UNIKOM
Universitas Komputer Indonesia
Quality is Our Tradition

Analisis & Perancangan Sistem Informasi – 2

(APSI – 2)

CLASS DIAGRAM & OBJECT DIAGRAM

Annisa Paramitha F., S.Kom.,M.Kom

Prodi Sistem Informasi & Manajemen Informatika

UNIKOM

Definisi *Class Diagram*

Diagram kelas adalah diagram struktur UML yang menunjukkan struktur sistem yang dirancang pada tingkat kelas dan antarmuka, menunjukkan fitur, *constraint*, dan relasi : asosiasi, generalisasi, dependensi, dll. (https://www.uml-diagrams.org/class-diagrams-overview.html)

Class Diagram adalah diagram statis, mewakili pandangan statis dari suatu aplikasi. ***Class diagram*** tidak hanya digunakan untuk memvisualisasikan, menggambarkan, dan mendokumentasikan aspek berbagai sistem tetapi juga untuk membangun kode eksekusi *(executable code)* dari perangkat lunak

Class Diagram menggambarkan atribut, *operation,* dan juga *constraint* yang terjadi pada sistem. *Class diagram* banyak digunakan dalam pemodelan sistem OO karena merupakan satu –satunya diagram UML , yang dapat dipetakan lagnsung dengan bahasa berorientasi objek. Oleh karena itu banyak digunakan pada saat *coding*

Tujuan *Class Diagram*

Tujuan dari *Class Diagram* adalah untuk memodelkan pandangan statis suatu aplikasi.
Secara lebih rinci, tujuan dari ***class diagram*** adalah :

- Analisis dan desain statis aplikasi.
- Menjelaskan tanggung jawab sistem
- Basis untuk diagram komponen dan penyebaran (*deployment*)

Hal – hal berikut yang harus diingat saat menggambar *class diagram* :

- Nama ***class diagram*** harus memiliki mankan untuk menggambarkan aspek sistem
- Setiap elemen dan relasi harus diidentifikasi sebelumnya.
- Atribut dan *operation* dari masing – masing *class* harus diidentifikasi secara jelas.

Relasi pada *Class Diagram*

- **Association**
  Hubungan statis antar class. Pada umumnya menggambarkan class yang memiliki atribut berupa class lain, atau class yang harus mengetahui eksistensi class lain.

- ***Agregation***
  Hubungan secara keseluruhan antara aggregate class dengan component class.

- ***Inheritance dan Generalization***
  Inheritance adalah hubungan hirarkis antar class. ***Class*** dapat diturunkan dari ***class*** lain dan mewarisi semua atribut dan metode ***class*** asalnya dan menambahkan fungsionalitas baru, sehingga ia disebut anak dari ***class*** yang mewarisinya. Kebalikan dari inheritance adalah ***Generalitation*** yang merupakan hubungan taksonomi antara class yang lebih umum dengan class yang lebih khusus.

Simbol	Deskripsi
Asosiasi/ assocoation 	Relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan *multiplicity*
Generalisasi 	Relasi antarkelas dengan makna generalisasi-spesialisasi (umum-khusus)
Kebergantungan/ dependency 	Relasi antarkelas dengan makna kebergantungan antarkelas
Agregasi/ agregation 	Relasi antarkelas dengan makna whole-part

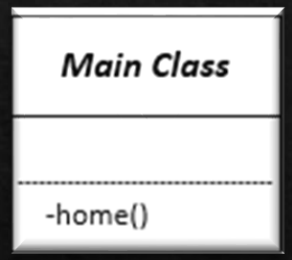
Struktur pada *Class Diagram*

Class yang ada pada struktur sistem harus dapat melakukan fungsi – fungsi sesuai dengan kebutuhan sistem, sehingga pembuat perangkat lunak atau programmer dapat membuat ***class*** didalam program sesuai dengan perancangan ***class diagram***.

Susunan kelas yang baik pada diagram kelas sebaiknya memiliki jenis- jenis kelas berikut:

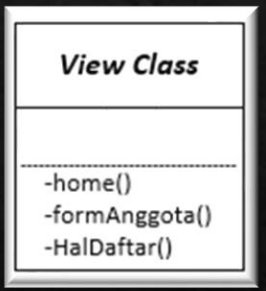
Main *Class*

Class yang akan dieksekusi pertama kali ketika sistem pertama kali berjalan,



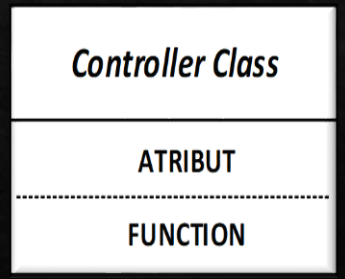
Kelas yg menangani tampilan sistem *(view)*

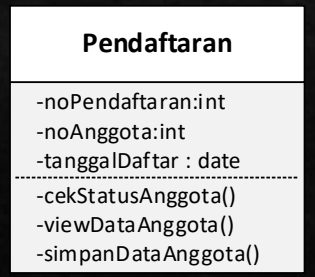
Kelas yang mendefinisikan dan mengatur tampilan ke pengguna/*user.*



Kelas yg diambil dari pendefinisian usecase *(controler)*

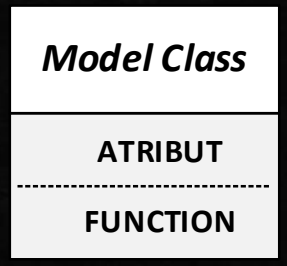
Kelas yg menangani fungsi – fungsi yg diambil dari pendefinisian *usecase,* kelas ini biasanya disebut dengan kelas proses yang menangani proses bisnis pada sistem.

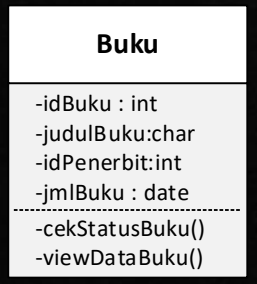


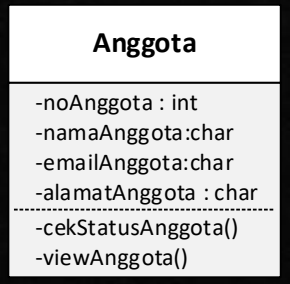


Kelas yg diambil dari pendefinisian data (model)

Kelas yg digunakan untuk memegang atau membungkus data menjadi kesatuan yang diambil maupun akan disimpan ke basis data.







Sifat Atribut dan Metoda (Visibility)

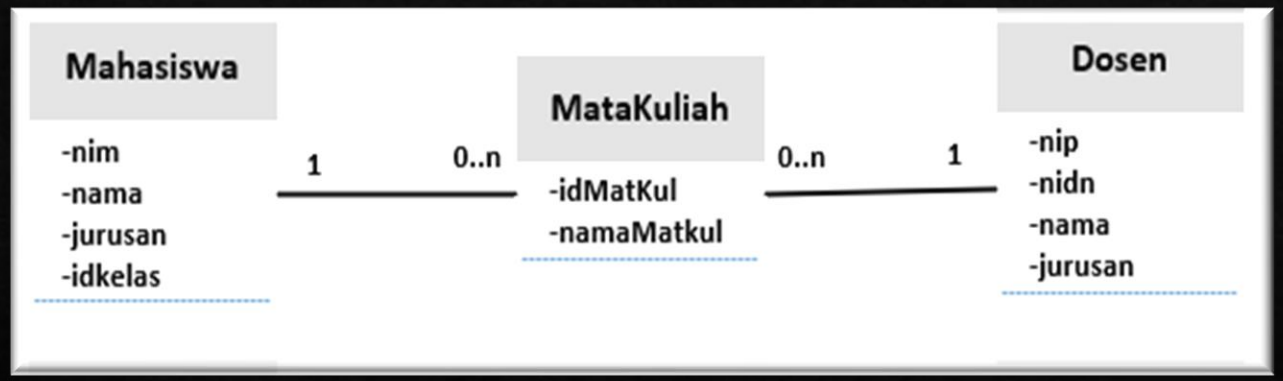
Visibility adalah kemampuan untuk menyembunyikan/ menampilkan atribut hanya pada ***claas*** tertentu.

Nama	Simbol	Deskripsi
Public	+	Bisa diakses oleh semua ***class***
Private	-	Hanya dapat diakses oleh ***class*** yang sama
Protected	#	Hanya dapat diakses oleh ***class*** yang bersangkutan dan anak-anak yang mewarisinya
Package	~	Hanya terlihat di dalam kelas yang ditentukan dan ke semua kelas dalam paket yang sama.

Relasi Asosiasi

- **Association**

  Menggambarkan hubungan antar class dengan ditandai dengan anak panah dan seringkali ditambahkan label dan multiplicity untuk memperjelas hubungan



Multiplicity	Arti
N (default)	Banyak
0..0	Nol
0..1	Nol atau satu
0..n	Nol atau banyak
1..1	Tepat satu
1..n	Satu atau banyak

Relasi Generalisasi

- **Generalisasi**

  Generalization adalah inheritance pada UML dimana sub class mewarisi feature dari super classnya. Sub class mampu overriding metode super classnya. Generalization dinotasikan dengan anak panah mengacu ke super class.

Pewarisan

Adalah mekanisme yang memungkinkan satu objek mewarisi sebagian atau seluruh definisi dan objek lain dari dirinya

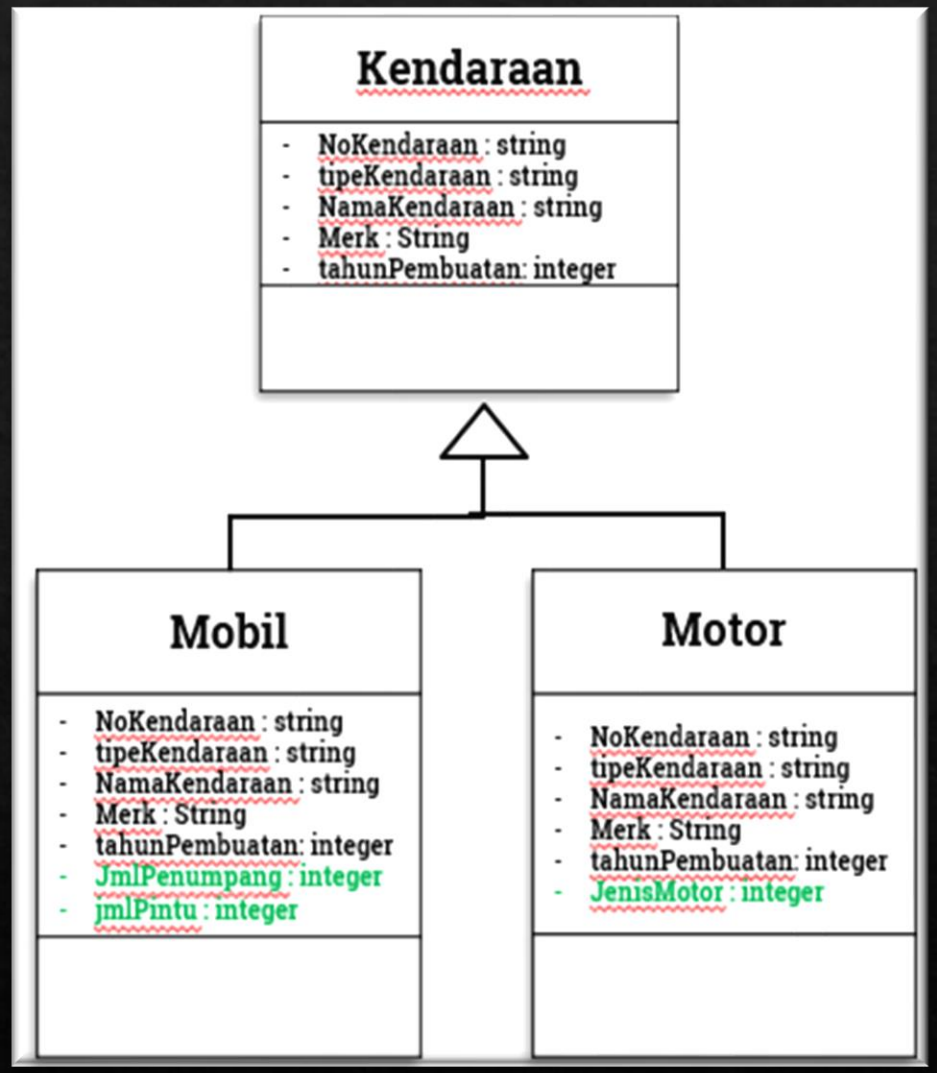
Pewarisan

Adalah konsep dimana kelas *parent* mewarisi baik atribut maupun operasi ke kelas *child*. Dengan demikikian kelas *parent* dan kelas *child* memiliki atribut dan operasi yang dama

Relasi Generalisasi

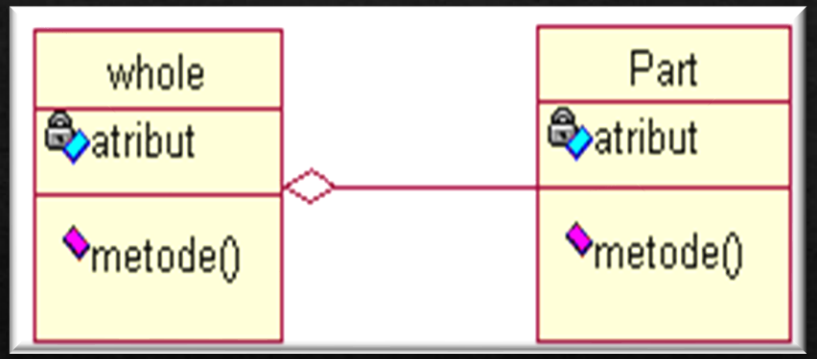
Yang perlu diperhatikan pada kelas **KENDARAAN** (***parent***) memiliki atribut yang kemudian diwariskan ke kelas **MOBIL** (***child***) dan **MOTOR** (***child***). Tapi kelas **MOBIL** dan **MOTOR** juga memiliki atribut bawaan miliki kelasnya tersendiri

Jadi pewarisan merupakan penurunan atribut dari kelas ***parent*** ke kelas ***child***, akan tetapi ***child*** memiliki atribut tersendiri yang membedakan antara ***child*** yang satu dengan ***child*** yang lain.

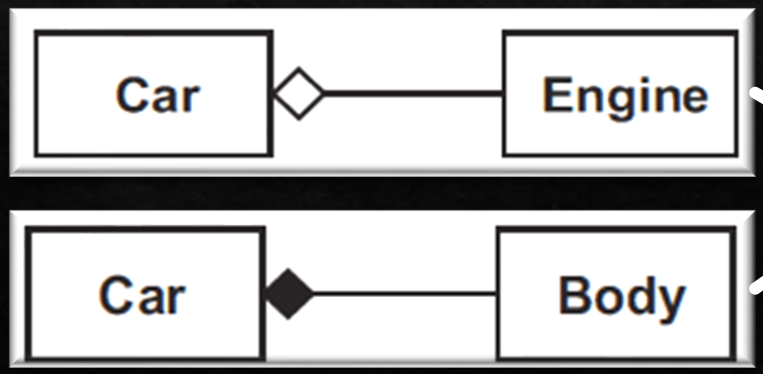


Relasi Agregasi

Sebuah ***aggregation*** adalah bentuk khusus ***association*** yang memodelkan hubungan ***whole-part*** antara sebuah ***aggregation*** dengan bagiannya.



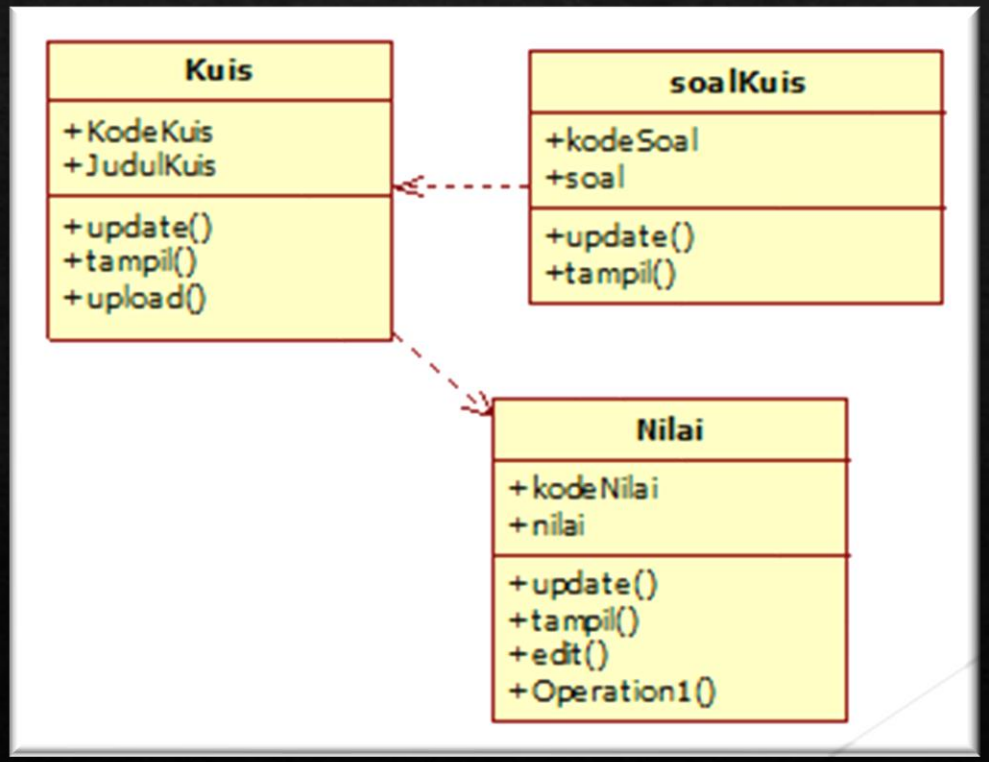
Hubungan secara keseluruhan antara aggregate class dengan component class.



- Mesin bagian dari mobil, jika tidak ada mesin mobil tidak bisa berjalan
- *Body* pun sama jika tidak ada *body* (rangka) maka mobil tidak akan lengkap.

Relasi *Dependency*

Dependency adalah perubahan pada salah satu elemen yang mengakibatkan perubahan pada elemen yang lain. Semakin kompleks sistem, maka ***dependency*** menjadi sesuatu yang harus dipertimbangkan. ***Dependency*** hanya berlaku satu arah.



OBJECT DIAGRAM

Definisi *Object Diagram*

Object Diagram berasal dari ***class diagram,*** sehingga ***object diagram*** bergantung pada ***class diagram.. Object diagram*** mewakili sebuah ***instance class***

Object diagram adalah gambaran objek – objek secara ringkas di sebuah sistem pada suatu waktu. ***Object diagram,*** sering disebut sebagai ***instance diagram*** karena menunjukan ***instance - instance*** dari ***class.***

Definisi *Object Diagram*

Diagram objek digunakan untuk menunjukkan keberadaan objek dan hubungannya dalam desain logis suatu sistem. Dengan kata lain, **diagram objek** merepresentasikan snapshot dalam waktu aliran transitoris kejadian sebaliknya atas konfigurasi objek tertentu.

Diagram objek menyajikan contoh konkret yang membantu dalam visualisasi diagram kelas terkait.

Object diagram digunakan untuk membuat satu set benda dan hubungan mereka sebagai contoh.

Tujuan *Object Diagram*

Tujuan dari *Object Diagram* mirip dengan ***class diagram***. Perbedaaanya, ***class diagram*** mewakili model abstrak yang terdiri dari ***class*** dan relasinya. Namun ***object diagram*** merupakan contoh pada saat tertentu yang bersifat konkrit. Secara singkat, tujuan ***object diagram*** dapat digambarkan sebagai berikut :

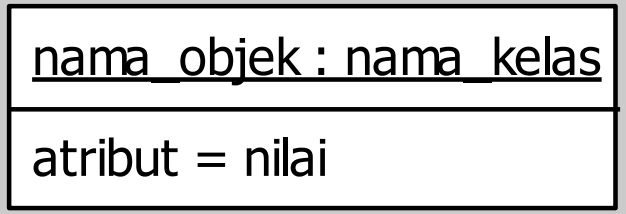
- Untuk *forward* dan *reverse engineering*
- Menunjukan relasi objek dari suatu sistem.
- Menunjukan pandangan statis dari suatu interaksi
- Memahami perilaku objek dan hubungan mereka dari persepektif praktis.

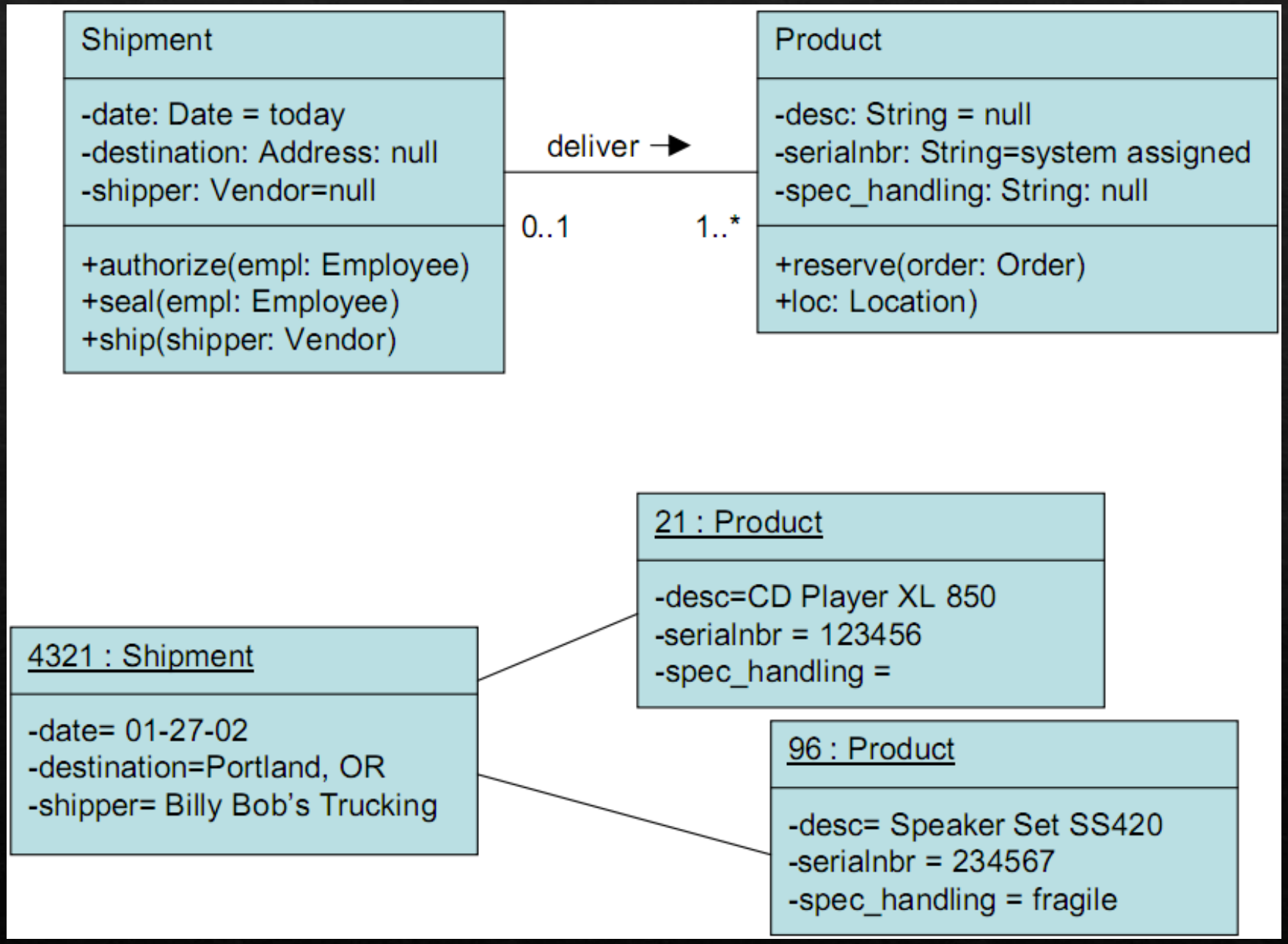
Pemodelan *Object Diagram*

Untuk memodelkan sebuah struktur objek bisa dilakuakn Langkah – Langkah sebagai berikut :

- Identifikasi mekanisme yang akan dimodelkan. Sebauah mekanisme mewakili beberapa fungsi atau perilaku dari Sebagian sistem yang akan dimodelkan sebagai hasil dari interaksi *class, interface,* dan hal – hal lain.
- Untuk setiap mekanisme, identifikasi *class-class*, *interface*, dan elemen – elemen yang lainnya yang berpartisipasi pada kolaborasi ini. Selanjutnay perlu diidentifikasikan relasi diantara semua elemn tersebut.
- Pertimbangkan satu scenario yang meliputi semua mekanismetersebut. Pertahankan scenario tersebut sementar waktu dan ulangi untuk setiap objek yang berpartisipasi pada mekanisme tersebut
- Ekspose nilai state dan atribut dari setiap objek untuk memahami scenario tersebt.
- Dengann cara yang sama, ekspose link diantara objel – objek yang mewakili instance dari asosiasi diantara mereka.

Relasi *Object Diagram*

Deskripsi	Notasi
Objek dari kelas yang berjalan saat sistem dijalankan	
relasi antar objek	


Shipment
-date: Date = today
-destination: Address: null
-shipper: Vendor=null
+authorize(empl: Employee)
+seal(empl: Employee)
+ship(shipper: Vendor)
deliver
0..1
1..*
Product
-desc: String = null
-serialnbr: String=system assigned
-spec_handling: String: null
+reserve(order: Order)
+loc: Location)
21 : Product
-desc=CD Player XL 850
-serialnbr = 123456
-spec_handling =
4321 : Shipment
-date= 01-27-02
-destination=Portland, OR
-shipper= Billy Bob's Trucking
96 : Product
-desc= Speaker Set SS420
-serialnbr = 234567
-spec_handling = fragile

Referensi

[1] https://www.uml-diagrams.org/class-diagrams-overview.html

[2] https://www.uml-diagrams.org/class-diagrams-overview.html#object-diagram

[3] MUNAWAR, Analisis Perancangan Sistem Berorientasi Objek dengan UML, 2018, Penerbit Informatika, Bandung.

[4] Rosa A. S, M. Shalahuddin, Rekayasa Perangkat Lunak Terstruktur dan Objek, 2014, Penerbit Informatika, Bandung.