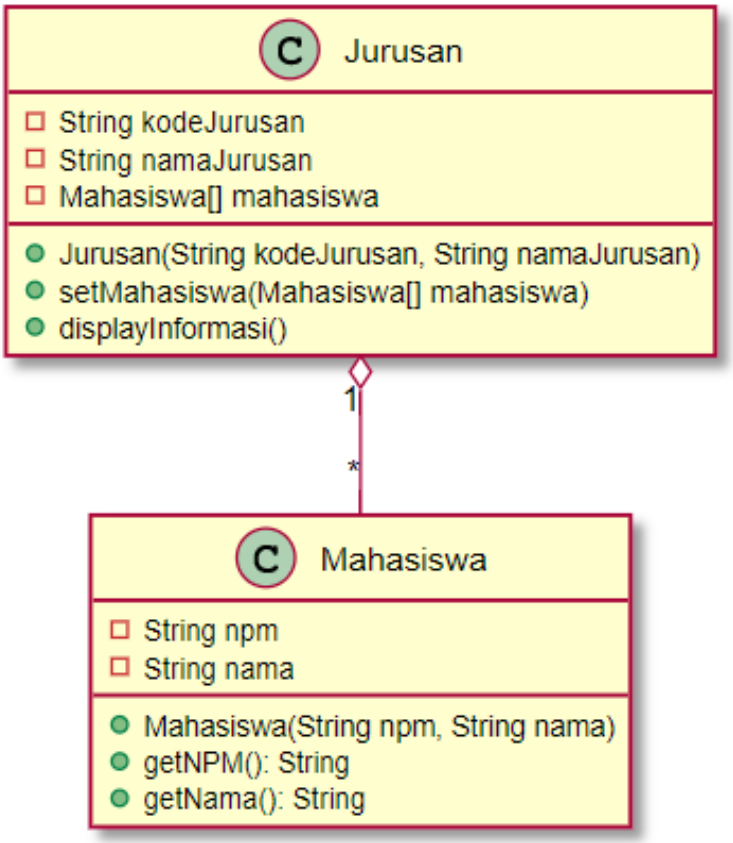
Aggregation/Agregasi

Agregasi adalah bentuk hubungan yang lebih khusus dari Assosiasi dimana sebuah object memiliki lifecycle nya sendiri tapi dengan kepemilikan. Relasinya biasa di sebut relasi “has-a”. Hubungan agregasi digambarkan dengan diamond putih, yang ditempelkan pada kelas yang memiliki, tidak dibubuhkan panah pada ujung yang tidak memiliki simbol diamond putih. Kemudian juga dibubuhkan kardinalitas seperti pada hubungan asosiasi.



Gambar Class Diagram di samping menunjukkan bahwa Class Mobil memiliki hubungan agregasi dengan Class Ban. Class Mobil memiliki 4 Class Ban. Ketika Class Mobil dihancurkan, Class Ban masih dapat berdiri sendiri.

Contoh lain adalah hubungan agregasi antara Class Jurusan dan Class Mahasiswa seperti pada gambar di bawah:



Jurusan.java

```
class Jurusan {
    private String kodeJurusan;
    private String namaJurusan;
    private Mahasiswa[] mahasiswa;

    public Jurusan(String kodeJurusan, String namaJurusan) {
        this.kodeJurusan = kodeJurusan;
        this.namaJurusan = namaJurusan;
    }

    public void setMahasiswa(Mahasiswa[] mahasiswas) {
        this.mahasiswa = mahasiswas;
    }

    public void displayInformasi() {
        System.out.println("Kode Jurusan" + this.kodeJurusan);
        System.out.println("JURUSAN: " + this.namaJurusan);
        System.out.println("Daftar Mahasiswa");
        // hitung jumlah mahasiswa
        int n = this.mahasiswa.length;
        int i = 0; // counter

        while (i < n) {
            System.out.println(mahasiswa[i].getNPM() + ": " +
mahasiswa[i].getNama());
            i++;
        }
    }
}
```

Implementasi App.java

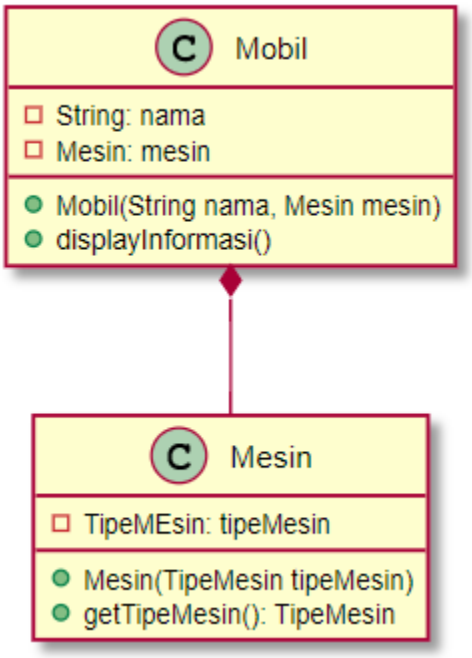
```
public class App {
    public static void main(String[] args) throws Exception {
        Mahasiswa rina = new Mahasiswa("123", "Rina");
        Mahasiswa rano = new Mahasiswa("124", "Rano");
        Mahasiswa doel = new Mahasiswa("125", "Doel");
        Mahasiswa jajang = new Mahasiswa("126", "Jajang");

        Jurusan informatika = new Jurusan("45", "Teknik Informatika");
        Jurusan elektro = new Jurusan("41", "Teknik Elektro");

        Mahasiswa[] mahasiswaInformatika = {doel, jajang};
        Mahasiswa[] mahasiswaElektro = {rina, rano};

  informatika.setMahasiswa(mahasiswaInformatika);
        elektro.setMahasiswa(mahasiswaElektro);

        informatika.displayInformasi();
        elektro.displayInformasi();
    }
}
```

Output

```
Kode Jurusan45
JURUSAN: Teknik Informatika
Daftar Mahasiswa
125: Doel
126: Jajang
Kode Jurusan41
JURUSAN: Teknik Elektro
Daftar Mahasiswa
123: Rina
124: Rano
```

Composition/Komposisi

Komposisi digambarkan menggunakan diamond penuh yang menyatakan memiliki bagian seperti pada agregasi. Pada komposisi lifecycle object bergantung pada object ownernya.



Pada gambar Class Diagram di atas, terlihat bahwa terdapat hubungan Komposisi antara Mobil dan Mesin. Ketika Mobil dihancurkan, Mesin juga ikut hancur.

TipeMesin.java

```
public enum TipeMesin {
    BENSIN,
    DIESEL
}
```

Mesin.java

```
class Mesin {
    private final TipeMesin tipeMesin;

    public Mesin(TipeMesin tipeMesin) {
        this.tipeMesin = tipeMesin;
    }

    public TipeMesin getTipeMesin() {
        return this.tipeMesin;
    }
}
```

Mobil.java

```
class Mobil {
    private String nama;
    private final Mesin mesin;

    public Mobil(String nama, Mesin mesin) {
        this.nama = nama;
        this.mesin = mesin;
    }

    public void displayInformasi() {
        System.out.println("Mobil " + this.nama);
        System.out.println("Mesinnya " +
this.mesin.getTipeMesin().toString());
    }
}
```

Implementasi App.java

```
public class App {
    public static void main(String[] args) throws Exception {
        Mobil avanza = new Mobil("Avanza", new Mesin(TipeMesin.BENSIN));
        Mobil pajero = new Mobil("Pajero", new Mesin(TipeMesin.DIESEL));

        avanza.displayInformasi();
        pajero.displayInformasi();
    }
}
```