Pengembangan Aplikasi Perangkat Lunak

OOAD

Class Diagram & Activity Diagram

By: Augury El Rayeb (AER)

Tujuan Pertemuan

- Memahami apa yang dimaksud dengan logical system models
- Memahami pendekatan pemodelan
- Memahami kegunaan class diagram
- Mampu membuat class diagram

System Development Process

Phase	Actions	Outcome
Initiation	Raising a business need	Business documents
Analysis	Interviewing stakeholders, exploring the system environment	Organized documentation
Specification	Analyze the engineering aspect of the system, building system concepts	Logical System Model
Implementation	Program, build, unit-testing, integrate, documentation	Testable system
Testing & Integration	Integrate all components, verification, validation, installation, guidance	Testing results, Working sys
Maintenance	Bug fixes, modifications, adaptation	System versions

Logical System Models

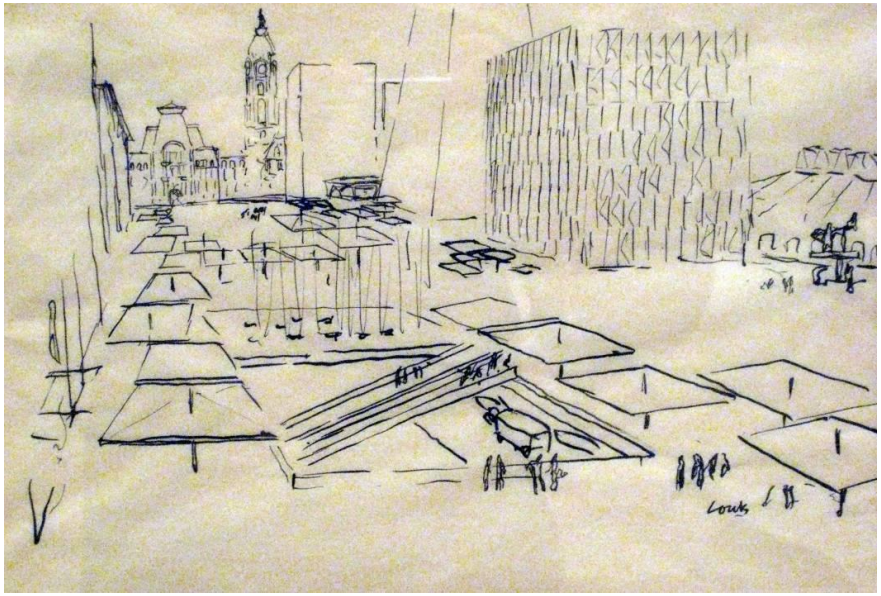
- Logical system models menggambarkan tentang sistem atau apa yang harus dilakukan sistem – bukan tentang bagaimana sistem akan diimplementasikan.
- model tersebut mungkin terdiri dari:
  - Data models (e.g., class diagram)
  - Process models (e.g., sequence diagram)
  - User interaction models (e.g., use case)
- Logical system models digambarkan dengan bahasa pemodelan
  - Formal (or partially formal)
  - Understandable (visual or textual)

Pendekatan Pemodelan

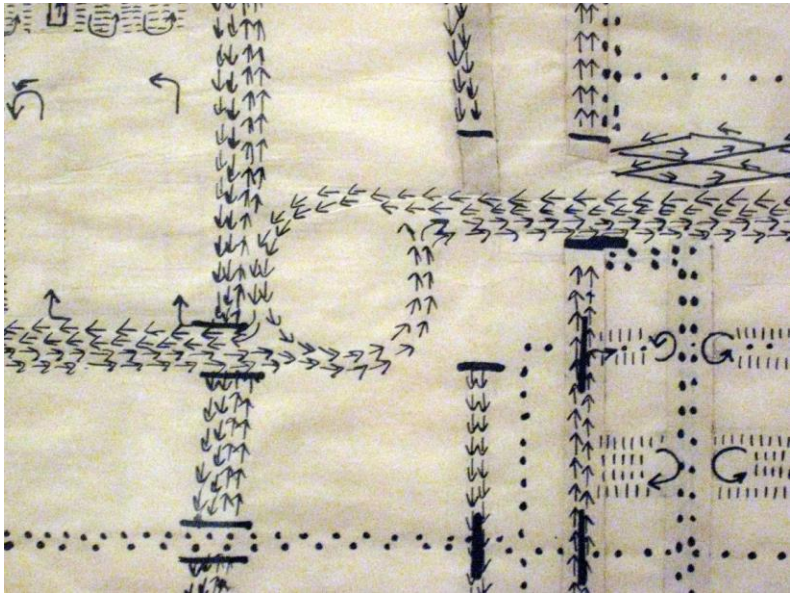
- Pendekatan pemodelan berbeda satu sama lain, tergantung sudut pandangnya.

Structural Models	**Behavior Models**
Berfokus pada menggambarkan struktur dari sistem - elemen yang tetap dan tidak berubah dari waktu ke waktu	Berfokus pada dinamika sistem: Bagaimana ia berubah seiring waktu.

Contoh Pemodelan



Plans for Philadelphia's City Center (1953)

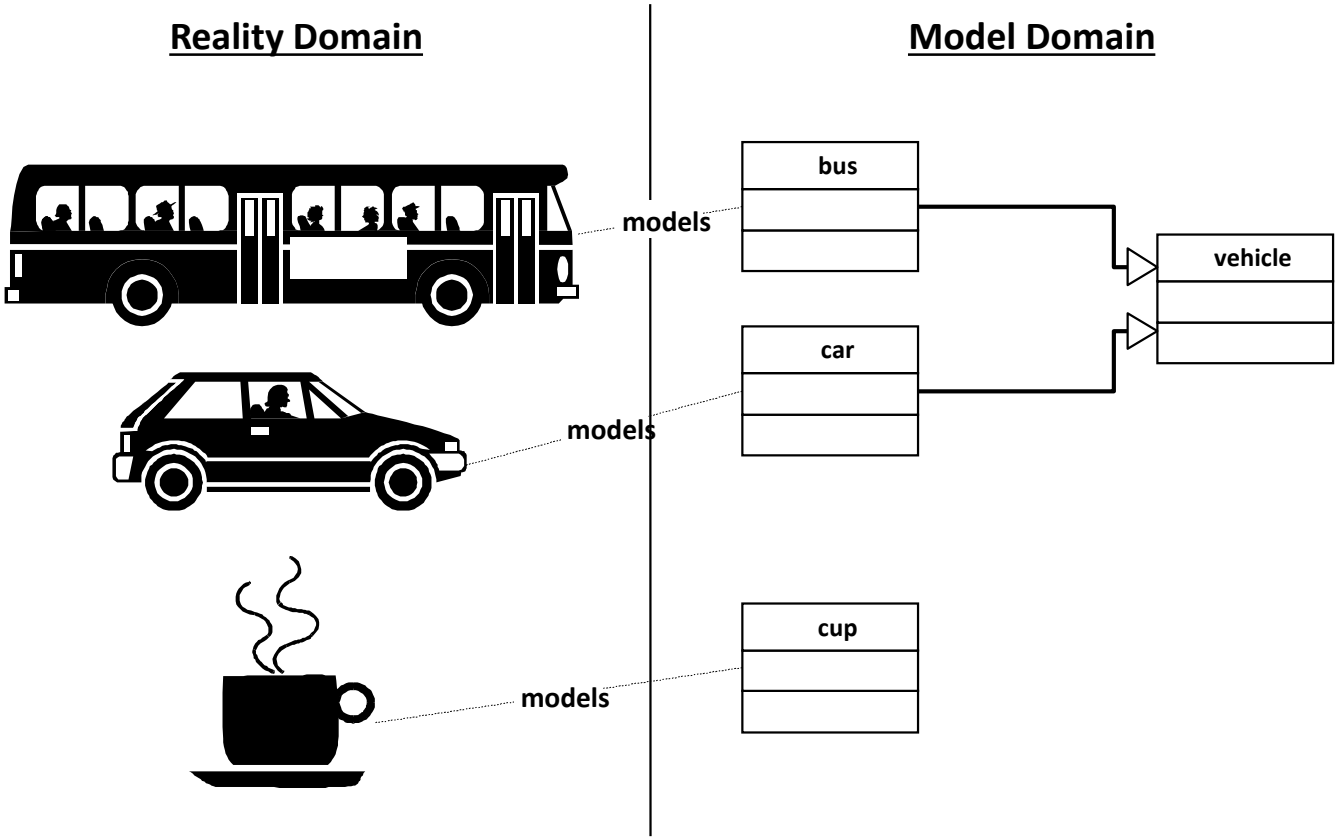


Models of Philadelphia's Traffic Behavior (1953)

Classes, attributes and operations

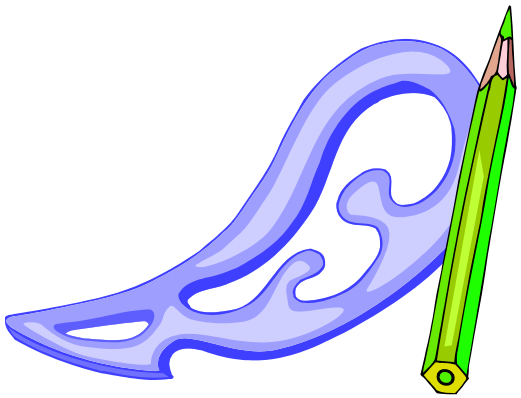
Pendekatan Object-Oriented

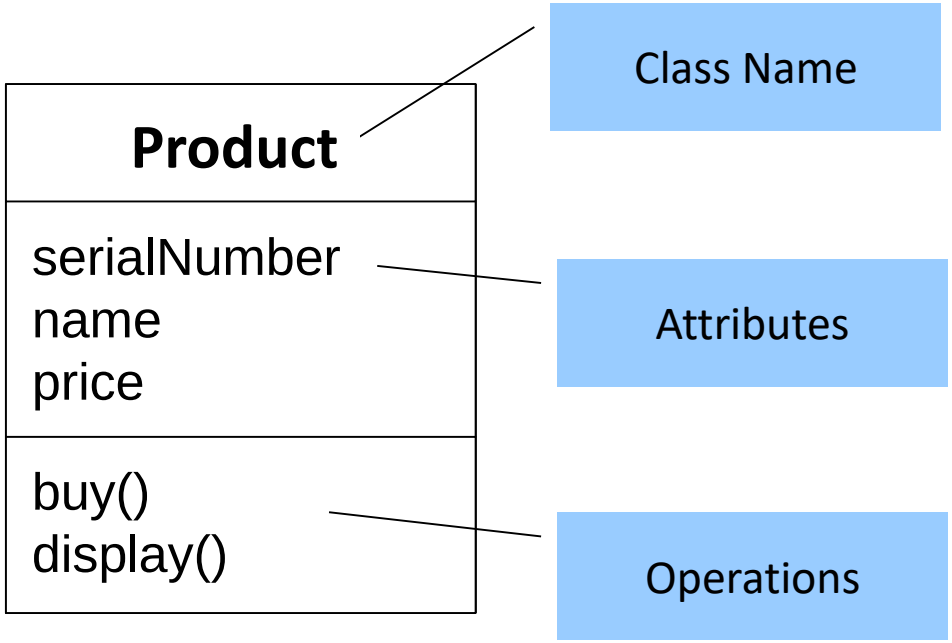
- **Object** adalah abstraksi dari entitas dunia nyata atau sistem.



Classes

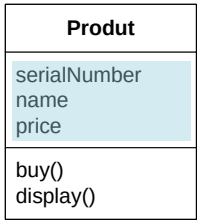
- Sebuah class adalah suatu template dari object yang sebenarnya, disimpan, dan sebagai contoh





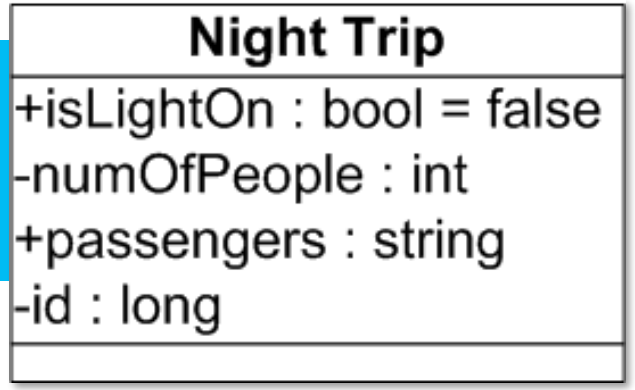
Attribute – Penulisan

[visibility] **name** [[multiplicity]] [: type] [=initial value] [{property}]



- **visibility**: Hak akses terhadap attribute
  - Symbol yang digunakan: +, #, -, ~
- **multiplicity**: Terdapat berapa instances dari attribute:
  - Contoh: middleName [0..1] : String, phoneNumber [1..*]
- **Type**: type dari attribute (integer, String, Person, Course)
- **initial value**: nilai inisial/default dari attribute.
  - Contoh: salary : Real = 10000, position : Point = (0,0)
- **property**: properties/sifat standar dari attribute
  - Contoh: Changeable, readOnly, addOnly, frozen (C++: const, Java: final)

Attribute - Visibility

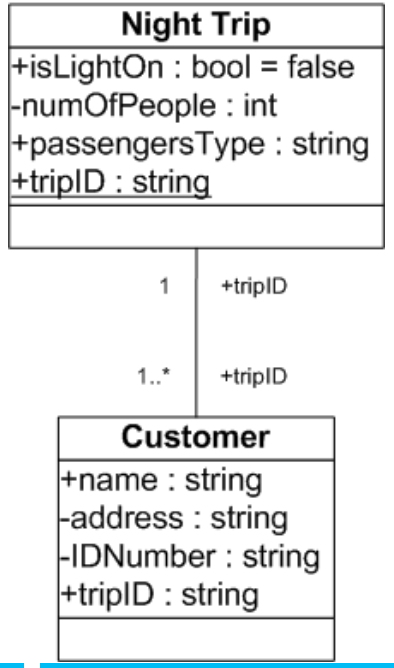


- Ditandai melalui simbol seperti berikut:

Symbol	Arti	Penjelasan
+	**Public**	visibilitas yang menunjukkan bahwa anggota kelas tersebut dapat diakses oleh kelas lain sekalipun tidak mewarisi kelas yang bersangkutan namun masih memiliki relasi dengan kelas tersebut
-	**Private**	menunjukkan bahwa anggota kelas tersebut hanya dapat dipanggil oleh kelas yang bersangkutan dan tidak dapat dipanggil oleh kelas lain.
#	**Protected**	menunjukkan bahwa anggota kelas tersebut hanya dapat dipanggil oleh kelas yang bersangkutan dan kelas anak yang mewarisinya.
~	**Package**	menunjukkan atribut tersebut dapat dilihat oleh kelas lain yang masih terdapat dalam paket yang sama.

Attribute - Multiplicity

- Multiplicity atau Cardinalitas adalah simbol yang menunjukkan jumlah instansiasi dari satu kelas terkait dengan kelas lainnya. (*detilnya pada bagian relation*)
- Biasanya berguna saat pembuatan relasi antar class
- Penulisan multiplicity:
  - **1** tidak lebih dari satu(tepat satu)
  - **0..1** *nol atau tepat satu*
  - ***** banyak
  - **0..*** *nol atau banyak*
  - **1..*** *tepat satu atau banyak*



Operations – Penulisan

[visibility] [direction] **name** [(parameter-list)] [: return-type] [{property}]

- Operasi dapat memiliki nol atau lebih parameter, masing-masing memiliki sintaks :
  - [direction] name : type [=default-value]
    - **Direction**: in (paremeter input – tidak bisa dimodifikasi),
      out (parameter output – dapat dimodifikasi),
      inout (keduanya, dapat dimodifikasi)
- Property:
  - {leaf} – operasi kongkrit
  - {abstract} – tidak bisa dipanggil secara langsung
  - {isQuery} – operasi akhir keadaan dimana operasi tidak berubah
  - …

Operation – Contoh parameter

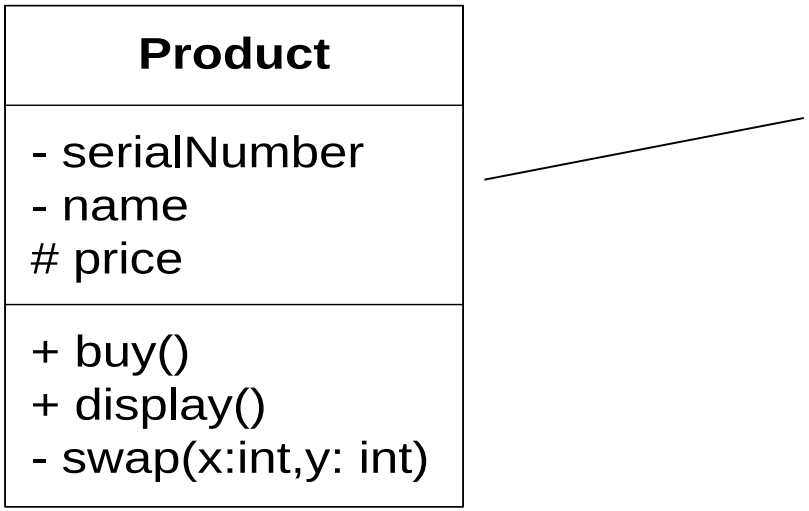
```
+ isLightOn() : boolean
+ addColor(newColor : Color)
+ addColor(newColor : Color) : void
# convertToPoint(x : int, y : int) : Point
- changeItem([in] key : string, [out] newItem :
      Item) : int
```

What's the difference?

Night Trip
+isLightOn : bool = false -numOfPeople : int +passengersType : string +tripID : string
+checkLightStatus() : bool -readNumOfPeople() : int -readPassengersType() : string #changeTripID(in id : string) : void

Operation – Visibility

- public (**+**) → object eksternal dapat mengakses member
- private (**-**) – Hanya internal method yang bisa mengakses member
- protected (#) – Hanya internal method, atau method dari object khusus dapat mengakses member

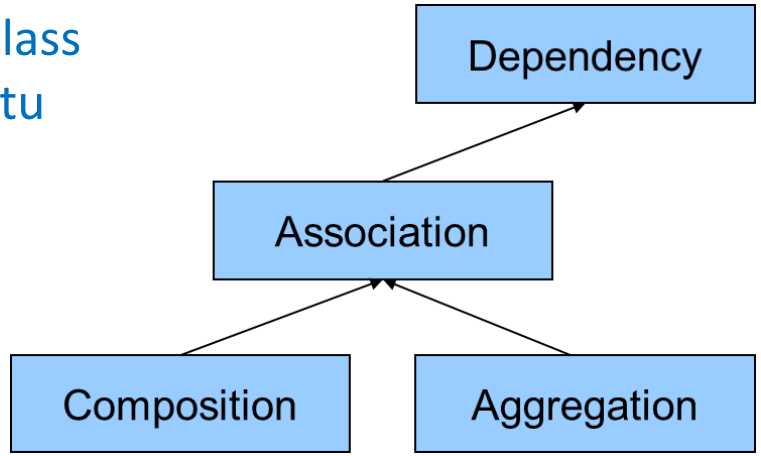


Kita akan mencoba untuk menjaga visibilitas seminimal mungkin

Relations

- Relation merupakan representasi jalur komunikasi antar object atau class
- Terdapat 3 tipe relasi pad UML:
  - **Dependency** → koneksi ketergantungan (satu arah), dimana perubahan pada suatu class akan mempengaruhi class yang tergantung padanya.
  - **Association** → koneksi dua arah antar class
  - **Aggregation** → menunjukkan bahwa satu class sebagai part class merupakan subordinat (bagian) dari class lain yang sebagai whole class.

    Namun jika whole class tidak ada, maka part class masih bisa berdiri sendiri.
  - **Composition** →merupakan strong aggregation, yaitu; kalau whole class tidak ada maka part class juga tidak boleh ada.



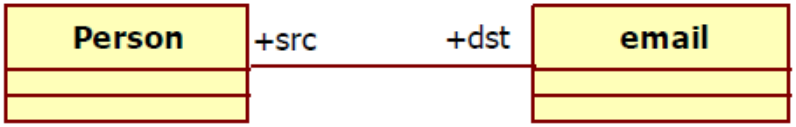
Relation - Dependency

- **Dependency**: Merupakan hubungan ketergantungan antar kelas. Suatu kelas memiliki ketergantungan terhadap kelas lain, tetapi tidak berlaku sebaliknya.

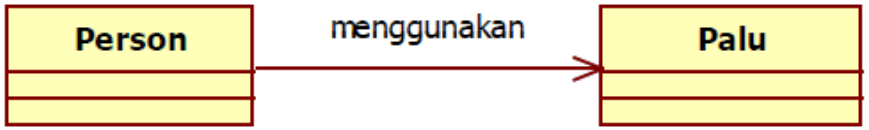
- Perhatikan diagram di atas!

  Perubahan pada class ***supplier*** akan berdampak pada class ***client***.

Relation - Association



Association



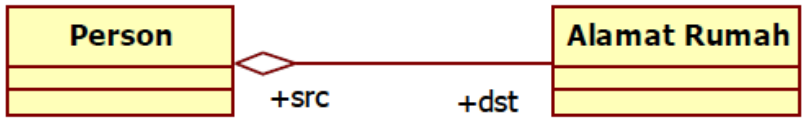
Directed Association

- **Association** menggambarkan koneksi dua arah antar dua class. Dapat diartikan sebagai relasi ***".. has a.."***

  (Keterangan: +**src** maksudnya source dan +**dst** maksudnya destination)

- **Directed Association** hampir sama dengan association, namun **menunjukkan aliran kejadian berasal dari salah satu class, sedang class yang lainnya bersifat pasif**

Relation – Aggregation & Composition



Aggregation

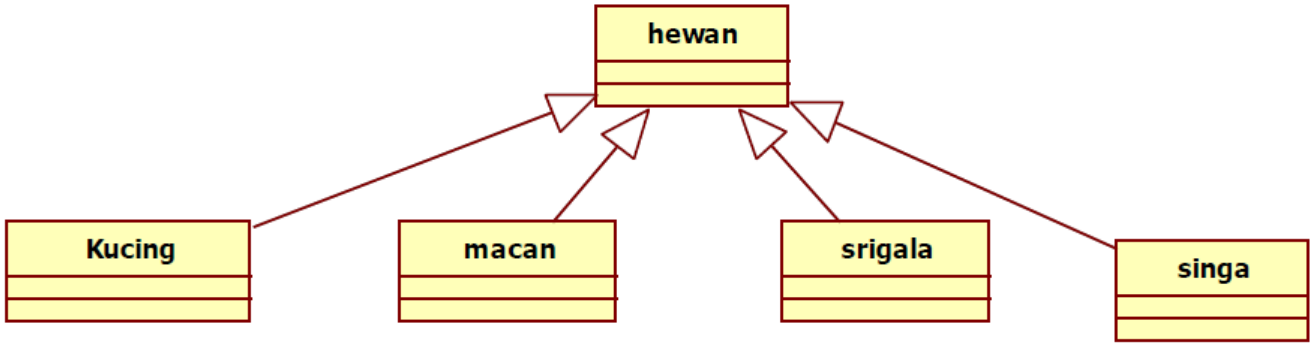
Composition

- **Aggregation** apabila class person dihilangkan, maka class alamat rumah masih dapat berdiri sendiri .

  (Keterangan: +**src** maksudnya source dan +**dst** maksudnya destination)

- **Composition** Apabila class mobil dihilangkan, maka classs tangki BBM mobil juga dihilangkan.

Relation lainnya - Generalization



Generalization

- Dapat diartikan sebagai relasi ***"..is a.."***

  Digunakan untuk merepresentasikan ***pewarisan*** (***inheritance***).

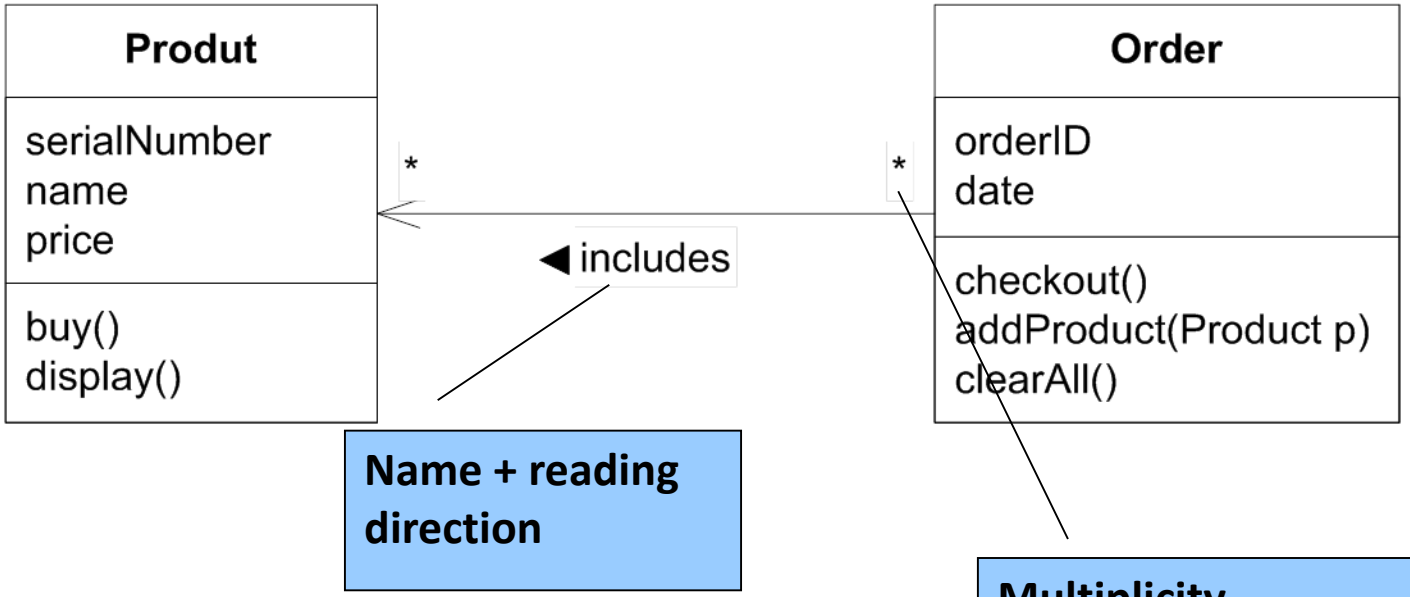
  Suatu class (***child class***) dapat diturunkan dari class lain dan mewarisi semua atribut dan method induknya (***parent class***) dan dapat menambah method atau atribut baru.

Relation lainnya - Realization

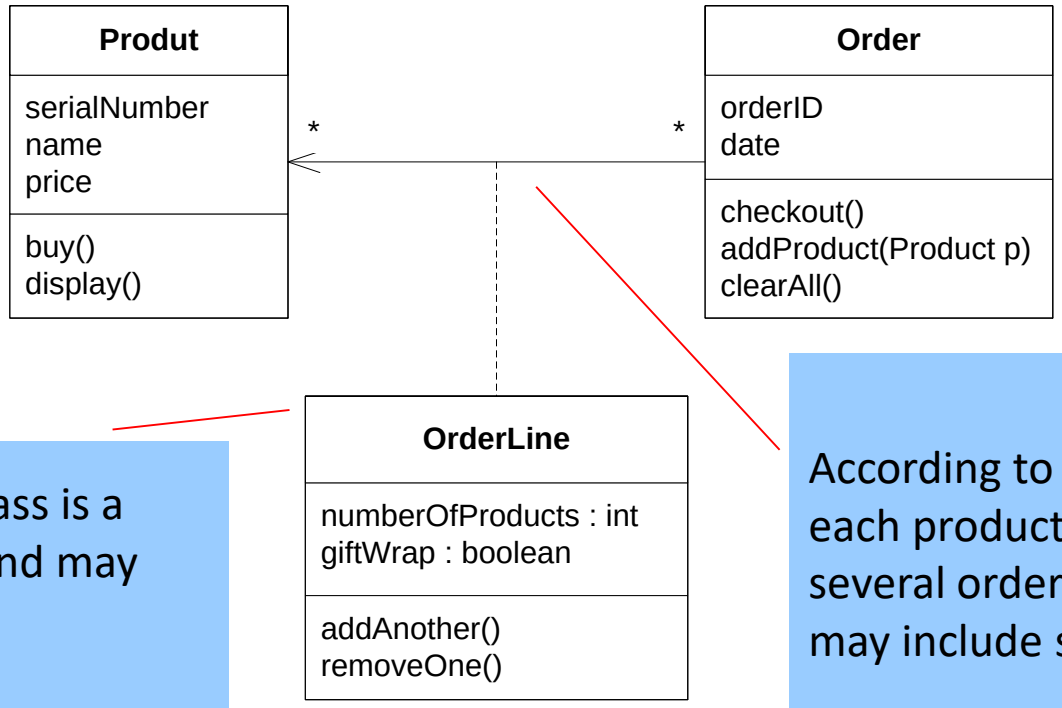
Realization

- sebuah relasi antar dua class yang mengharuskan class yang satu harus mengikuti aturan dari class yang lainnya. Biasanya terjadi antara kelas dengan antarmuka (*interface*)

Relation – Contoh lain



Relation – Contoh lain



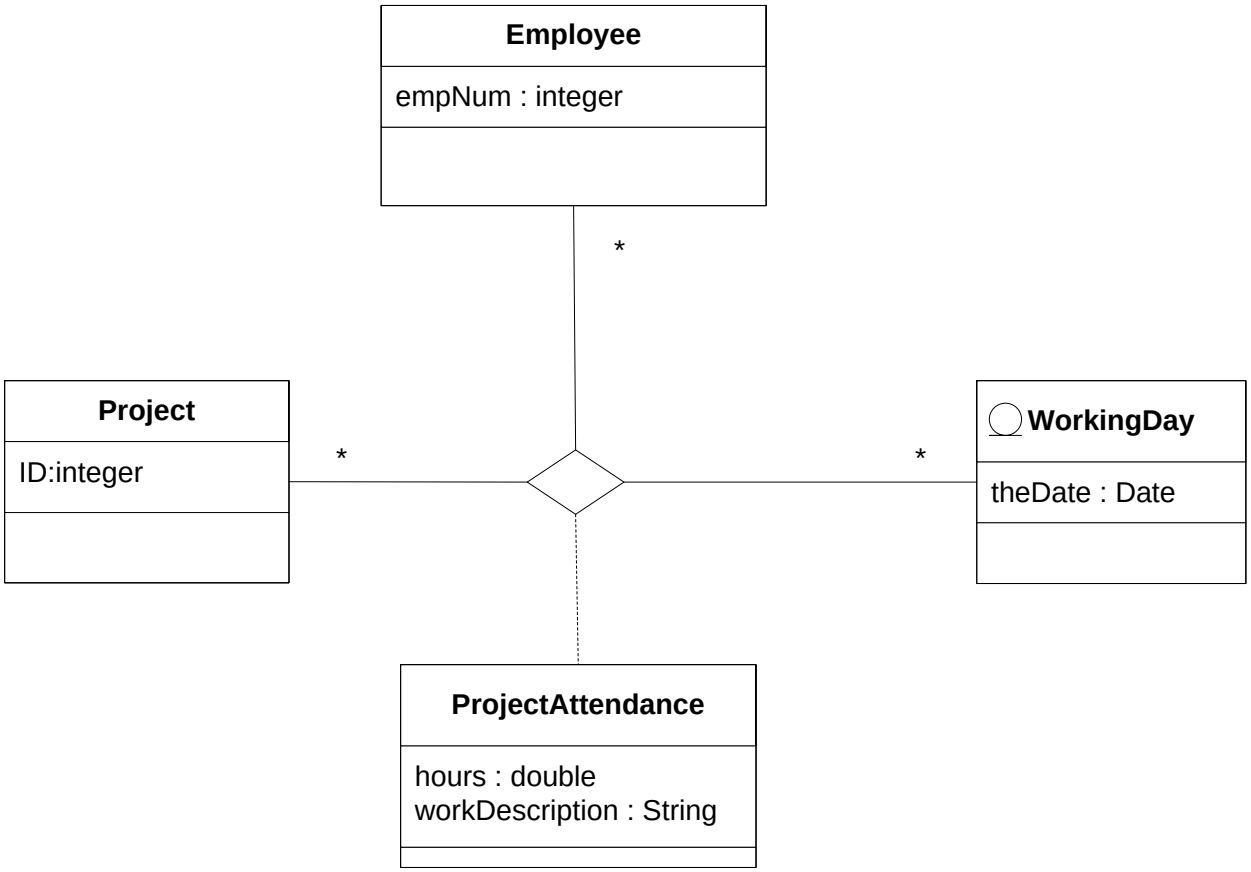
An association class is a "normal" class, and may include relations, inheritance etc.

According to the requirements, each product can appear in several orders, and each order may include several products

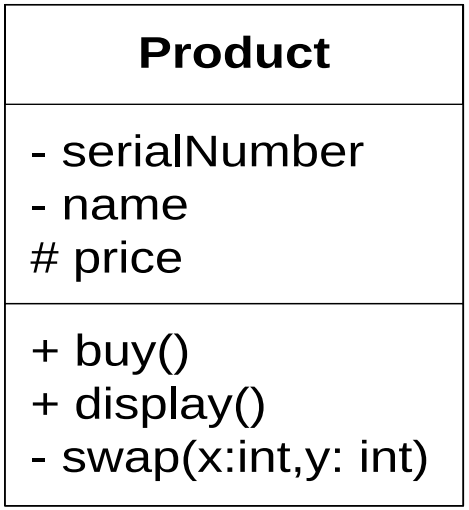
- Perhatikan sebuah class dipasangkan pada association, dan menspesifikasi association

Relation – Contoh lain

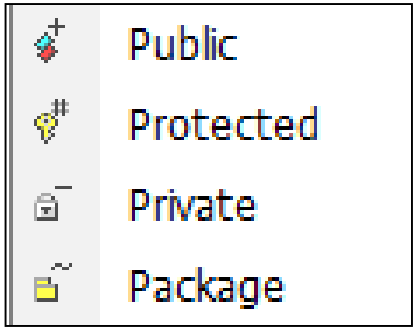
- Ternary Associations



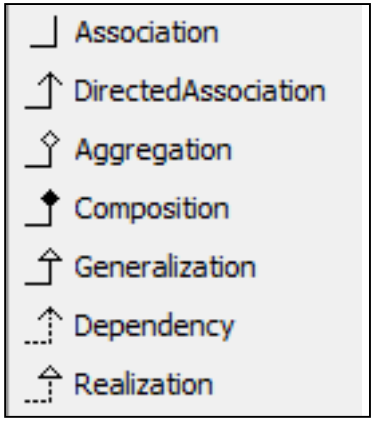
Class – Notation or Symbols



Class



Visibility



Relations

1	*tidak lebih dari satu(tepat satu)*
0..1	*nol atau tepat satu*
*****	*banyak*
0..*	*nol atau banyak*
1..*	*tepat satu atau banyak*

Multiplicity

Thank's

- See Ya Next Week