




# Relasi Class

Pemrograman Berorientasi Object



# Kompetensi

- 
- Memahami konsep relasi kelas;
  - Memahami jenis-jenis relasi kelas dalam program.

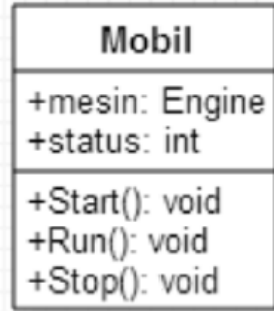
## Relasi Antar Class

- Dalam paradigma pemrograman berorientasi objek, sebuah aplikasi dibangun dengan menggabungkan beberapa kelas.
- Kelas-kelas tersebut saling bekerja sama untuk menyelesaikan suatu masalah.
- Dalam aplikasi yang berukuran cukup kompleks, banyak kelas yang terlibat dalam aplikasi tersebut
- Maka untuk aplikasi yang kompleks dibutuhkan pemodelan kelas untuk menggambarkan aplikasi yang dibangun.
- Tools yang digunakan untuk memodelkan kelas-kelas dalam OOP/PBO adalah Unified Modelling Language (UML)

- UML merupakan spesifikasi pemodelan yang paling banyak digunakan untuk memodelkan struktur dan perilaku aplikasi.
- UML juga digunakan untuk meodelkan 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 diagram 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:



- 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 di depan 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 Class

Terdapat beberapa macam relasi antar class yaitu:

- Dependence (“uses-a”)
- Aggregation (“has-a”)
- Inheritance (“is-a”)

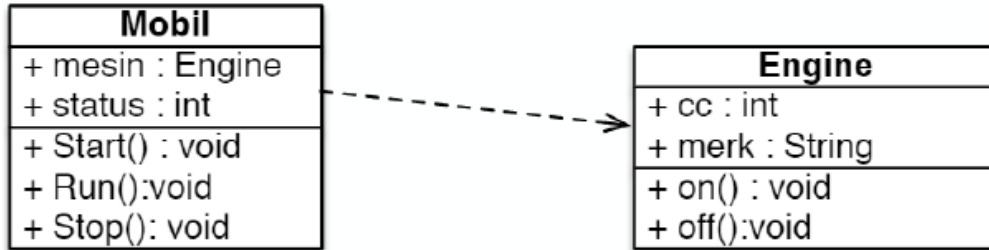
(Horstmann dan Cornell, 2008)

## Dependence (“uses-a”)

- Dependence merupakan relasi antar class 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 yang lainnya. Arah panah menunjukkan kelas yang dibutuhkan.
- Contoh pada kelas Mobil dan Engine



- Mobil membutuhkan Engine, sehingga relasi class Mobil dan Engine adalah:



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

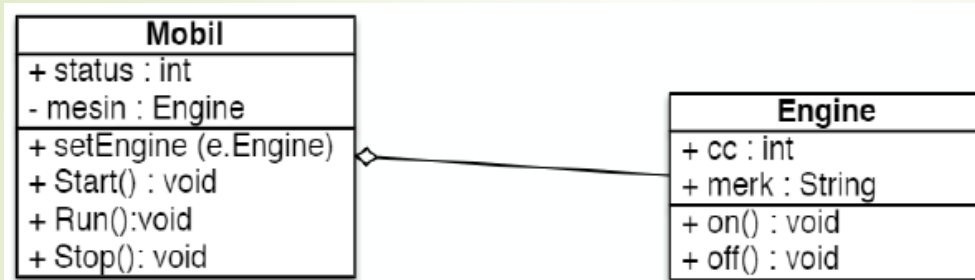
```
class Engine
{
    public int cc;
    public String merek;
    public void On()
    { Console.WriteLine("Mesin ON")
    }
    public void Off()
    { Console.WriteLine("Mesin OFF");
    }
}

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 (“has-a”)

- Relasi aggregation merupakan bentuk khusus dari relasi dependence. Pada relasi dependence 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

- Contoh relasi aggregation pada Mobil dan Engine dapat dilihat pada gambar berikut:



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

```

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

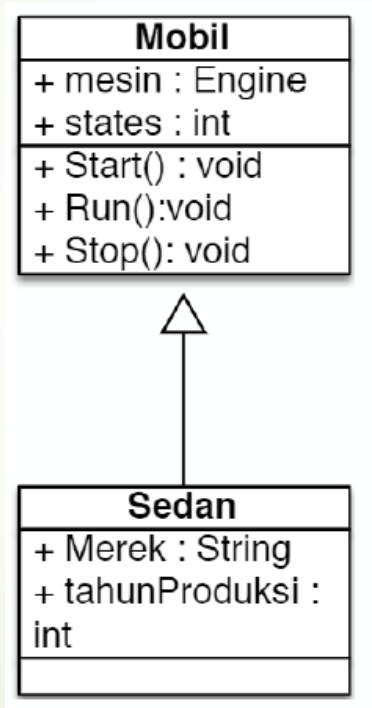
class Mobil
{
    private Engine mesin;
    public int status;
    public void setEngine(Engine e)
    { mesin=e;
    }
    public void Start()
    { mesin.On();
    }
    public void Run()
    { Console.WriteLine("Run...!");
    }
    public void Stop()
    { mesin.Off();
    }
}

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



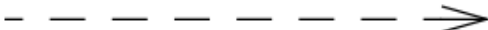
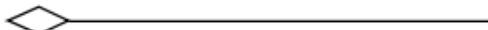


# Inheritance (“is-a”)

- Inheritance merupakan relasi turunan dimana sebuah class diciptakan berdasarkan class lainnya. Class yang diciptakan disebut dengan class anak dan class asalnya disebut dengan class induk
- Class anak akan mewarisi seluruh method dan properti yang dimiliki class induknya.
- Berikut contohnya:

- Sedan turunan dari Mobil dapat juga disebut Sedan **is a** Mobil

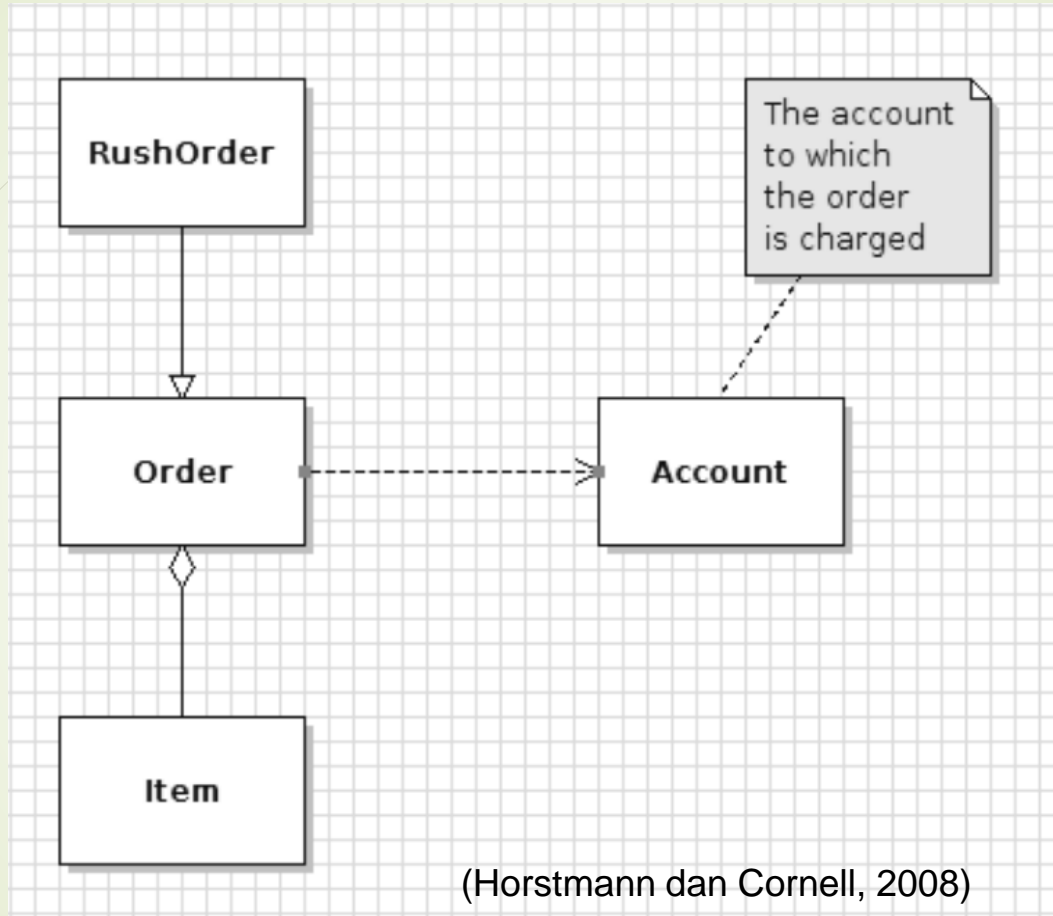


# Notasi UML Relasi Kelas

Relationship	UML Connector
Inheritance	
Interface inheritance	
Dependency	
Aggregation	
Association	
Directed association	

(Horstmann dan Cornell, 2008)

## Contoh Relasi dalam Diagram Kelas



(Horstmann dan Cornell, 2008)

## UML Tools

Kakas untuk membantu dalam merancang diagram UML



**Visual Paradigm + NetBeans**

[Visual-paradigm.com/tutorials/modelinginnetbeans.jsp](http://Visual-paradigm.com/tutorials/modelinginnetbeans.jsp)



## UML Tools

Kakas untuk membantu dalam merancang diagram UML



**ArgoUML**

[argouml.trigis.org](http://argouml.trigis.org)

## UML Tools

Kakas untuk membantu dalam merancang diagram UML



**Violet**

[violet.sourceforge.net](http://violet.sourceforge.net)





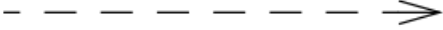


---

Thank You

---

# TUGAS

- Buat artikel singkat terkait relasi class yang belum ada pada slide ini

Relationship	UML Connector
Inheritance	
Interface inheritance	
Dependency	
Aggregation	
Association	
Directed association	