Relasi Antar Kelas

Brigida Arie Minartiningtyas, M.Kom.

Relasi Antar Kelas

- Dalam paradigma pemrograman berorientasi objek, sebuah aplikasi dibangun dengan menggabungkan beberapa kelas. Kelas-kelas tersebut saling bekerjasama untuk menyelesaikan suatu masalah. Dalam aplikasi yang berukuran yang cukup kompleks, banyak kelas kelas yang terlibat dalam aplikasi tersebut. Maka untuk aplikasi yang kompleks tersebut dibutuhkan pemodelan kelas untuk menggambarkan aplikasi yang dibangun.

Unified Modelling Language

- Tools yang digunakan untuk memodelkan kelas-kelas dalam PBO adalah UML (Unified Modelling Language).
- Unified Modelling Language (UML) merupakan spesifikasi pemodelan yang paling banyak digunakan untuk memodelkan struktur dan perilaku aplikasi. UML juga digunakan untuk memodelkan perilaku dan arsitektur aplikasi. UML memiliki banyak jenis diagram yang dapat digunakan untuk memodelkan aplikasi.

Class Diagram

- Namun pembahasan UML disini dibatasi hanya pada kelas diagram saja. Kelas diagram merupakan diagram UML yang digunakan untuk memodelkan kelas-kelas dalam PBO. Kelas diagram ini termasuk dalam kategori pemodelan struktur aplikasi dalam UML.

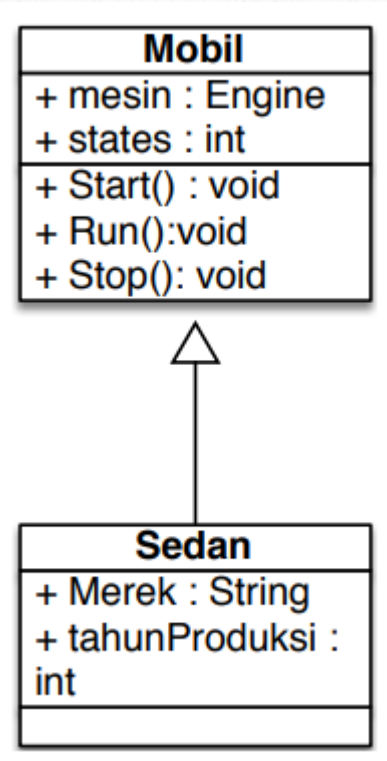
Jenis-jenis Relasi Antar Kelas

- Terdapat beberapa macam relasi antar kelas yaitu :
  - Inheritance
  - Realization
  - Dependency
  - Aggregation
  - Composition

Inheritance

- Inheritance merupakan relasi turunan dimana sebuah kelas diciptakan berdasarkan kelas lainnya. Kelas yang diciptakan disebut dengan kelas anak dan kelas asalnya disebut dengan kelas induk. Kelas anak akan mewarisi seluruh method an property yang dimiliki oleh kelas induknya. Pembahasan tentang inheritance ini telah dibahas pada pertemuan sebelumnya. Simbol UML untuk relasi inheritance dapat dilihat pada gambar berikut ini. Pada gambar tersebut kelas Sedan merupakan turunan dari kelas Mobil.

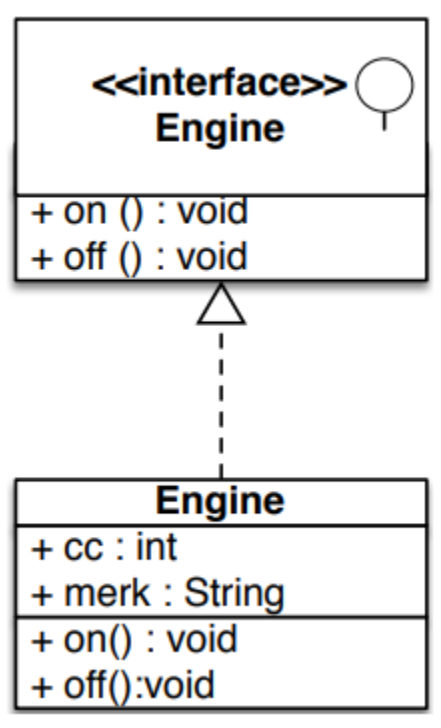
- Relasi turunan sering juga disebut dengan relasi IS-A
- Sedan turunan dari Mobil bisa juga disebut Sedan IS A Mobil



Realization

- Realization merupakan relasi yang terjadi akibat implementasi dari interface. Dalam relasi realization, sebuah kelas yang mengimplementasikan interface tertentu, harus mendefinisikan/ mengimplementasikan seluruh method yang dideklarasikan dalam interface. Pembahasan tentang interface telah dibahas pada pembahasan sebelumnya.

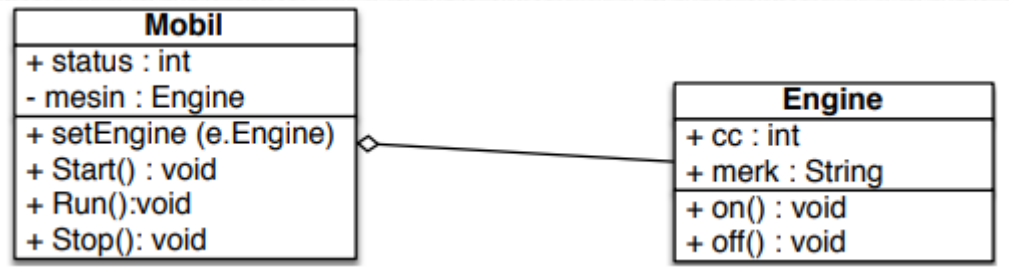
Realization



Agregation

- Relasi antara dua objek dengan mengatakan bahwa satu objek memiliki atau mengandung atau berisi objek yang lain
- Relasi aggregation sering juga disebut relasi HAS-A
  - mobil memiliki mesin
  - rumah memiliki dapur
  - fakultas memiliki jurusan

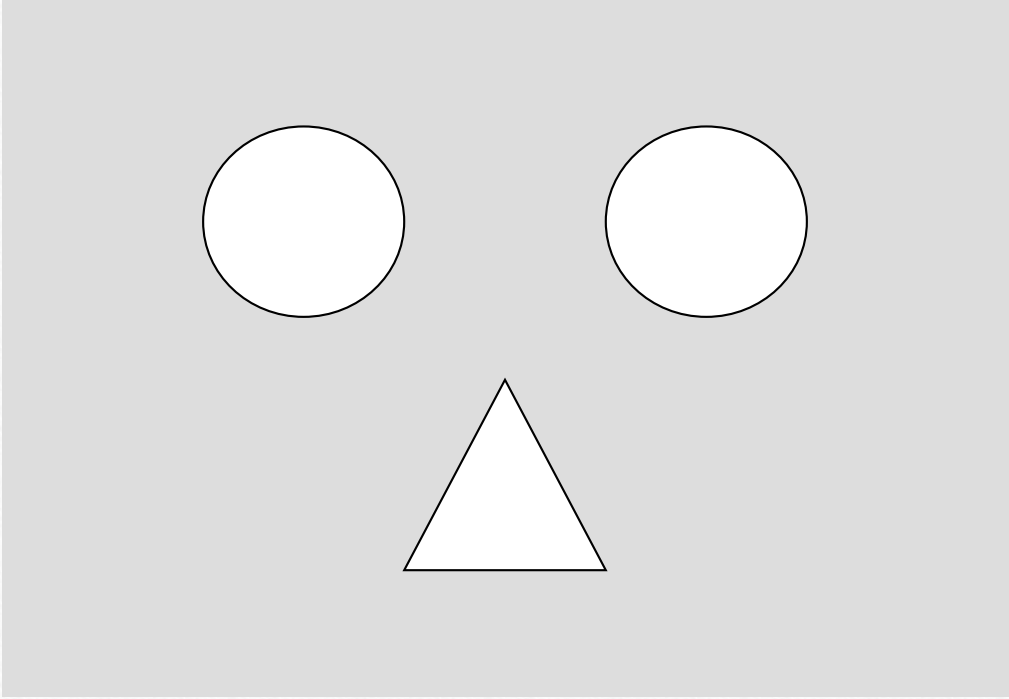
Aggregation

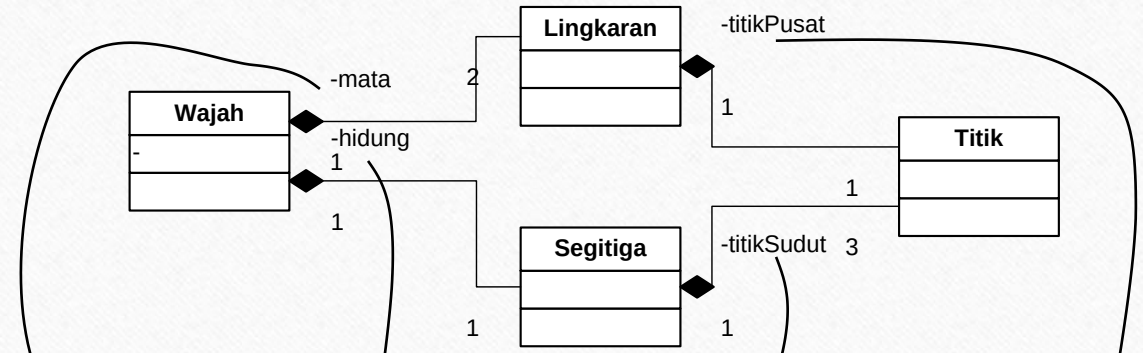



```
class Engine
{  public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON”)
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
  }
```

```
class Program
{ static void Main(string[] args)
  { Engine engine = new Engine();
    Mobil mobil = new Mobil();
    mobil.setEngine(engine);
  }
}
```

```
class Mobil
{  private Engine mesin;
    public int status;
    public void setEngine(Engine e)
    {  mesin=e;
    }
    public void Start()
    {  mesin.On();
    }
    public void Run()
    {  Console.WriteLine(“Run…!”);
    }
    public void Stop()
    {  mesin.Off();
    }
}
```






```
class Wajah
{
    Lingkaran mata[ ];
    Segitiga hidung;
    .....
}
```

```
class Segitiga
{
    Titik titikSudut[ ];
    .....
}
```

```
class Lingkaran
{
    Titik titikPusat;
    .....
}
```

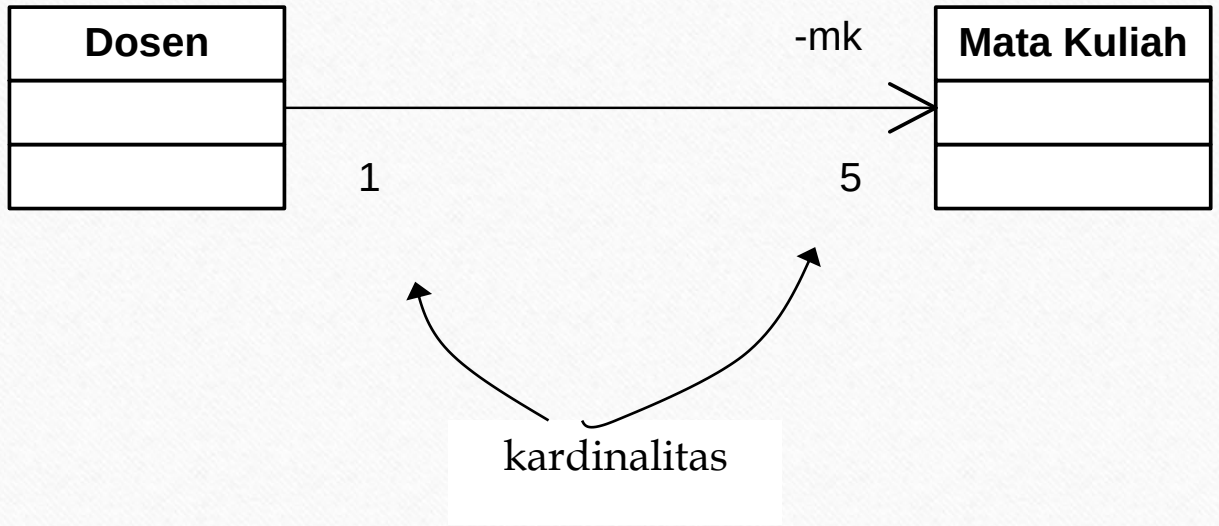
Asosiasi

- Bagaimana relasi yang terjadi antara objek dosen dengan mata kuliah ?
- Bagaimana relasi yang terjadi antara objek mahasiswa dengan mata kuliah ?
- Dalam Relasi perlu diperhatikan Kardinalitas
  - berapa objek yang terlibat dari masing-masing kelas yang terlibat.
  - apakah relasi tersebut bersifat wajib (*mandatory*) atau opsional.

Asosiasi

- Menyatakan suatu hubungan struktural antar objek. yang menggambarkan objek dari suatu kelas dihubungkan ke objek dari kelas lain
- Menunjukkan variabel dalam suatu kelas yang menyimpan rujukan bertipe kelas lain

Asosiasi

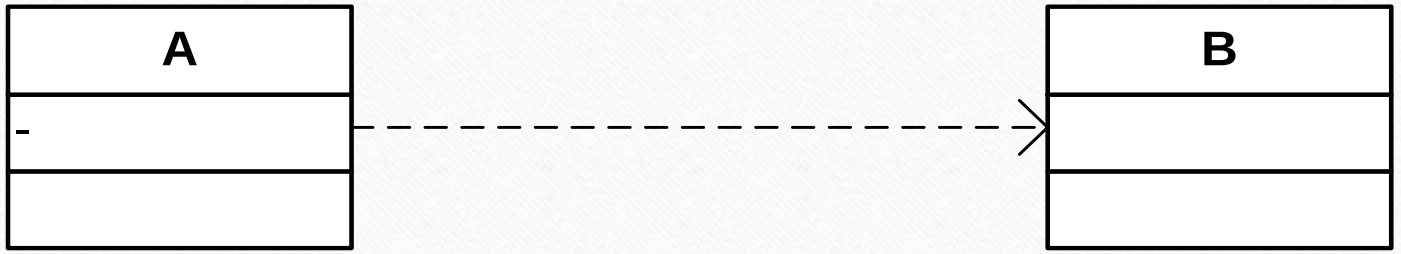


Asosiasi

Depedency

- Dependency merupakan relasi antar kelas dimana satu kelas membutuhkan atau tergantung kepada kelas lainnya. Tapi ketergantungan tersebut tidak timbal balik.
- Suatu kelas A bergantung pada kelas B → jika kelas B mengalami perubahan maka kelas A akan terkena dampak perubahan tersebut
- Relasi dependency ini digambarkan dengan panah yang dari satu kelas ke kelas lainnya. Arah panah menunjukkan kelas yang dibutuhkan.

Depedency



Perwujudan Relasi Depedency

1. Penggunaan kelas B sebagai parameter pada fungsi di kelas A

```
class B { ... }
class A
{
    void fungsiA(B varB) { ..... }
}
```

```java
public class email {
    private String subjek, pesan, kepada;

    public email(String subjek, String pesan) {
        this.subjek = subjek;
        this.pesan = pesan;
    }

    public String getSubjek() {
        return subjek;
    }

    public String getPesan() {
        return pesan;
    }

    public String getKepada() {
        return kepada;
    }

    public void setKepada(String kepada) {
        this.kepada = kepada;
    }

}
```

```java
public class Aplikasi {
    private String namaplikasi;
    private email e1,e2;

    public email getE1(){
        return e1;
    }
    public email getE2(){
       return e2;
    }
    public Aplikasi(String namaplikasi) {
        this.namaplikasi = namaplikasi;
    }

    public String getNamaplikasi() {
        return namaplikasi;
    }

    public void buatPesan(int nopesan, String judul, String pesan){
     if (nopesan==1){
         e1 = new email(judul,pesan);
     }   else{
         e2 = new email(judul,pesan);
     }
    }

    public void kirimPesan(email e, String penerima){
        e.setKepada(penerima);
        System.out.println(getNamaplikasi());
        System.out.println("Pesan anda terkirim");
        System.out.println("Judul    = "+e.getSubjek());
        System.out.println("Isi Pesan = "+e.getPesan());
    }
```

Perwujudan Relasi Depedency

2. Penggunaan kelas B sebagai nilai balikan pada fungsi di kelas A

```
class B { ... }
class A
{
    B fungsiA(....) { ..... }
}
```

```java
public class email {
    private String subjek, pesan, kepada;

    public email(String subjek, String pesan) {
        this.subjek = subjek;
        this.pesan = pesan;
    }

    public String getSubjek() {
        return subjek;
    }

    public String getPesan() {
        return pesan;
    }

    public String getKepada() {
        return kepada;
    }

    public void setKepada(String kepada) {
        this.kepada = kepada;
    }

}
```

```java
public class Aplikasi {
    private String namaplikasi;
    private email e1,e2;

    public email getE1(){
        return e1;
    }
    public email getE2(){
       return e2;
    }
    public Aplikasi(String namaplikasi) {
        this.namaplikasi = namaplikasi;
    }

    public String getNamaplikasi() {
        return namaplikasi;
    }

    public void buatPesan(int nopesan, String judul, String pesan){
     if (nopesan==1){
         e1 = new email(judul,pesan);
     }   else{
         e2 = new email(judul,pesan);
     }
    }

    public void kirimPesan(email e, String penerima){
        e.setKepada(penerima);
        System.out.println(getNamaplikasi());
        System.out.println("Pesan anda terkirim");
        System.out.println("Judul    = "+e.getSubjek());
        System.out.println("Isi Pesan = "+e.getPesan());
    }
```

Perwujudan Relasi Depedency

3. Penggunaan kelas B sebagai variabel lokal pada fungsi di kelas A

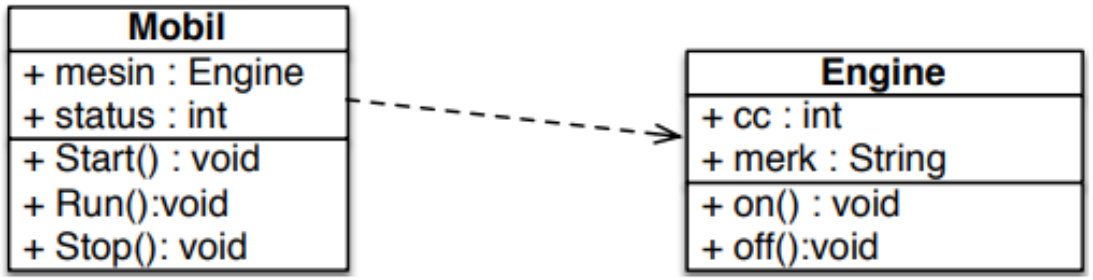
```
class B { ... }
class A
{
        void fungsiA(....)
{
        B varLokal;
 }
}
```

```
public class Jalan {
    public static void main (String [] args){
        Aplikasi app = new Aplikasi ("Yahoo Mail");
        app.buatPesan(2, "Greeting", "Hai Dian apakabar?");
        app.kirimPesan(app.getE2(), "Andi");

        System.out.println("");

        Aplikasi app2 = new Aplikasi ("Gmail");
        app2.buatPesan(1, "HGD", "Selamat kak semoga cepat dapat kerja");
        app2.kirimPesan(app2.getE1(), "Andi");

    }
}
```



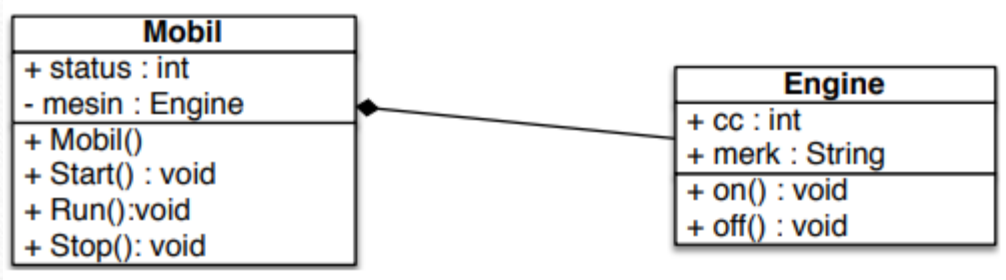

```
class Engine
  {
    public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON”)
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
  }
class Mobil
```

```
class Mobil
{
    public int status;
    public void Start(Engine e)
    {
      e.On();
    }
    public void Run()
    {  Console.WriteLine("Run...!");
    }
    public void Stop(Engine e)
    {
      e.Off();
    }
  }
```

Composition

- Composition merupakan relasi yang lebih spesifik dari relasi aggregation (Strong Aggregation)
- Objek dari kelas penyusun hanya ada selama objek kelas komposit ada
- Sehingga relasi has-a pada agregasi menjadi Relasi part-of

Composition

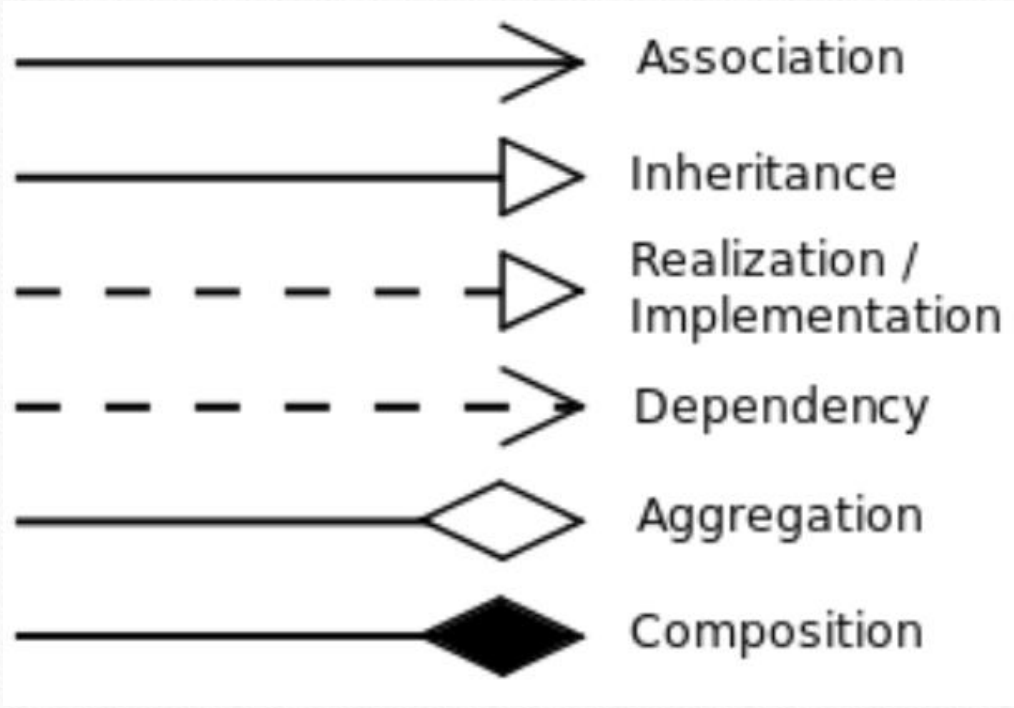



```
class Engine
{  public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON")
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
  }
```

```
class Program
{ static void Main(string[] args)
  { Mobil mobil = new Mobil();
  }
}
```

```
class Mobil
{  private Engine mesin;
    public int status;
    public Mobil()
    {  mesin=new Engine();
    }
    public void Start()
    {  mesin.On();
    }
    public void Run()
    {  Console.WriteLine("Run…!");
    }
    public void Stop()
    {  mesin.Off();
    }
}
```

Notasi UML Class Diagram



- Buatlah UML untuk kelas limas, kelas prisma dan kelas segitiga
- Buatlah UML untuk kelas kerucut dan kelas lingkaran
- Buatlah UML untuk kelas Kubus, Balok dan kelas segi empat
- Termasuk agregasi atau komposisi

- Berikan contoh dari relasi agregasi? Berikan contoh dari relasi asosiasi?
- Apakah perbedaan dari komposisi dan agregasi?
- Berdasarkan kasus bangun ruang tsb, bagaimana dengan relasinya komposisi atau agregasi?

```java
public class email {
    private String subjek, pesan, kepada;

    public email(String subjek, String pesan) {
        this.subjek = subjek;
        this.pesan = pesan;
    }

    public String getSubjek() {
        return subjek;
    }

    public String getPesan() {
        return pesan;
    }

    public String getKepada() {
        return kepada;
    }

    public void setKepada(String kepada) {
        this.kepada = kepada;
    }

}
```

```java
public class Aplikasi {
    private String namaplikasi;
    private email e1,e2;

    public email getE1(){
        return e1;
    }
    public email getE2(){
       return e2;
    }
    public Aplikasi(String namaplikasi) {
        this.namaplikasi = namaplikasi;
    }

    public String getNamaplikasi() {
        return namaplikasi;
    }

    public void buatPesan(int nopesan, String judul, String pesan){
     if (nopesan==1){
         e1 = new email(judul,pesan);
     }   else{
         e2 = new email(judul,pesan);
     }
    }

    public void kirimPesan(email e, String penerima){
        e.setKepada(penerima);
        System.out.println(getNamaplikasi());
        System.out.println("Pesan anda terkirim");
        System.out.println("Judul    = "+e.getSubjek());
        System.out.println("Isi Pesan = "+e.getPesan());
    }
```