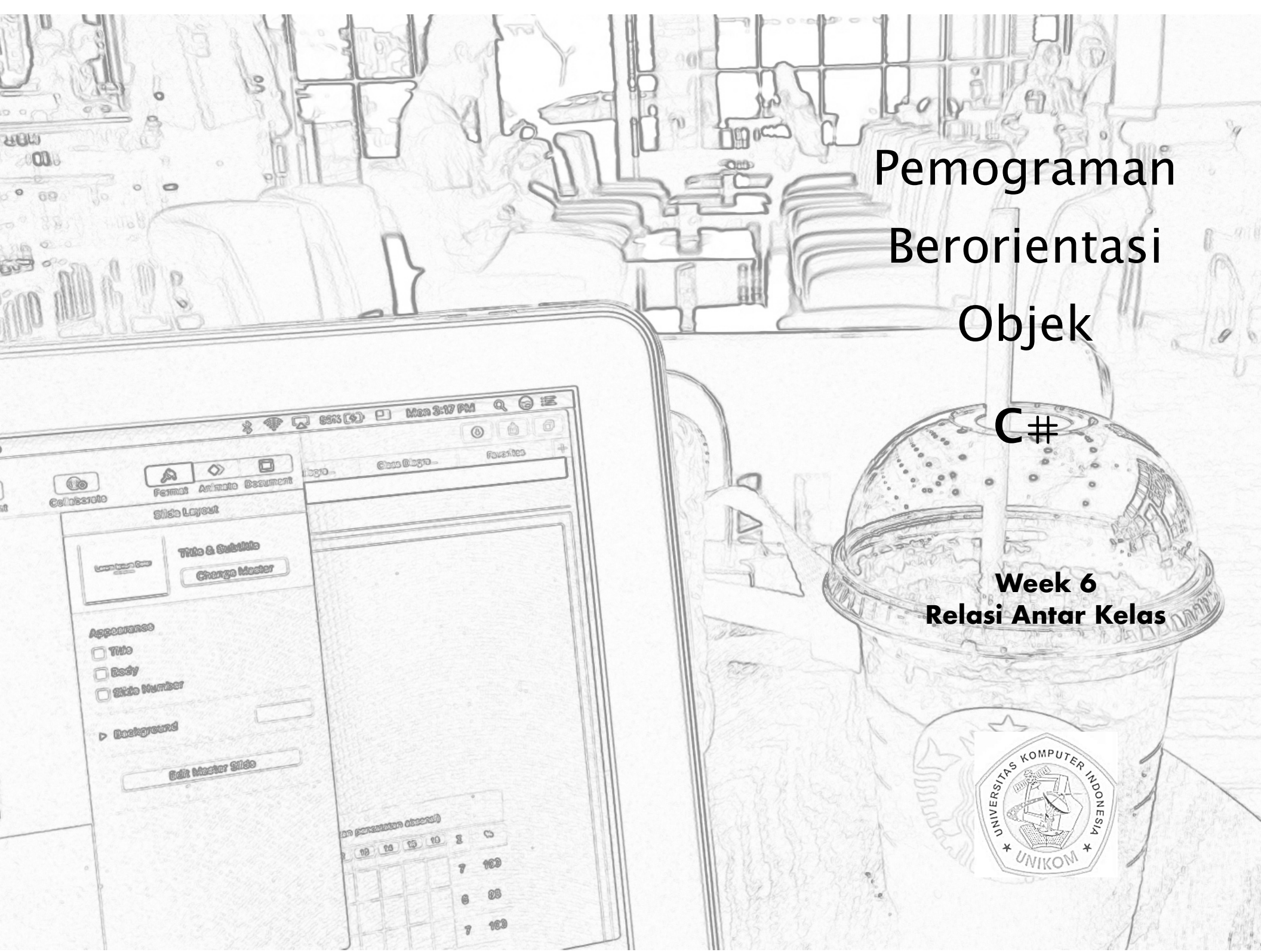
Pemograman Berorientasi Objek

C#

Week 6
Relasi Antar Kelas



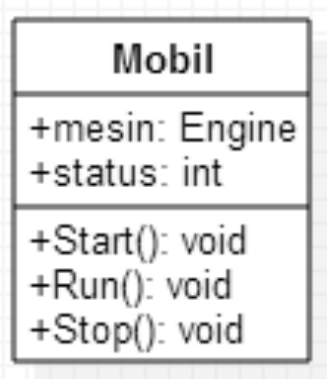
Relasi Antar Kelas

- 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.
- 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.
- 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.

Class Diagram

- Kelas dalam UML dimodelkan dalam bentuk persegi yang terdiri dari 3 bagian yaitu ***Nama Kelas, properti*** dan ***method*** yang dimiliki oleh kelas tersebut. Contoh kelas diagram dapat dilihat pada gambar dibawah ini.



- Contoh diatas merupakan kelas diagram untuk kelas Mobil yang memiliki 2 buah properti yaitu mesin dan status. Tipe data mesin adalah Engine dan tipe data status adalah integer. Tanda didepan properti merupakan akses level masing-masing properti.

- Simbol tersebut adalah

No	Simbol	Arti
1.	+	Public
2.	-	Private
3.	#	Protected

- Method yang dimiliki oleh kelas Mobil ada 3 yaitu Start(), Run(), dan Stop(). Masing-masing method tidak membutuhkan argumen dan tipe data kembalian dari method tersebut adalah void.

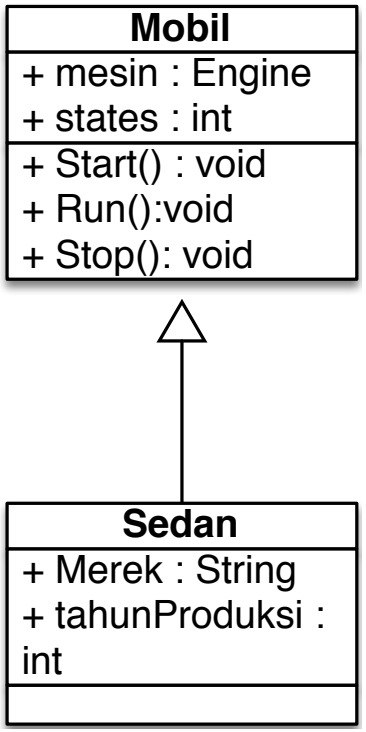
Jenis-jenis Relasi Antar Kelas

- Seperti yang telah dinyatakan sebelumnya, bahwa sebuah aplikasi yang dibangun dengan menggunakan paradigma OOP akan menggunakan banyak kelas. Kelas-kelas tersebut saling berhubungan antara satu dengan yang lainnya. Hal ini menimbulkan relasi antar kelas. Pembahasan selanjutnya akan membahas tentang 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.***

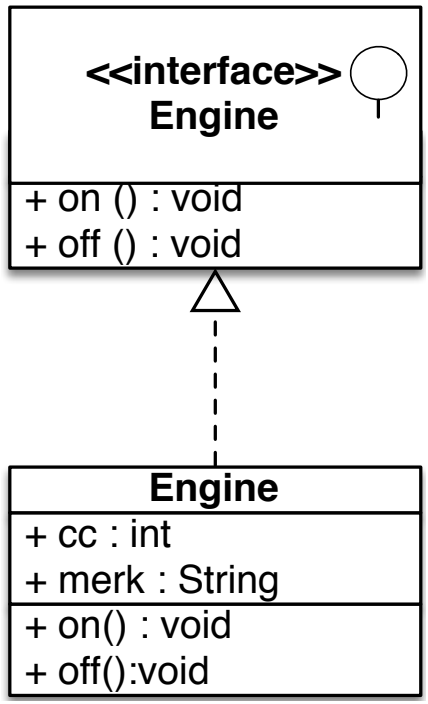
- Sedan turunan dari Mobil bisa juga disebut Sedan IS A Mobil



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.

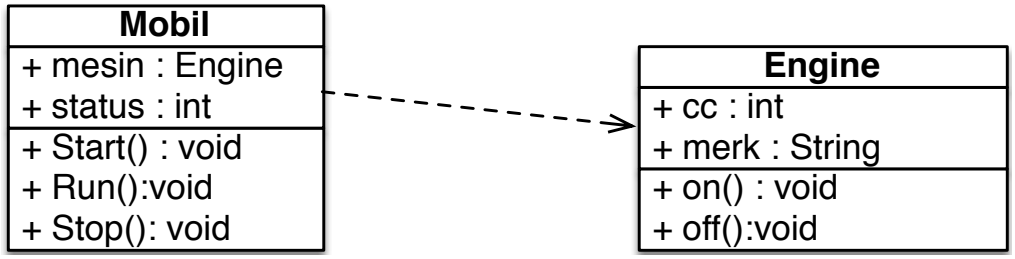
- Diagram kelas untuk realization dapat dilihat pada gambar dibawah ini. Pada gambar dibawah ini dapat dinyatakan bahwa kelas Engine mengimplementasikan interface IEngine.



Dependency

- Dependency merupakan relasi antar kelas dimana satu kelas membutuhkan atau tergantung kepada kelas lainnya. Tapi ketergantungan tersebut tidak timbal balik.
- Relasi dependency ini digambarkan dengan panah yang dari satu kelas ke kelas lainnya. Arah panah menunjukkan kelas yang dibutuhkan.
- Contoh pada kelas Mobil dan Engine.

- Mobil membutuhkan Engine sehingga relasi kelas Mobil dan Mesin dapat dilihat pada Gambar dibawah ini.



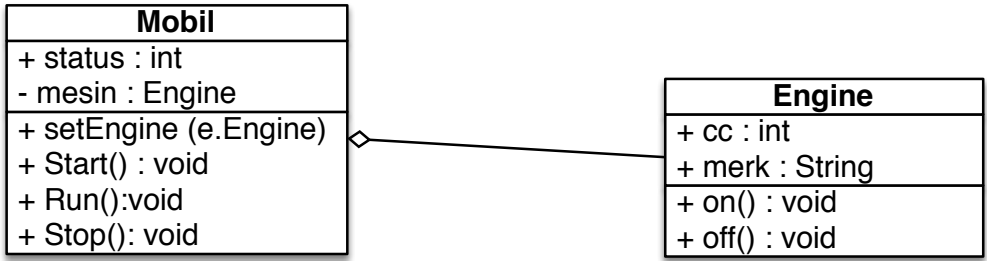
- Pada contoh diatas dapat dilihat bahwa kelas Mobil membutuhkan objek dari kelas Engine. Hal ini bisa dilihat dari method Start dan Stop yang dimiliki oleh kelas Mobil. Kedua method tersebut membutuhkan argumen berupa objek dari kelas Engine. Objek tersebut selanjutnya nanti digunakan dalam method tersebut dengan mengeksekusi method yang ada dalam objek Engine.

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

Aggregation

- Relasi aggregation merupakan bentuk khusus dari relasi dependency. Pada relasi dependency tidak ada dinyatakan kepemilikan kelas Engine.
- Pada relasi aggregation, terdapat kepemilikan kelas Engine semisal terdapat sebuah properti yang memiliki tipe Engine. Namun pada relasi ini tidak diatur siklus hidup dari kelas Engine. Objek dari kelas Engine dimiliki oleh kelas Mobil dan disimpan dalam properti yang memiliki tipe Engine ini.
- ***Relasi aggregation sering juga disebut relasi HAS-A.***

- Contoh relasi aggregation pada antara Mobil dan Engine dapat dilihat pada gambar dibawah ini.



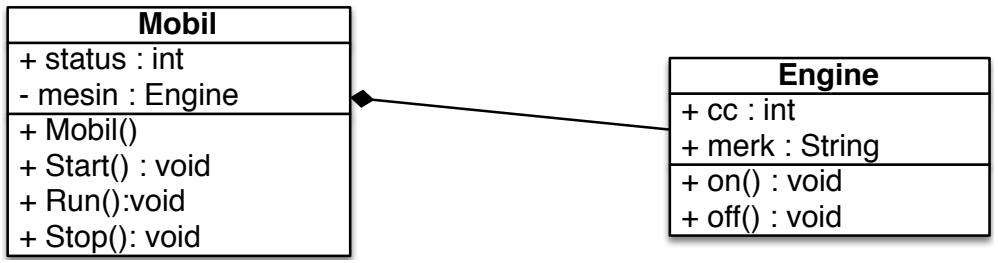
- Pada contoh diatas, dapat dilihat bahwa kelas Mobil memiliki properti mesin yang bertipe Engine. Objek dari kelas Engine nantinya akan disimpan dalam properti mesin tersebut. Contoh diatas terlihat bahwa kelas Mobil memiliki kelas Engine.
- Pada contoh diatas objek dari kelas Engine dibuat di luar kelas Mobil. Artinya siklus hidup dari kelas Engine tidak tergantung pada kelas Mobil.

```
class Engine
{  public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON”)
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
  }
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 Program
{ static void Main(string[] args)
  { Engine engine = new Engine();
    Mobil mobil = new Mobil();
    mobil.setEngine(engine);
  }
}
```

Composition

- Composition merupakan relasi yang lebih spesifik dari relasi aggregation.
- Pada relasi ini suatu kelas tidak hanya dimiliki oleh kelas lainnya, tapi juga siklus hidup kelas tersebut juga ditentukan oleh kelas yang memilikinya.
- Pada relasi ini biasanya objek dari kelas yang dimiliki diciptakan di dalam kelas yang memilikinya.

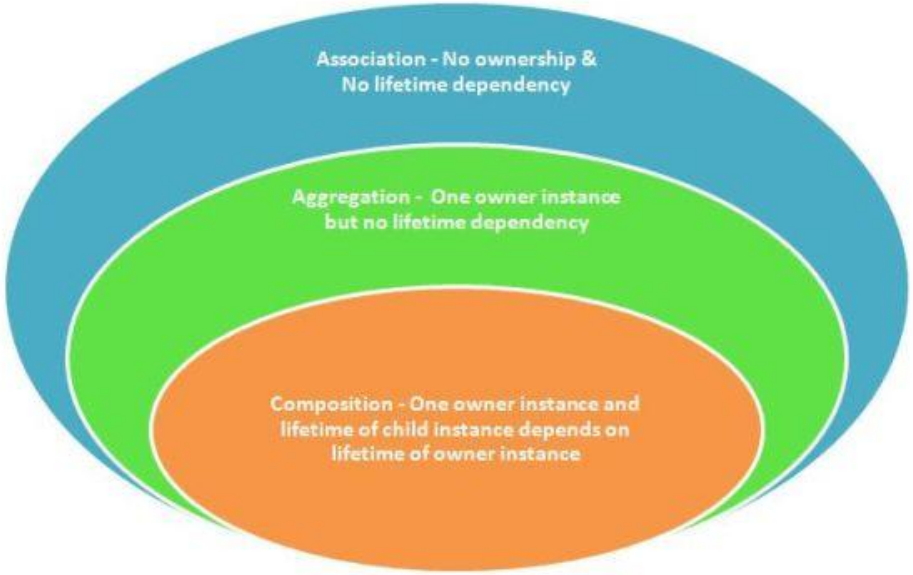
- Contoh relasi composition dapat dilihat pada gambar dibawah ini. Pada gambar dibawah ini dapat dilihat bahwa kelas Engine dimiliki dan dikontrol sepenuhnya oleh kelas Mobil. Relasi composition ini digambarkan dengan tanda diamond bold. Ujung diamond bold menunjukkan bahwa kelas tersebut memiliki kelas yang ada diujung lainnya.




```
class Engine
{  public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON”)
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
  }
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();
    }
}
class Program
{ static void Main(string[] args)
  { Mobil mobil = new Mobil();
  }
}
```

Association - Aggregation - Composition

- Jika disimpulkan antara relasi association, aggregation dan composition maka dapat disimpulkan hubungan ketiga jenis relasi tersebut dapat digambarkan melalui gambar berikut ini.



- Diagram tersebut menggambarkan bahwa relasi Composition merupakan bentuk khusus dari relasi aggregation dan relasi aggregation merupakan bentuk khusus dari relasi association.

Notasi UML Class Diagram