

# Hubungan Antar Class

Pemograman Berorientasi Objek

# Hubungan antar kelas

- Dalam Obyek Oriented Programming, kelas-kelas yang terbentuk dapat memiliki hubungan satu dengan yang lainnya, sesuai dengan kondisi dari kelas-kelas yang bersangkutan

# Hubungan antar kelas

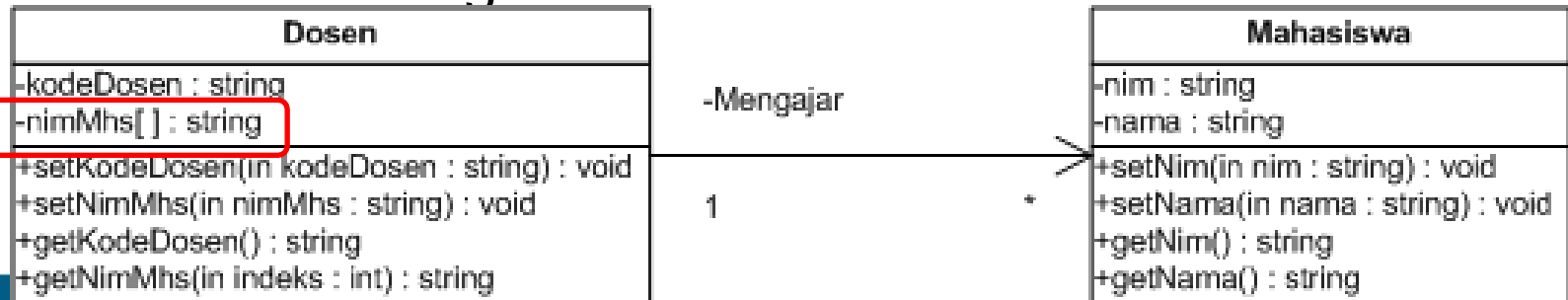
- Asosiasi
- Agregasi
- Komposisi

# Asosiasi

- Asosiasi merupakan hubungan antara dua kelas di yang merupakan hubungan struktural yang menggambarkan himpunan link antar obyek.
- Asosiasi bisa **uni-directional** (satu arah) atau **bi-directional** (dua arah)
- Simbol:
  - Garis tegas dari suatu class ke class lain
  - Panah merupakan pernyataan “navigable”

# Contoh Implementasi

- Membuat sebuah class dosen dengan link atribut nim mahasiswa
- Tipe data dari atribut link mengikuti tipe data asli pada kelas asal
- Contoh class diagram:



# Contoh

```
Mahasiswa.java  Dosen.java  Main.java

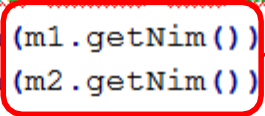
1  public class Mahasiswa {
2      private String nim;
3      private String nama;
4      public void setNama (String nama) {
5          this.nama = nama;
6      }
7      public void setNim (String nim) {
8          this.nim = nim;
9      }
10     public String getNim () {
11         return this.nim;
12     }
13     public String getNama () {
14         return this.nama;
15     }
16 }
17
```

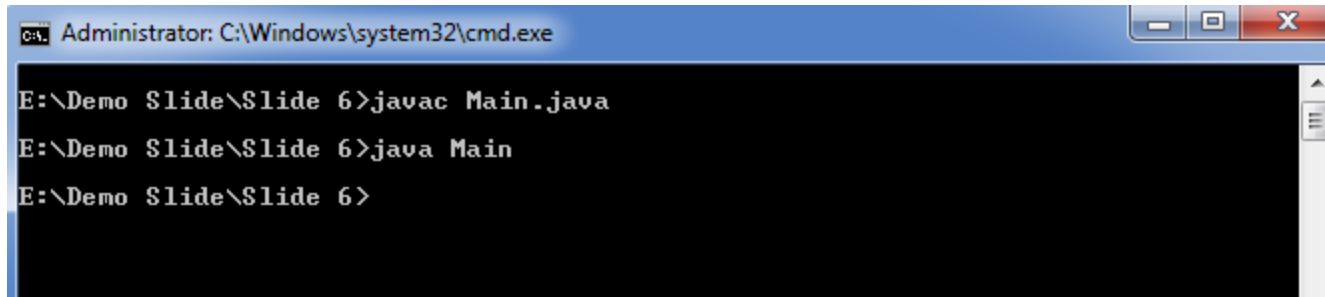
```
1 public class Dosen {
2     private String kodeDosen;
3     private String[] nimMhs = new String[5];
4     private int jumlahMhs;
5
6     public void setKodeDosen(String kodeDosen) {
7         this.kodeDosen = kodeDosen;
8     }
9     public void setNimMahasiswa(String nimMhs) {
10         if (jumlahMhs < this.nimMhs.length) {
11             this.nimMhs[jumlahMhs] = nimMhs;
12             jumlahMhs++;
13         }
14     }
15     public String getKodeDosen() {
16         return this.kodeDosen;
17     }
18     public int getJumlahMhs() {
19         return this.jumlahMhs;
20     }
21     public String getNimMhs(int indeks) {
22         return (nimMhs[indeks]);
23     }
24 }
```



```
1 public class Main{
2     public static void main(String args[]){
3         //membentuk objek mhs 1
4         Mahasiswa m1 = new Mahasiswa();
5         m1.setNim("6301090021");
6         m1.setNama("Dita Dwita Hayuning Tyas Putri");
7
8         //membentuk objek mhs 2
9         Mahasiswa m2 = new Mahasiswa();
10        m2.setNim("6301100190");
11        m2.setNama("Nukky Sugandi");
12
13        //membentuk objek Dosen
14        Dosen d = new Dosen();
15        d.setKodeDosen("RBD");
16
17        //hubungkan antara dosen dan mahasiswa
18        d.setNimMahasiswa(m1.getNim());
19        d.setNimMahasiswa(m2.getNim());
20    }
21 }
```

Perhatikan adanya pengiriman atribut dari objek m1 dan m2 ke objek d



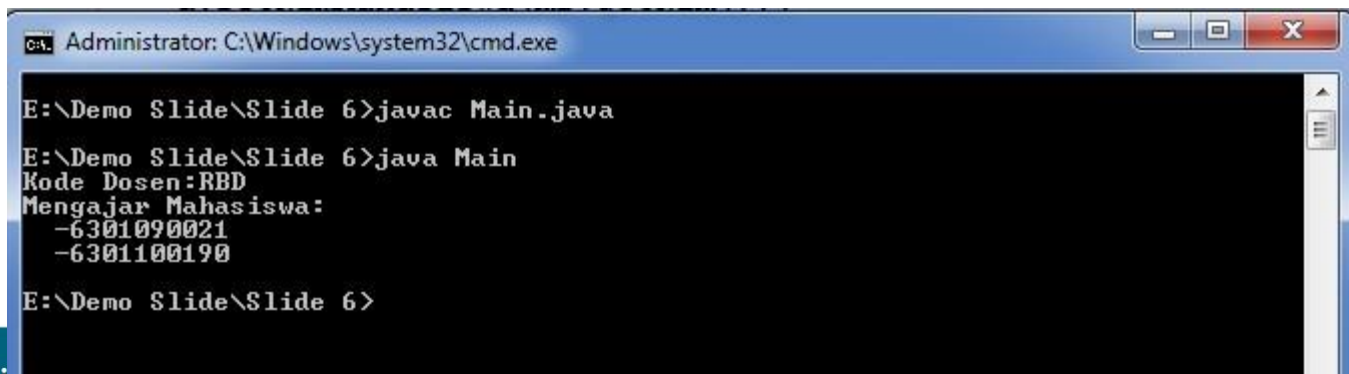


```
Administrator: C:\Windows\system32\cmd.exe

E:\Demo Slide\Slide 6>javac Main.java
E:\Demo Slide\Slide 6>java Main
E:\Demo Slide\Slide 6>
```

Sekarang....  
Bagaimana menampilkannya?

```
21 //menampilkan data dosen "d" dan mahasiswa yang diajar
22 System.out.println("Kode Dosen:"+d.getKodeDosen());
23 System.out.println("Mengajar Mahasiswa:");
24
25 //ambil jumlah mahasiswa yang diajar dosen d
26 int jum = d.getJumlahMhs();
27
28 //ambil nim yang diajar oleh dosen d dengan perulangan
29 for(int i=0; i<jum; i++){
30     System.out.println("    -"+d.getNimMhs(i));
31 }
```



The screenshot shows a Windows command prompt window titled "Administrator: C:\Windows\system32\cmd.exe". The user has entered the following commands and received the following output:

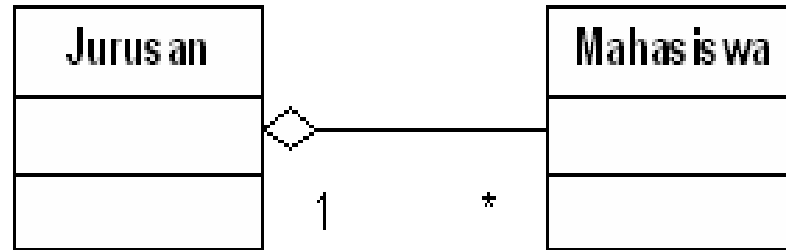
```
E:\Demo Slide\Slide 6>javac Main.java
E:\Demo Slide\Slide 6>java Main
Kode Dosen:RBD
Mengajar Mahasiswa:
-6301090021
-6301100190
E:\Demo Slide\Slide 6>
```

# Agregasi

- Agregasi merupakan hubungan antara dua kelas di mana kelas yang satu merupakan **bagian** dari kelas yang lain namun kedua kelas ini dapat **berdiri sendiri-sendiri**.
- Simbol yang digunakan: hollow diamond
- Simbol panah menyatakan suatu class navigable terhadap class lain

# Contoh Class Diagram

- Jurusan menyimpan nilai atribut dari mahasiswa dengan tipe data class bentukan “Mahasiswa”



# Contoh Agregasi

- Mahasiswa dengan jurusanannya
  - Mahasiswa memiliki objek sendiri
  - Jurusan memiliki objek sendiri
  - Mahasiswa menjadi bagian dari jurusanannya
- Member dari sebuah idol group
  - Sebuah idol group memiliki badan usaha sendiri
  - Setiap artis merupakan 1 objek sendiri yang tergabung ke sebuah agency
  - Artist tersebut merupakan bagian dari idol teater group

# Hubungan Antar Kelas

- Asosiasi

- Merupakan hubungan “link”
- Menyimpan nilai atribut dengan tipe data asli
- Digambarkan dengan garis tegas

- Agregasi

- Merupakan hubungan “bagian”
- Menyimpan nilai atribut dengan tipe data class bentukan
- Digambarkan dengan hollow diamond

# Contoh Implementasi

kelas yang satu  
merupakan **bagian**  
dari kelas yang lain

Suatu kelas menjadi  
atribut bagi kelas lain

Pada Driver Class  
**terdapat objek**  
**referensi** tiap kelas  
dan **pengiriman objek**

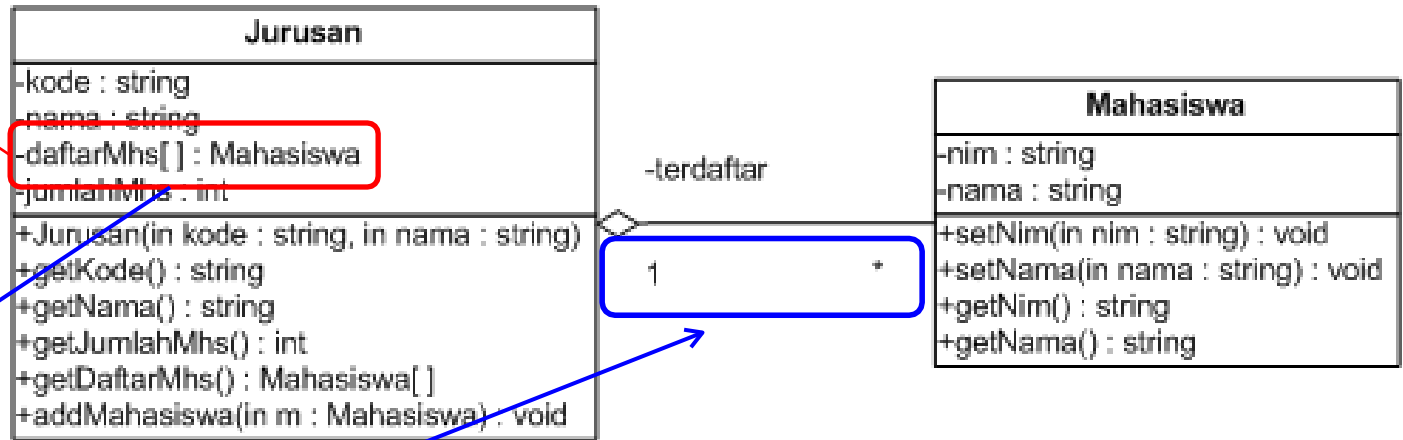


## Contoh Soal

- Buatlah sebuah hubungan agregasi antara jurusan/prodi dengan mahasiswanya.
- Jurusan menyimpan nilai dari mahasiswa
- Satu jurusan hanya bisa ditempati maksimal 10 mahasiswa

# Contoh Class Diagram

Terdapat class  
yang menjadi  
atribut



Tipe array of object,  
karena hubungan 1..n

# Contoh

```
Mahasiswa.java  Jurusan.java  Main.java

1  public class Mahasiswa {
2      private String nim;
3      private String nama;
4      public void setNama (String nama) {
5          this.nama = nama;
6      }
7      public void setNim (String nim) {
8          this.nim = nim;
9      }
10     public String getNim () {
11         return this.nim;
12     }
13     public String getNama () {
14         return this.nama;
15     }
16 }
17
```

```
1 public class Jurusan{
2     private String kode,nama;
3     private Mahasiswa daftarMhs[] = new Mahasiswa[10];
4     private int jumlahMhs;
5
6     public Jurusan(String kode, String nama){
7         this.kode = kode;
8         this.nama = nama;
9     }
10    public String getKode(){
11        return this.kode;
12    }
13    public String getNama(){
14        return this.nama;
15    }
16    public int getJumlahMhs(){
17        return this.jumlahMhs;
18    }
```

```
19
20
21
22
23
24
25    public void addMahasiswa(Mahasiswa m){
26        if(jumlahMhs<this.daftarMhs.length){
27            daftarMhs[jumlahMhs] = m;
28            jumlahMhs++;
29        }
30    }
31    public Mahasiswa[] getDaftarMhs(){
32        return this.daftarMhs;
33    }
34 }
```

```
Mahasiswa.java  Jurusan.java  Main.java
1 public class Main{
2     public static void main(String args[]){
3         //bentuk objek jurusan
4         Jurusan j = new Jurusan("MI","Manajemen Informatika");
5
6         //membentuk objek mhs 1
7         Mahasiswa m1 = new Mahasiswa();
8         m1.setNim("6301090021");
9         m1.setNama("Dita Dwita Hayuning Tyas Putri");
10
11        //membentuk objek mhs 2
12        Mahasiswa m2 = new Mahasiswa();
13        m2.setNim("6301100190");
14        m2.setNama("Nukky Sugandi");
15
16        //hubungkan mahasiswa dengan jurusan
17        j.addMahasiswa(m1);
18        j.addMahasiswa(m2);
19    }
20 }
```

Terdapat pengiriman objek

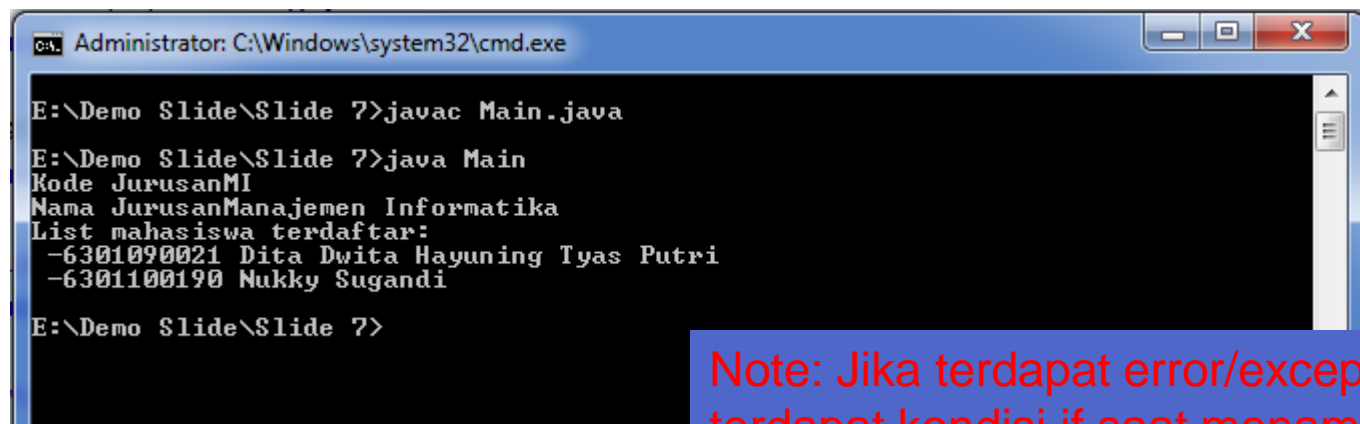


```
Administrator: C:\Windows\system32\cmd.exe

E:\Demo Slide\Slide 7>javac Main.java
E:\Demo Slide\Slide 7>java Main
E:\Demo Slide\Slide 7>
```

Sekarang....  
Bagaimana menampilkannya?

```
System.out.println("Kode Jurusan"+j.getKode());  
System.out.println("Nama Jurusan"+j.getNama());  
System.out.println("List mahasiswa terdaftar:");  
//ambil data mahasiswa di jurusan  
Mahasiswa[] daftar = j.getDaftarMhs();  
for(int i=0; i<daftar.length;i++){  
    System.out.println(" -"+daftar[i].getNim()+" "+daftar[i].getNama());  
}
```



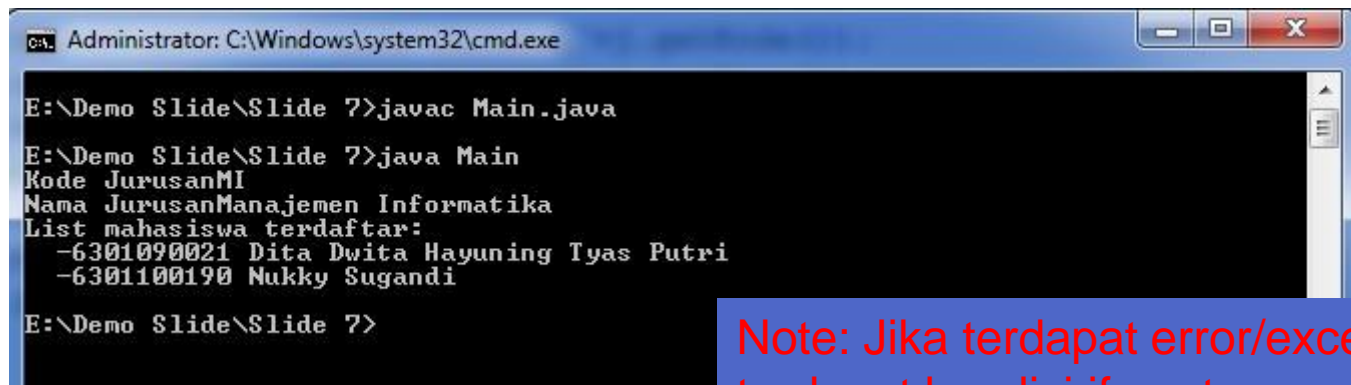
```
Administrator: C:\Windows\system32\cmd.exe  
  
E:\Demo Slide\Slide 7>javac Main.java  
E:\Demo Slide\Slide 7>java Main  
Kode JurusanMI  
Nama JurusanManajemen Informatika  
List mahasiswa terdaftar:  
-6301090021 Dita Dwita Hayuning Tyas Putri  
-6301100190 Nukky Sugandi  
E:\Demo Slide\Slide 7>
```

Note: Jika terdapat error/exception, pastikan terdapat kondisi if saat menampilkan isi array



```
System.out.println("Kode Jurusan"+j.getKode());  
System.out.println("Nama Jurusan"+j.getNama());  
System.out.println("List mahasiswa terdaftar:");  
//ambil data mahasiswa di jurusan  
Mahasiswa[] daftar = j.getDaftarMhs();  
for(Mahasiswa m:daftar){  
    System.out.println("    -"+m.getNim()+" "+m.getNama());  
}
```

→ Gunakan  
for-loop  
sebagai alternatif



```
Administrator: C:\Windows\system32\cmd.exe  
E:\Demo Slide\Slide 7>javac Main.java  
E:\Demo Slide\Slide 7>java Main  
Kode JurusanMI  
Nama JurusanManajemen Informatika  
List mahasiswa terdaftar:  
-6301090021 Dita Dwita Hayuning Tyas Putri  
-6301100190 Nukky Sugandi  
E:\Demo Slide\Slide 7>
```

Note: Jika terdapat error/exception, pastikan terdapat kondisi if saat menampilkan isi array

```
System.out.println("Kode Jurusan"+j.getKode());  
System.out.println("Nama Jurusan"+j.getNama());  
System.out.println("List mahasiswa terdaftar:");  
//ambil data mahasiswa di jurusan  
Mahasiswa[] daftar = j.getDaftarMhs();  
for(Mahasiswa m:daftar){  
    if(m!=null)  
        System.out.println("  "+m.getNim()+" "+m.getNama());  
}
```

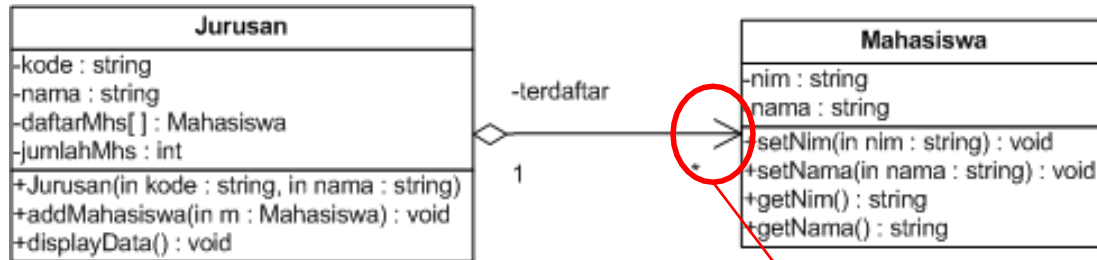
# Alternatif Kode?!!

```
1 public class JurusanV2{
2     private String kode,nama;
3     private Mahasiswa daftarMhs[] = new Mahasiswa[10];
4     private int jumlahMhs;
5
6     public JurusanV2(String kode, String nama){
7         this.kode = kode;
8         this.nama = nama;
9     }
10    public void addMahasiswa(Mahasiswa m){
11        if(jumlahMhs<this.daftarMhs.length){
12            daftarMhs[jumlahMhs] = m;
13            jumlahMhs++;
14        }
15    }
16    public void displayData(){
17        System.out.println("Kode Jurusan"+this.kode);
18        System.out.println("Nama Jurusan"+this.nama);
19        for(Mahasiswa m:daftarMhs){
20            if(m!=null)
21                System.out.println("    -"+m.getNim()+" "+m.getNama());
22        }
23    }
24 }
```

```
1 public class MainV2{
2     public static void main(String args[]){
3         //bentuk objek jurusan
4         JurusanV2 j = new JurusanV2("MI", "Manajemen Informatika");
5
6         //membentuk objek mhs 1
7         Mahasiswa m1 = new Mahasiswa();
8         m1.setNim("6301090021");
9         m1.setNama("Dita Dwita Hayuning Tyas Putri");
10
11        //membentuk objek mhs 2
12        Mahasiswa m2 = new Mahasiswa();
13        m2.setNim("6301100190");
14        m2.setNama("Nukky Sugandi");
15
16        //hubungkan mahasiswa dengan jurusan
17        j.addMahasiswa(m1);
18        j.addMahasiswa(m2);
19
20        //tampilkan data
21        j.displayData();
22    }
23 }
```

## Bedanya Apa??

Kode 2 bisa dibilang lebih bersifat **navigable**,  
karena class jurusan mengakses class Mahasiswa.  
Tidak hanya menyimpan, tapi juga **mengakses**  
dan **menampilkan**.



**navigable**

# Kenapa Disebut Agregasi



```
//bentuk objek jurusan  
Jurusan j = new Jurusan("MI", "Manajemen Informatika");  
  
//membentuk objek mhs 1  
Mahasiswa m1 = new Mahasiswa();  
m1.setNim("6301090021");  
m1.setNama("Dita Dwita Hayuning Tyas Putri");
```

Perhatikan bahwa terdapat pembuatan objek dari tiap kelas.

Nilai dari jurusan bisa didapat dari objek "j"

Nilai dari mahasiswa bisa didapat dari objek "m1" dan "m2"

Mereka **berdiri sendiri**, tapi **nilai mahasiswa juga bisa didapat dari "j"**

→ **"m1" dan "m2" bagian dari "j"**

# Komposisi

- Komposisi merupakan **bentuk khusus dari agregasi** di mana kelas yang menjadi part (bagian) baru dapat diciptakan setelah kelas yang menjadi whole (seluruhnya) dibuat dan ketika kelas yang menjadi whole dimusnahkan, maka kelas yang menjadi part ikut musnah.
- Simbol: black diamond
- Simbol panah menyatakan navigable



# Contoh Implementasi

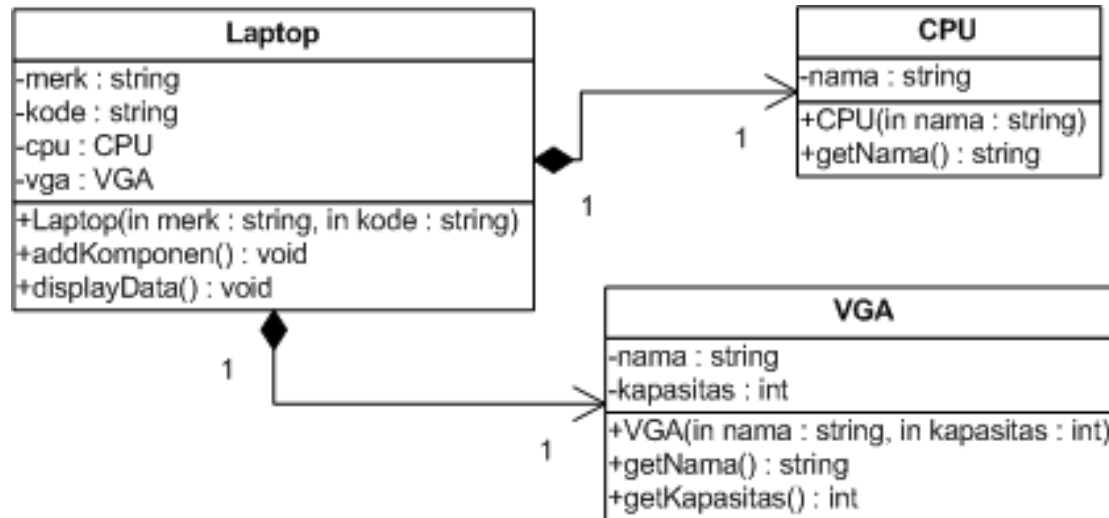
Kelas satu menjadi bagian kelas yang lain

- Implementasi:
  - Objek suatu kelas dibentuk di kelas non Driver
  - Membentuk objek tanpa nama referensi

# Contoh Soal

- Buatlah sebuah hubungan komposisi antara sebuah laptop dengan komponennya.
- Contoh objek:
  - Laptop Asus A43S memiliki CPU intel core i7; VGANvidia 2GB;
  - Laptop Toshiba S006 memiliki CPU intel core i5; VGA Nvidia 2GB;

# Contoh Class Diagram



```
CPU.java  VGA.java  Laptop.java  MainKomp.java
1  public class CPU{
2      private String nama;
3      public CPU(String nama){
4          this.nama = nama;
5      }
6      public String getNama(){
7          return this.nama;
8      }
9  }
```

```
CPU.java  VGA.java  Laptop.java  MainKomp.java
1  public class VGA{
2      private String nama;
3      private int kapasitas;
4      public VGA(String nama, int kapasitas){
5          this.kapasitas = kapasitas;
6      }
7      public String getNama(){
8          return this.nama;
9      }
10     public int getKapasitas(){
11         return this.kapasitas;
12     }
13 }
```

```
1 public class Laptop{
2     private String merk, tipe;
3     private CPU cpu;
4     private VGA vga;
5     public Laptop(String merk, String tipe){
6         this.merk = merk;
7         this.tipe = tipe;
8     }
9     public void addKomponen(CPU cpu, VGA vga){
10        this.cpu = cpu;
11        this.vga = vga;
12    }
13    public void displayData(){
14        System.out.println("Laptop "+this.merk+" dengan tipe "+this.tipe);
15        System.out.println("Punya spesifikasi:");
16        System.out.println("  -"+cpu.getNama());
17        System.out.println("  -"+vga.getNama()+" dengan kapasitas "+vga.getKapasitas()+"GB");
18    }
19 }
```

```
CPU.java  VGA.java  Laptop.java  MainKomp.java

1  public class MainKomp{
2      public static void main(String args[]){
3          //bentuk objek laptop 1
4          Laptop l1 = new Laptop("Asus","A43S");
5          l1.addKomponen(new CPU("Intel Core i7"), new VGA("NVidia",2));
6          l1.displayData();
7
8          //bentuk objek laptop 2
9          Laptop l2 = new Laptop("Toshiba","S006");
10         l2.addKomponen(new CPU("Intel Core i5"), new VGA("NVidia",2));
11         l2.displayData();
12     }
13 }
```

```
Administrator: C:\Windows\system32\cmd.exe

E:\Demo Slide\Slide 7>javac MainKomp.java

E:\Demo Slide\Slide 7>java MainKomp
Laptop Asus dengan tipe A43S
Punya spesifikasi:
-Intel Core i7
-null dengan kapasitas 2GB
Laptop Toshiba dengan tipe S006
Punya spesifikasi:
-Intel Core i5
-null dengan kapasitas 2GB

E:\Demo Slide\Slide 7>
```

Set nama  
Pada  
Konstruktor  
VGA

# Kenapa Disebut Komposisi

```
//bentuk objek laptop 1
Laptop l1 = new Laptop("Asus", "A43S");
l1.addKomponen(new CPU("Intel Core i7"), new VGA("NVidia", 2));
l1.displayData();
```

Perhatikan bahwa terdapat pembuatan objek dari tiap kelas.

Tapi **tidak setiap class punya referensi objek**.

Nilai **CPU** dan **VGA** tidak bisa didapat jika tidak melalui nilai objek "l

Objek "l1" merupakan objek Laptop.

Artinya, ada **ketergantungan penuh** dari class CPU & VGA ke Lapto



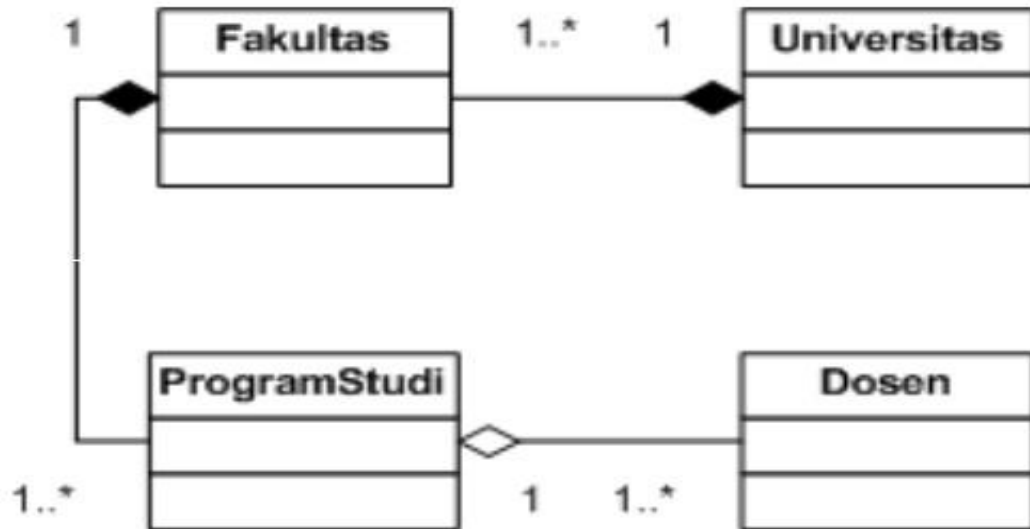
## Agregasi vs Komposisi

- Agregasi **tidak ada** kepemilikan.
  - Bila object yang dibentuk hilang maka object-object penyusunnya akan tetap ada
- Komposisi **ada** kepemilikan.
  - Bila object yang dibentuk hilang maka object-object penyusunnya juga akan hilang

# Agregasi vs Komposisi

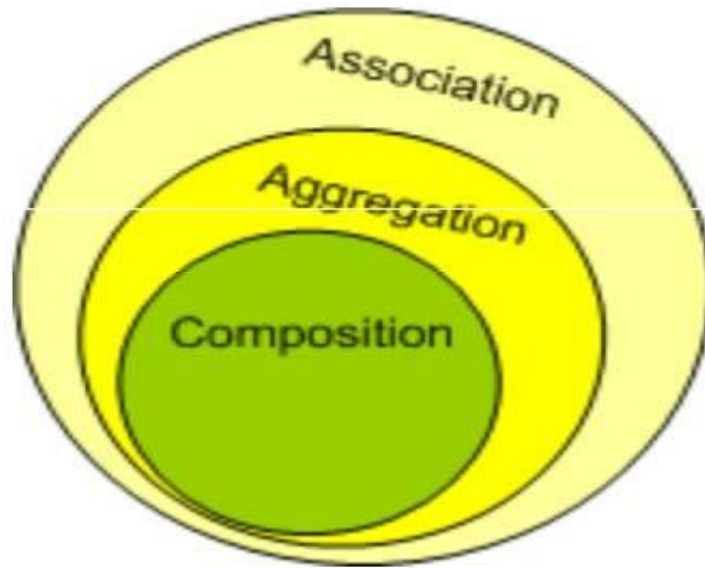
- Universitas terdiri dari 1 atau lebih fakultas
- Fakultas terdiri dari 1 atau lebih Program Studi
- Program Studi memiliki 1 atau lebih Dosen

## Agregasi vs Komposisi



Bila Universitas ditutup maka Fakultas dan Program Studi akan hilang, tapi Dosen tidak.  
Bila suatu Fakultas ditutup maka program studi-program studi di bawahnya juga akan hilang

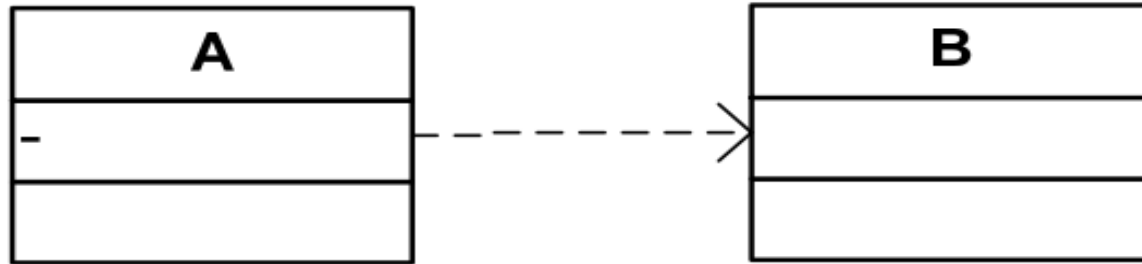
# Asosiasi, Agregasi dan Komposisi



# Dependency

- Relasi yang menggambarkan **ketergantungan** suatu class pada class lainnya
- Contoh: class A memiliki dependency pada class B. Bila class B berubah maka class A juga harus diubah
- Relasi dependency digambarkan dengan garis **putus-putus**

# Dependency



# Dependency

- Ada 3 bentuk dependency
  - Penggunaan class B sebagai **parameter** pada fungsi di class A
  - Penggunaan class B sebagai **nilai kembalian** (return value) pada fungsi di class A
  - Penggunaan class B sebagai **variabel lokal** pada fungsi di class A

# Dependency

- Penggunaan class B sebagai **parameter** pada fungsi di class A

```
class KHS { ... }
```

```
class mahasiswa {  
    float hitungIPKSemester(KHS khs, int sem) {  
        ...  
    }  
}
```






# Dependency

- Penggunaan class B sebagai **nilai kembalian** pada fungsi di class A

```
class KHS { ... }
```

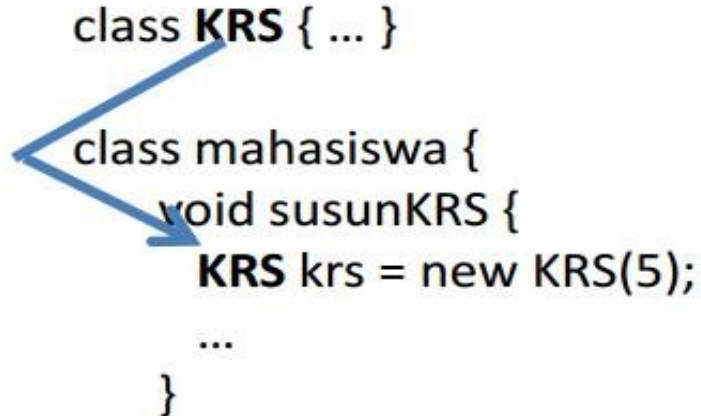
```
class mahasiswa {  
    KHS getKHS(int sem) { ... }  
}
```



# Dependency

- Penggunaan class B sebagai variabel **lokal** pada fungsi di class A

```
class KRS { ... }  
  
class mahasiswa {  
    void susunKRS {  
        KRS krs = new KRS(5);  
        ...  
    }  
}
```

A blue arrow originates from the **KRS** class definition and points to the **KRS** variable used in the `susunKRS` method of the `mahasiswa` class, illustrating a dependency.

# Referensi

- <https://cloud.politala.ac.id/politala>
- <http://materi.amikom.ac.id/2020/04/>