

Unified Modeling Language (UML)



Presentation

...

With
Group 6

Our Team

LM. YUDHY PRAYITNO
E1E122064



DIKHSAN DWIRANAGGA TIBONG
E1E122092



DELA PUSPITA HELMI
E1E122007



GAMALIEL GUSRAYANTO
E1E122054



ANNISA AULIYA RAMADHANI
E1E122003



Apa itu UML?

UML (Unified Modelling Language) adalah suatu metode pemodelan secara visual yang berfungsi sebagai sarana perancangan sistem berorientasi objek (OOAD/Object Oriented Analysis and Design).

UNIFIED
MODELING
LANGUAGE™



Definisi UML adalah sebagai suatu bahasa yang sudah menjadi standar pada visualisasi, perancangan, dan juga pendokumentasian sistem aplikasi. Saat ini UML menjadi bahasa standar dalam penulisan blue print software (arsitektur).

Problem

Pentingnya Memahami UML

UML diharapkan mampu mempermudah pengembangan perangkat lunak (RPL) serta memenuhi semua kebutuhan pengguna dengan efektif, lengkap, dan tepat. Hal itu termasuk faktor-faktor scalability, robustness, security, dan sebagainya.

01

Enhancing Communication and
Understanding Among Teams

02

Can be used as a blueprint for
software systems.

03

Simplifying System Maintenance and
Evolution

Solution

UML VS DFD

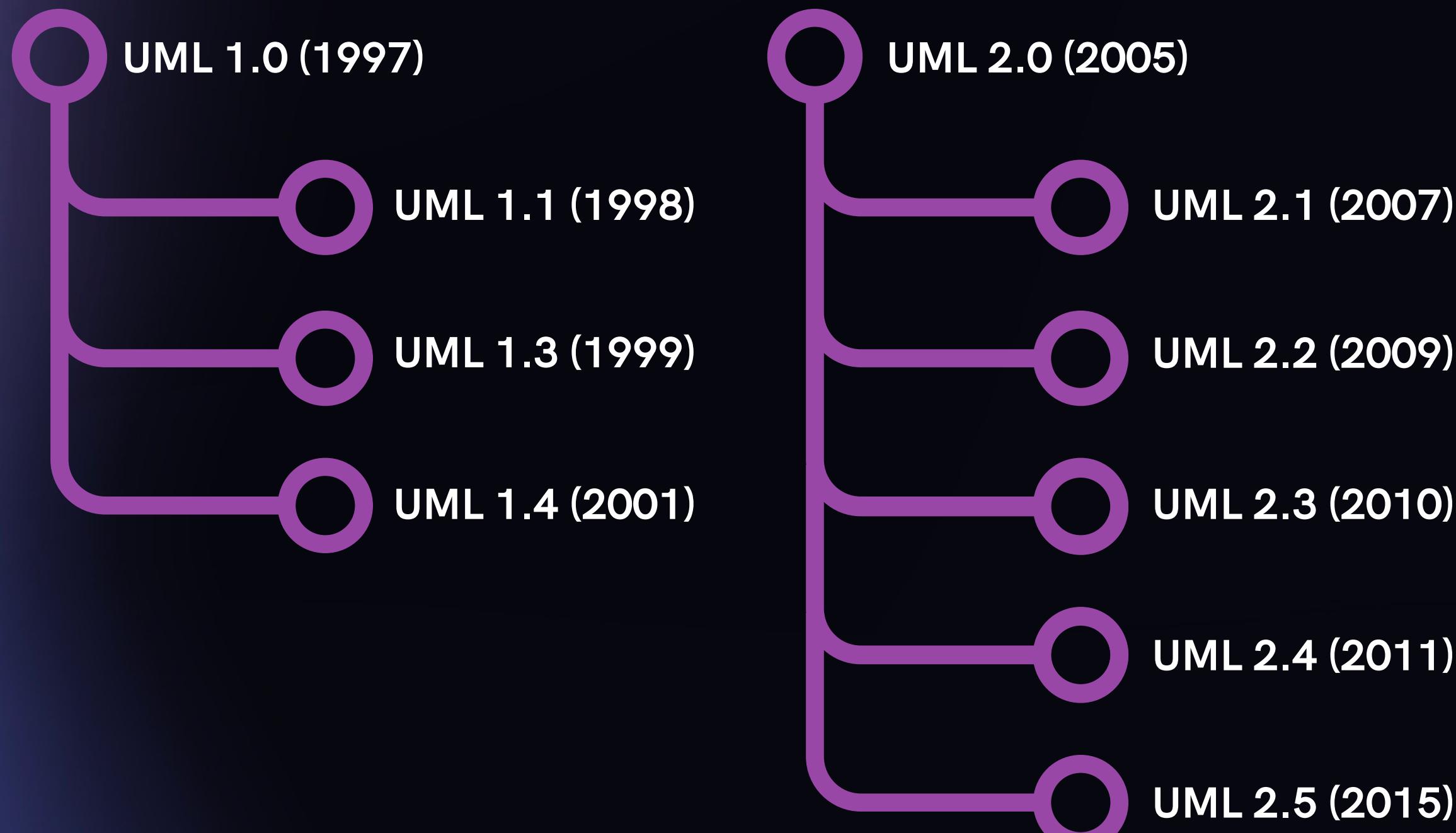
Data Flow Diagram (DFD) dan Unified Modeling Language (UML) adalah dua bahasa pemodelan yang digunakan untuk menggambarkan sistem informasi. DFD lebih berfokus pada menggambarkan aliran data dalam sistem, sedangkan UML lebih berfokus pada menggambarkan struktur dan perilaku sistem.

What is the difference between UML and DFD?

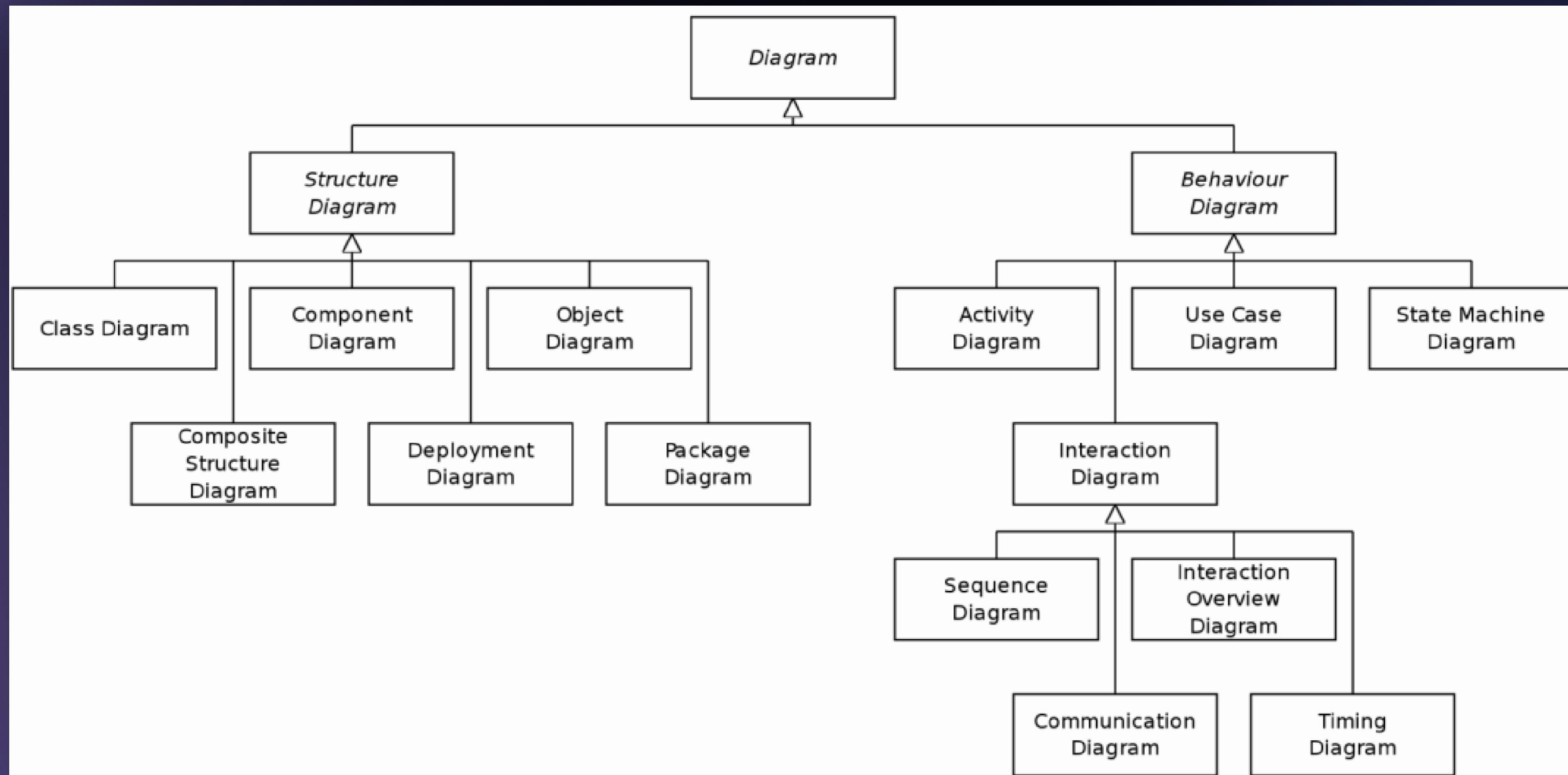
Aspek	UML	DFD
Orientasi	Berorientasi objek	Berorientasi data
Kompleksitas	Struktur dan perilaku sistem	Aliran data dalam sistem
Fokus	Lebih kompleks	Lebih sederhana
Jenis Diagram	13	3

UML Versions

UML mulai diperkenalkan oleh Object Management Group (OMG), UML dikembangkan sebagai suatu alat untuk analisis dan desain berorientasi objek oleh Grady Booch, Jim Rumbaugh, dan Ivar Jacobson.



Types of UML Diagrams?



Structur Diagram

...

Untuk menjelaskan suatu struktur statis dari sistem yang dimodelkan.

1

Class Diagram : to model the class structure

Class Diagram penggambaran dari class, atribut, dan objek disamping itu juga hubungan satu sama lain seperti pewarisan, containmet, asosiasi dan lainnya.

Class Diagram disebut jenis diagram struktur (structure diagram) karena menggambarkan apa yang harus ada dalam sistem yang dimodelkan dengan berbagai komponen.

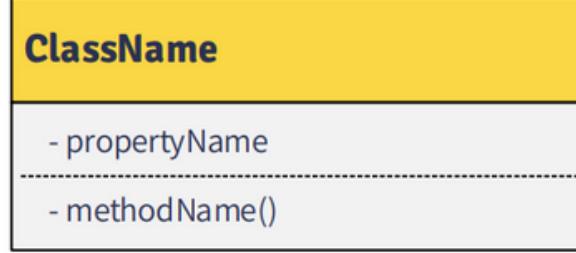
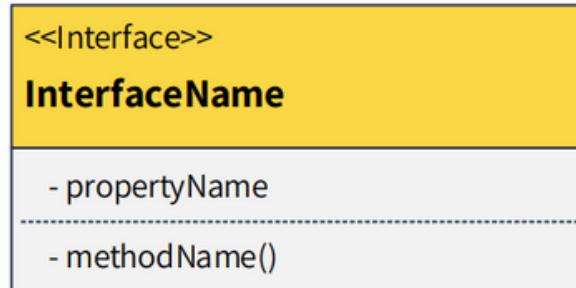
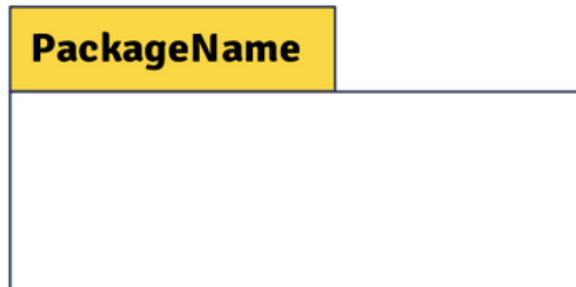
Kegunaan, menjelaskan suatu model data untuk program informasi, tidak peduli apakah model data tersebut sederhana maupun kompleks.

Structur Diagram

...

1

Class Diagram : to model the class structure

Simbol Komponen	Nama	Fungsi
	Class	Merepresentasikan stuktur Class, termasuk nama class, property dan methodnya.
	Interface	Merepresentasikan Class Abstract (Interface), termasuk nama interface, property dan methodnya.
	Package (Expanded)	Sebuah bungkusan yang berisi satu atau lebih class.

Structur Diagram

...

1

Class Diagram : to model the class structure

Simbol Hubungan	Nama	Fungsi
—	Association	Relasi antar class dengan arti umum, asosiasi biasanya juga disertai dengan Kardinalitas.
↔	Direct Association	Relasi antar class dengan makna class yang satu digunakan oleh class yang lain, asosiasi biasanya juga disertai dengan Kardinalitas.
→	Inheritance	Relasi antar subclass ke superclass, dengan superclass bukan class abstrak (interface).
— — — >	Interface Realization	Relasi antar subclass ke superclass, dengan superclass adalah class abstrak (interface).
— — — — >	Dependency	Relasi antar class dengan makna kebergantungan antar class.
— ◊	Aggregation	Relasi antar class dengan makna semua bagian (whole-part).
— →	Composition	Relasi antar class dengan makna merupakan bagian antar class.

Simbol Kardinalitas	Dibaca
N (*)	Banyak
0..0	Nol
0..1	Nol atau satu
0..N	Nol atau Banyak
1..1	Satu
1..N	Satu atau Banyak

Hubungan antara objek atau instance dari satu kelas dengan objek atau instance dari kelas lain dalam suatu sistem

Structur Diagram

1

Class Diagram : to model the class structure

Mobil

```
- merk : string
# banyakBan : integer
+ autoDriving : boolean
...
+ constructor(merk, banyakBan,
  autoDriving, ...): void
- setMerk( merk ) : void
# getMerk() : string
...
```

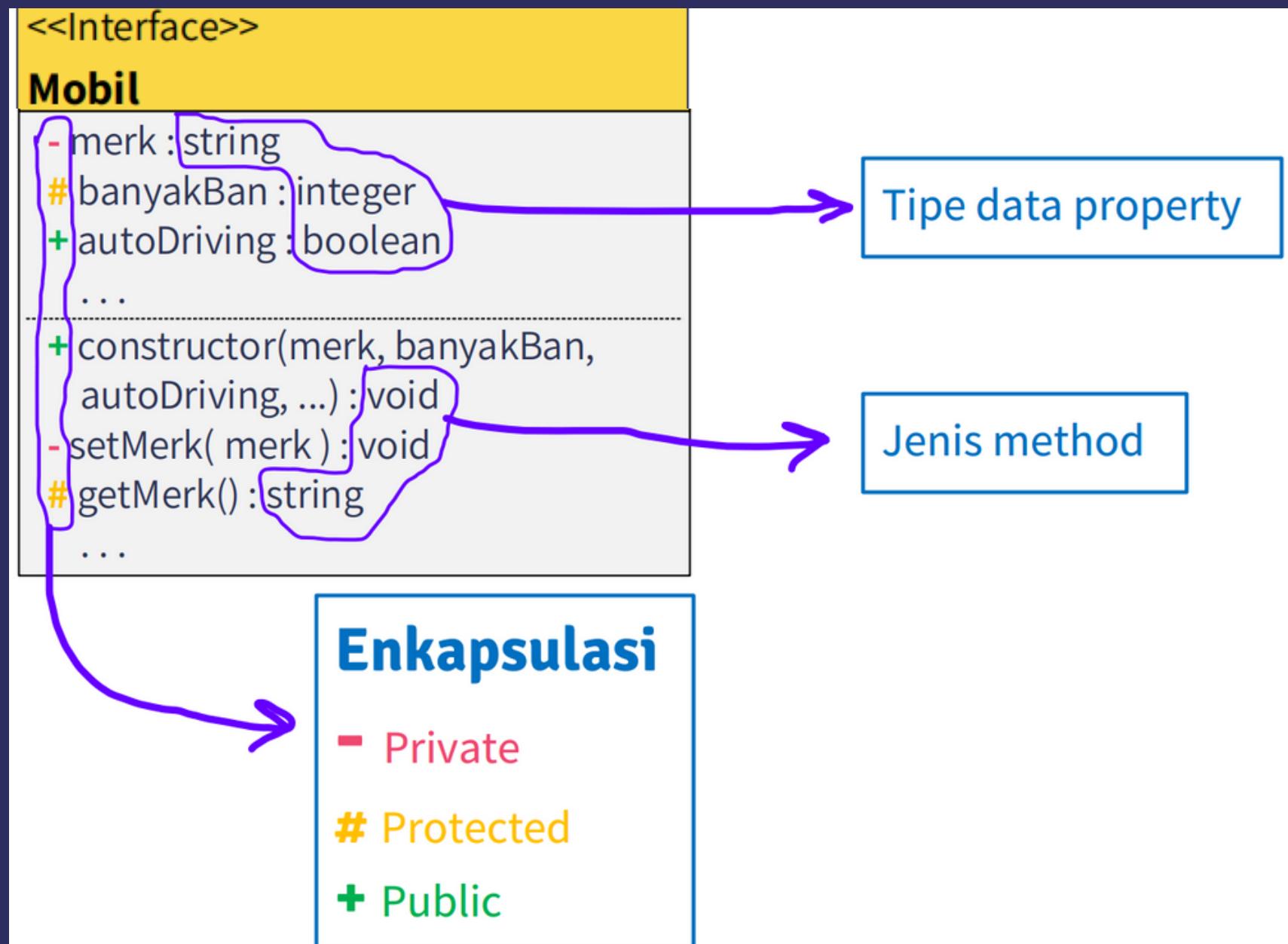
Merepresentasikan struktur Class, termasuk nama class, property dan methodnya.

...

Structur Diagram

1

Class Diagram : to model the class structure



Menulis Method Abstract

`<<abstract>> writeMethod`

Contoh :

`<<abstract>> + jenisMobil() : void`

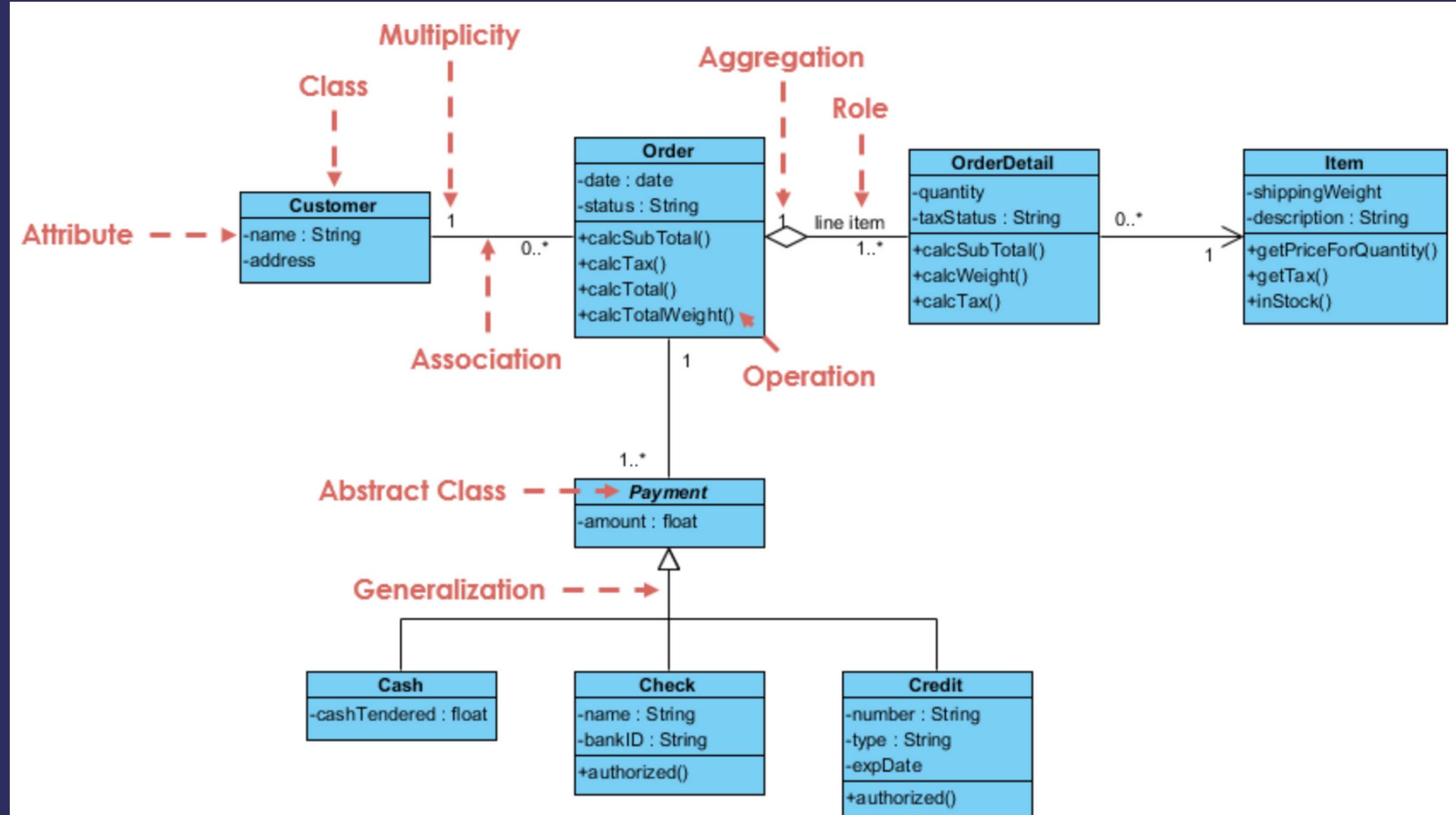
Merepresentasikan Class Abstract (Interface), termasuk nama interface, property dan methodnya.

...

Structur Diagram

1

Class Diagram : to model the class structure



...

Structur Diagram

2

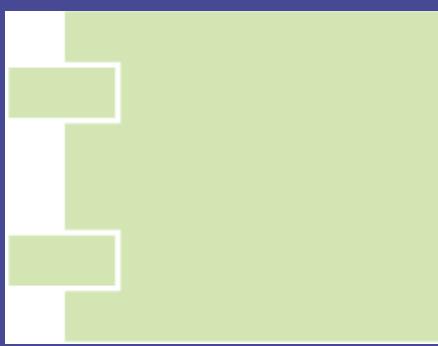
Component Diagram : to model the object component

- Component Diagram menggambarkan struktur antar komponen piranti perangkat lunak
- Diagram ini biasanya digunakan saat bekerja dengan sistem yang kompleks yang memiliki banyak komponen. Komponen-komponen berkomunikasi satu sama lain menggunakan antarmuka. Antarmuka-antarmuka ini dihubungkan menggunakan penghubung.
- Komponent piranti perangkat lunak berisi code yang meliputi; Source code, binary code, library maupun executable

...

Structur Diagram

Component symbol



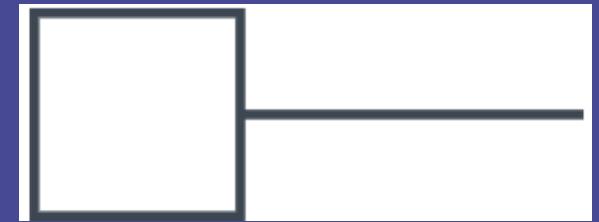
Merupakan entitas yang diperlukan untuk menjalankan fungsi stereotipe.

Node symbol



Mewakili objek perangkat keras atau perangkat lunak yang berada pada level yang lebih tinggi daripada komponen.

Port symbol



Menyatakan titik interaksi terpisah antara the component and the environment.

Note symbol



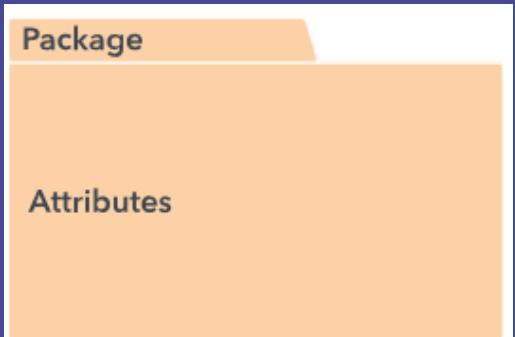
Memungkinkan pengembang menambahkan meta-analisis ke diagram komponen.

Dependency symbol



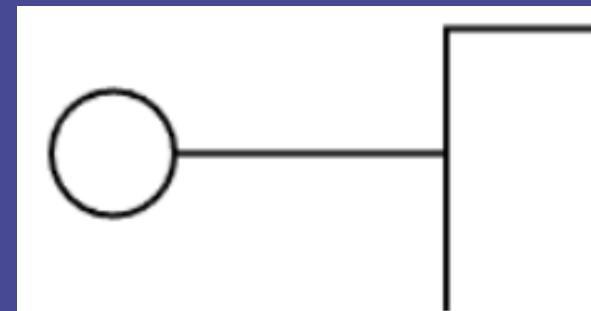
Menunjukkan bahwa satu bagian dari sistem Anda bergantung pada yang lain.

Package symbol



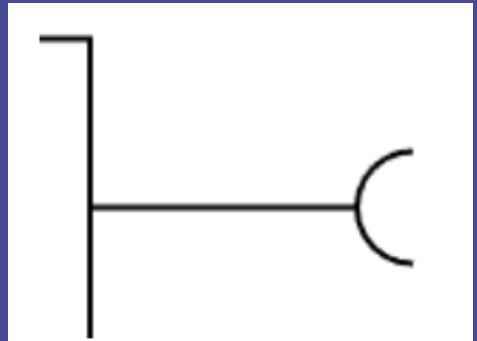
Mengelompokkan beberapa elemen sistem dan direpresentasikan oleh folder file

Provided interfaces



Simbol ini mewakili antarmuka di mana sebuah komponen menghasilkan informasi yang digunakan oleh antarmuka yang diperlukan oleh komponen lain.

Required interfaces



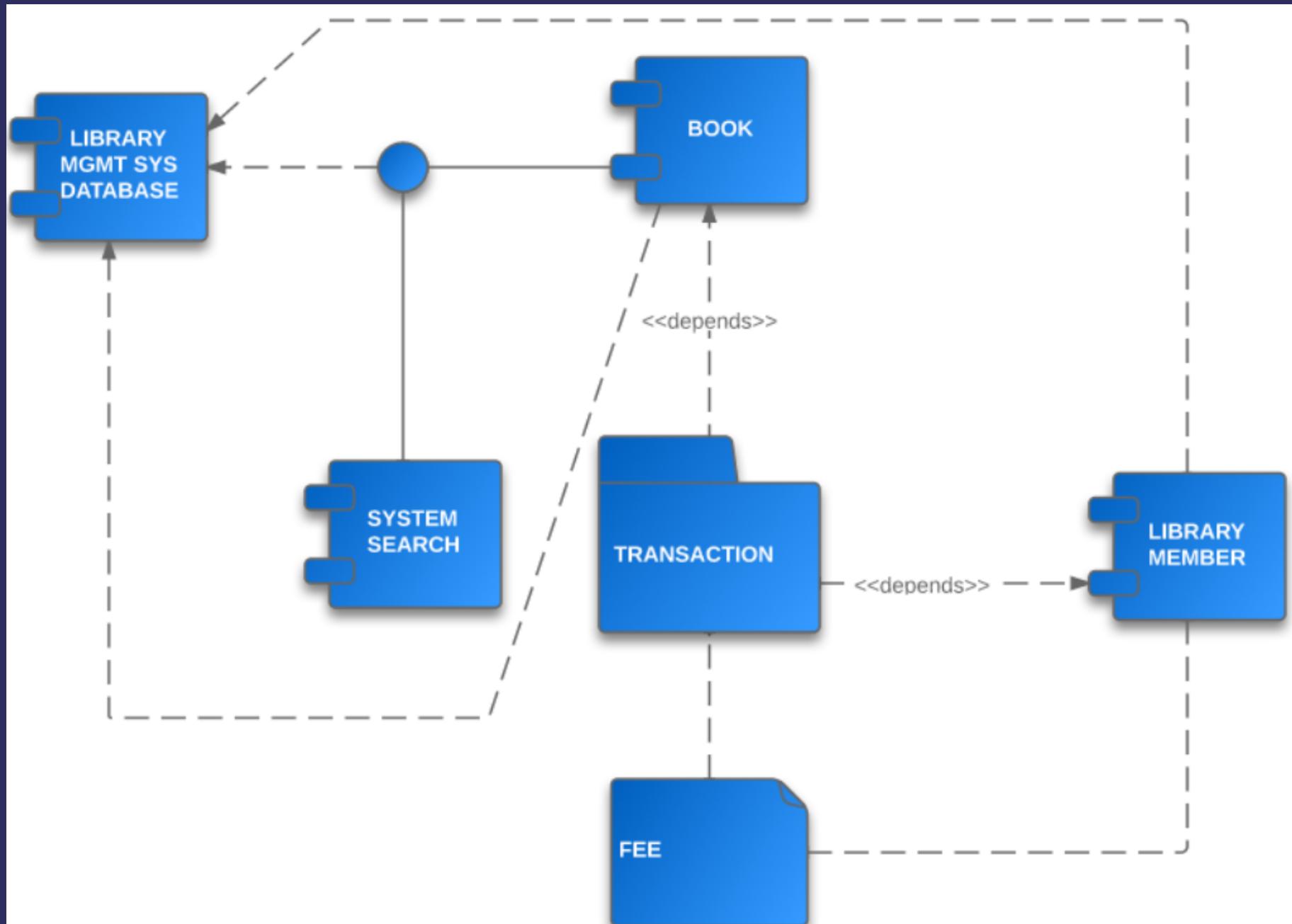
Simbol ini mewakili antarmuka di mana komponen memerlukan informasi agar dapat menjalankan fungsinya dengan baik.

Structur Diagram

...

2

Component Diagram : to
model the object component



Structur Diagram

3

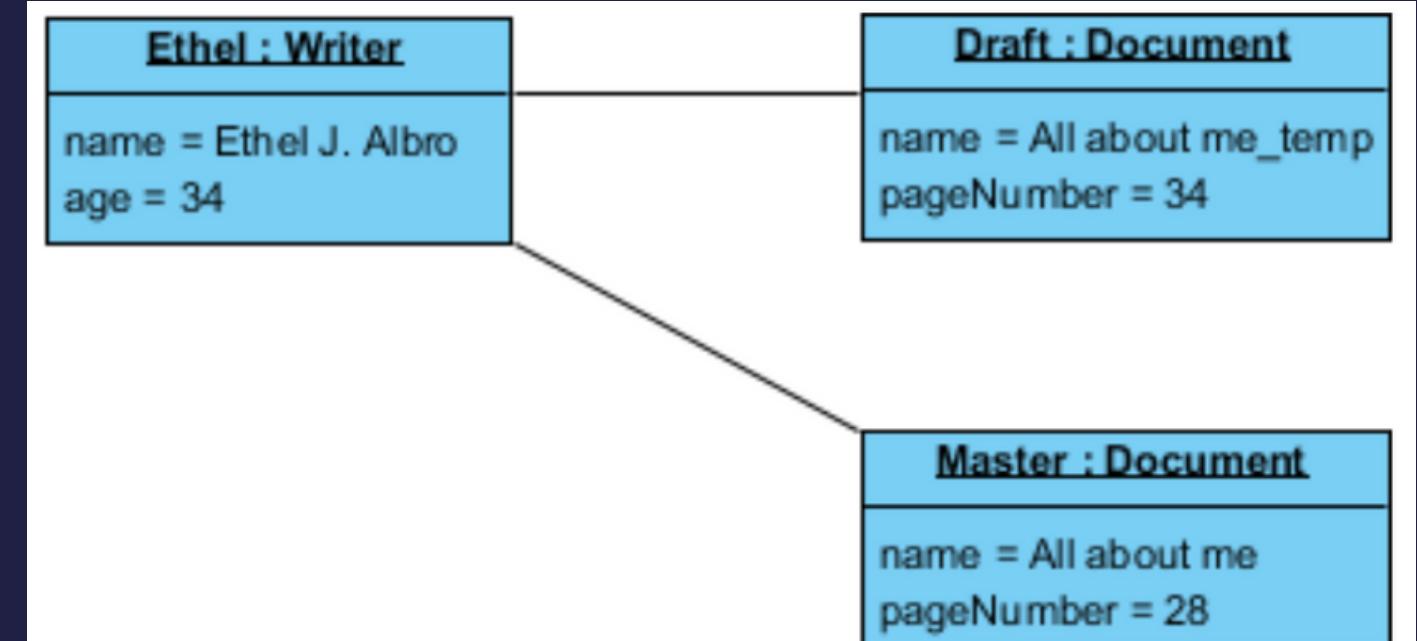
Object Diagram : to model the object structure

Objek adalah Perwujudan dari sebuah kelas

Class



Object

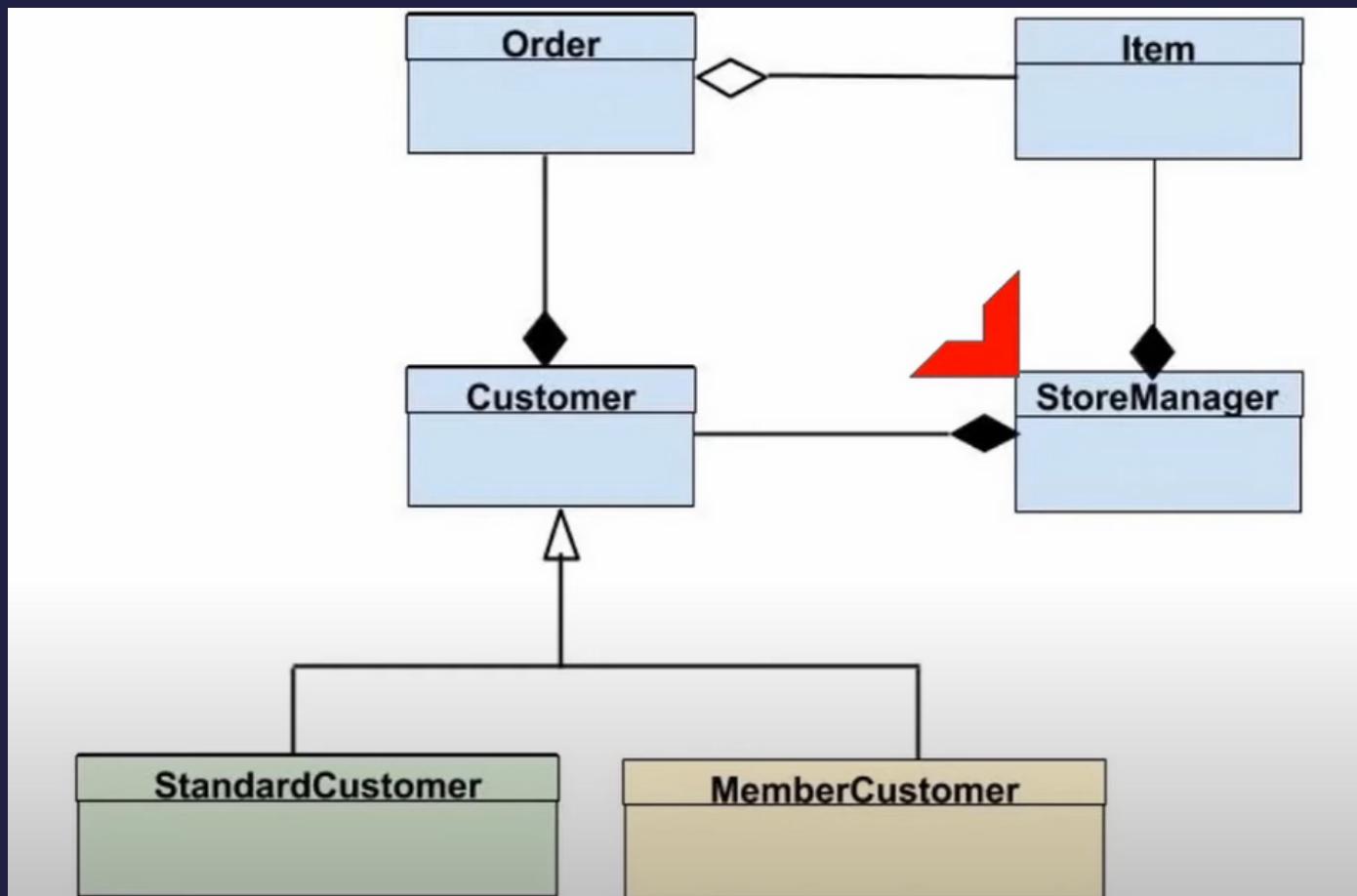


Structur Diagram

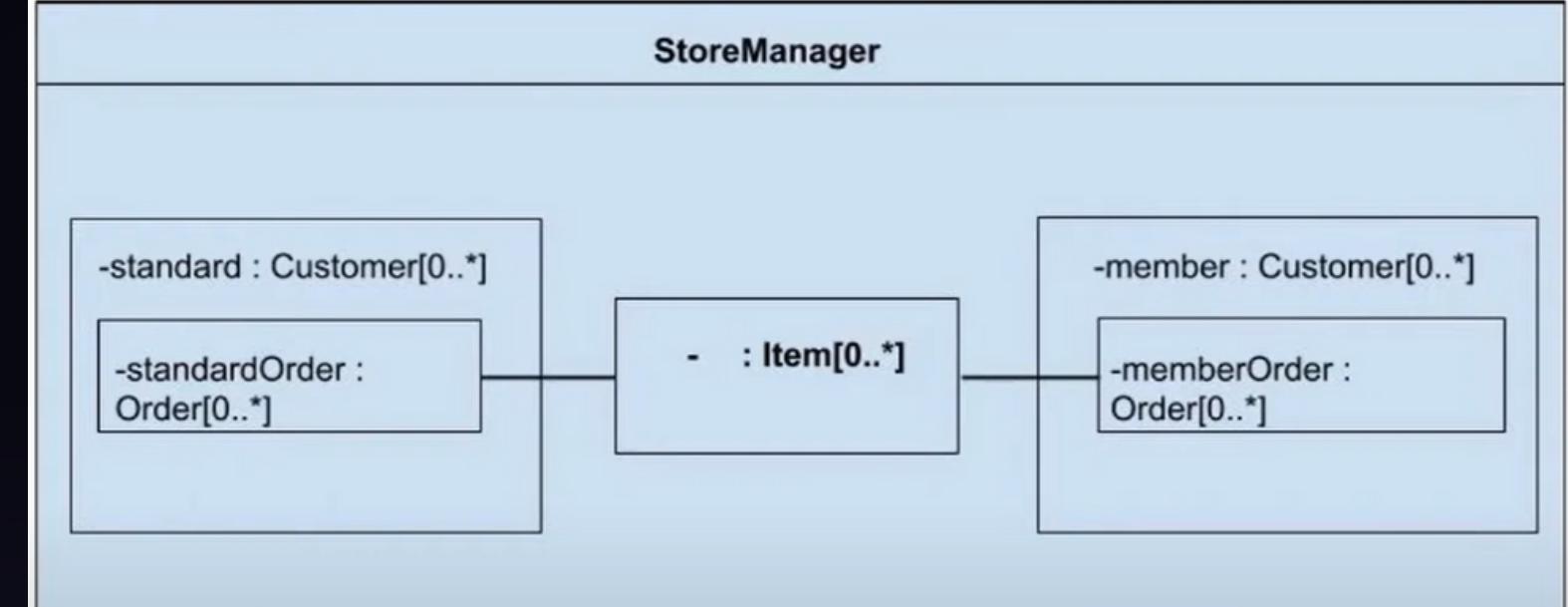
4

Composite Structure Diagram

Class



Composite



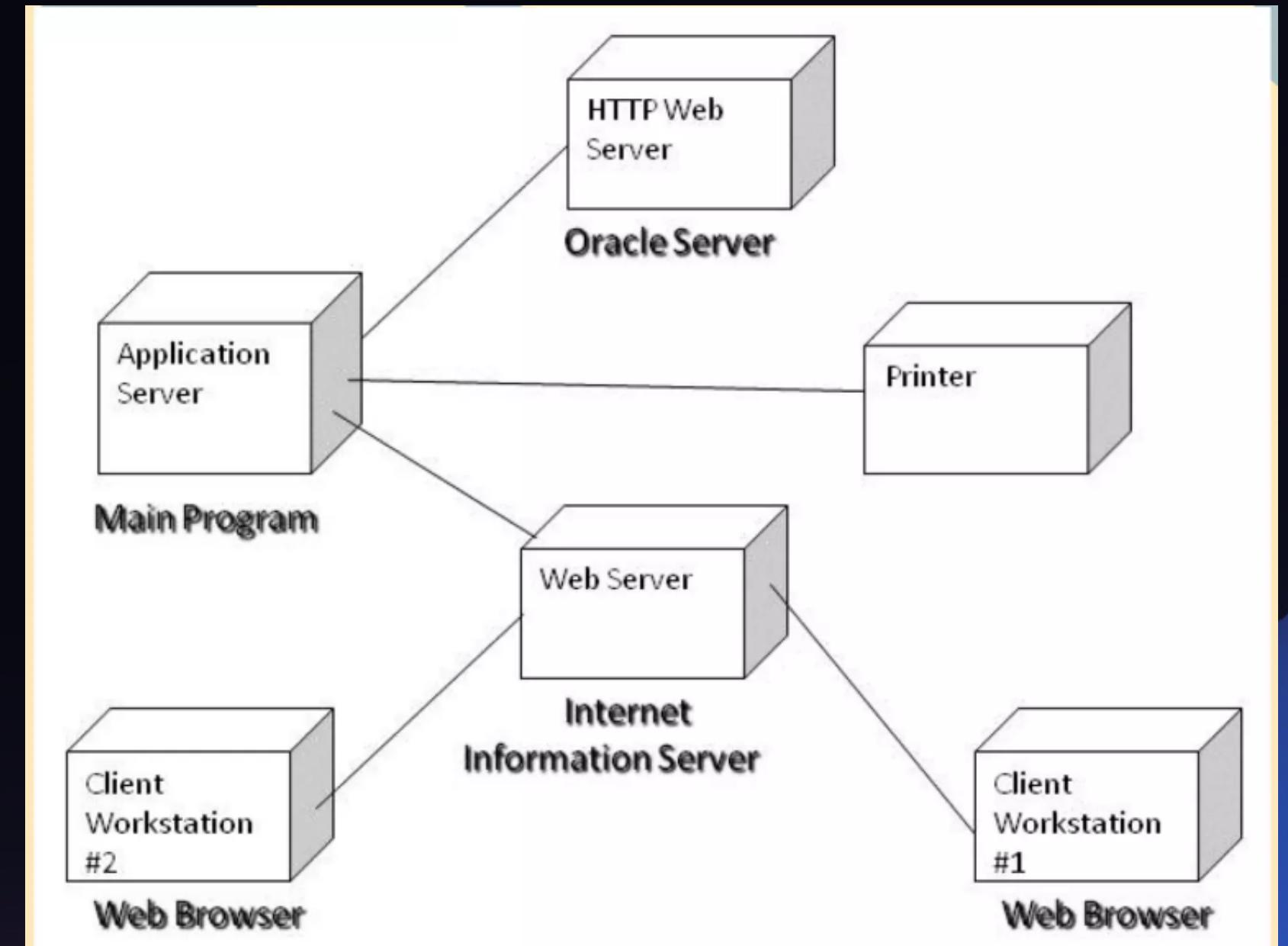
untuk memodelkan struktur internal dari (component, class, dan use case), termasuk hubungan pengklasifikasian ke bagian lain dari sebuah program

Structur Diagram

5

Deployment Diagram: to modeling the distribution of application

Deployment diagram menunjukkan tata letak sebuah sistem secara fisik, menampakkan bagian-bagian software yang berjalan pada bagian-bagian hardware yang digunakan untuk mengimplementasikan sebuah sistem dan keterhubungan antara komponen-komponen hardware tersebut

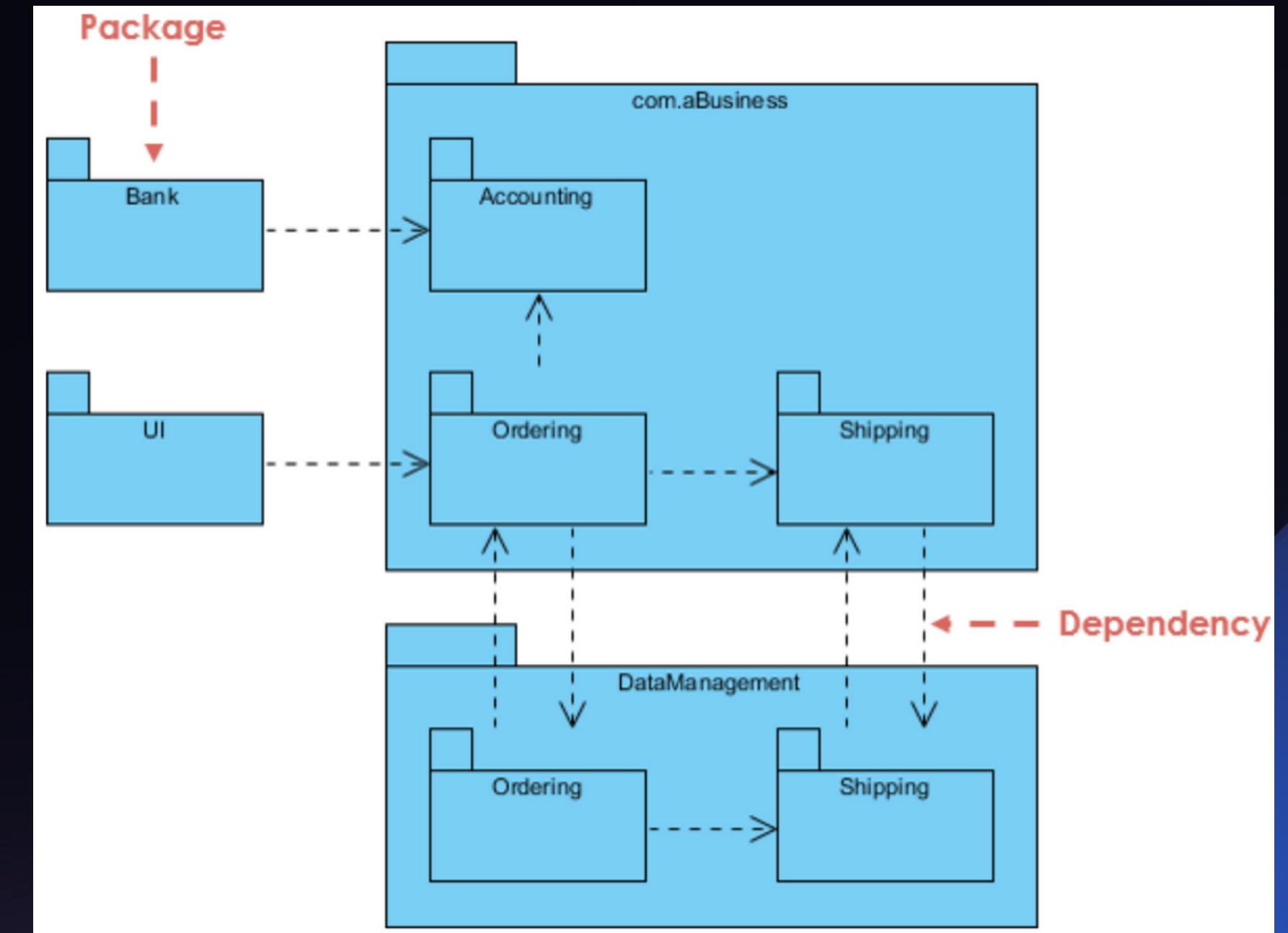


Structur Diagram

6

Package Diagram

Diagram paket adalah diagram yang menggambarkan struktur dan ketergantungan antara paket-paket yang membentuk model. Paket adalah kumpulan elemen model yang saling terkait, seperti kelas, objek, use case, dan komponen.



Behavior Diagram

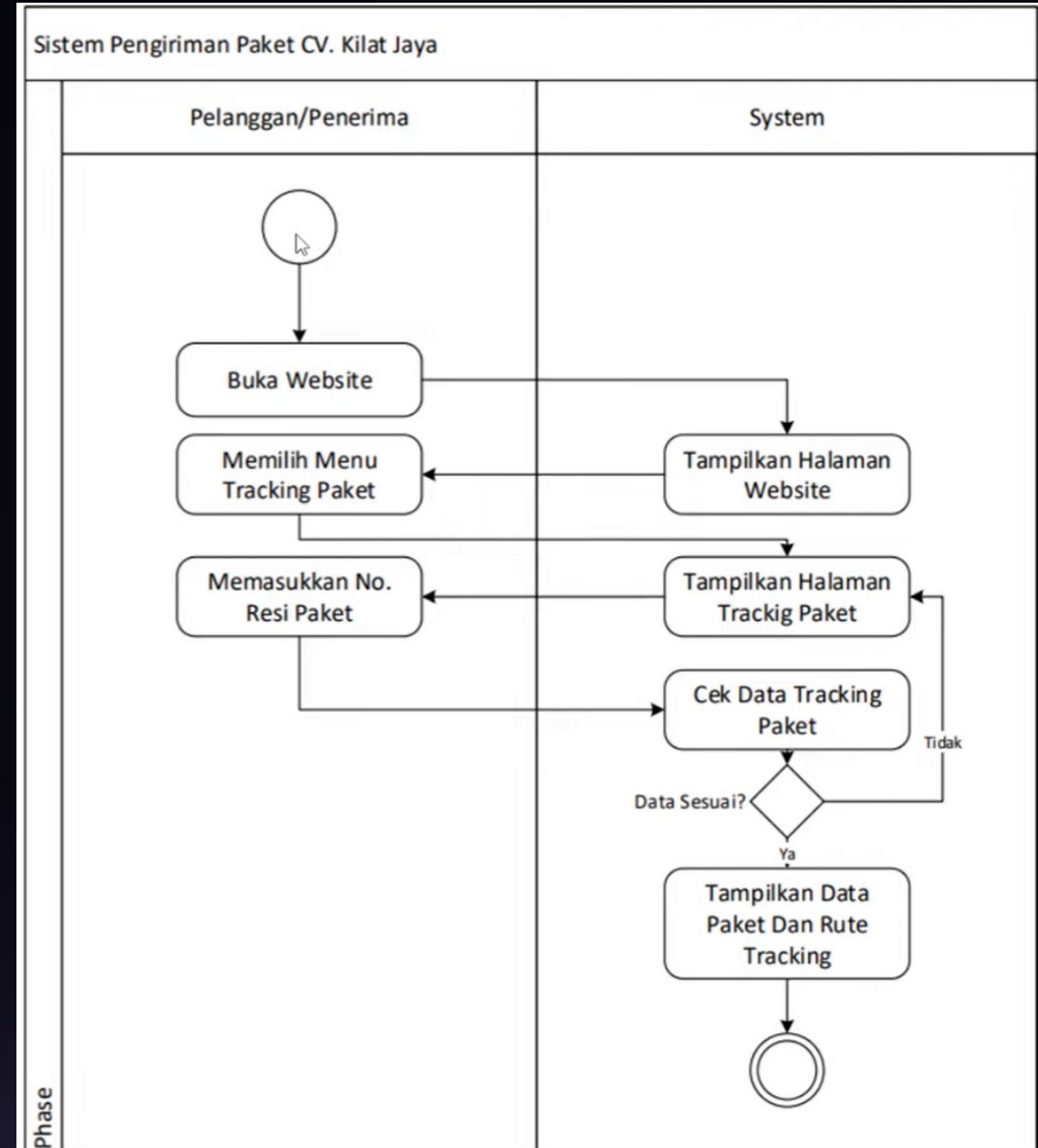
untuk menjelaskan kelakuan sistem atau rangkaian perubahan yang terjadi pada sebuah sistem.

1

Activity diagram

Activity diagram adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aluran kerja dalam banyak kasus.

Activity diagram mempunyai peran seperti halnya flowchart, akan tetapi perbedaannya dengan flowchart adalah activity diagram bisa mendukung perilaku paralel sedangkan flowchart tidak bisa.



Behavior Diagram

2

Use Case Diagram

Use case diagram adalah gambar dari beberapa atau seluruh aktor dari use case dengan tujuan mengenali interaksi mereka dalam sebuah sistem.

Gambaran fungsionalitas yang harapkan dari sebuah sistem menekankan pada "APA" bukan "BAGAIMANA" yang dibuat sistem.

Menyatakan suatu job/pekerjaan tertentu, misal; login ke sistem, dll

NO	GAMBAR	NAMA	KETERANGAN
1		Actor	Menspesifikasi kan himpuan peran yang pengguna mainkan ketika berinteraksi dengan use case.
2		Include	Menspesifikasi bahwa use case sumber secara eksplisit.
3		Extend	Menspesifikasi bahwa use case target memperluas perilaku dari use case sumber pada suatu titik yang diberikan.
4		Association	Apa yang menghubungkan antara objek satu dengan objek lainnya.
5		System	Menspesifikasi paket yang menampilkan sistem secara terbatas.
6		Use Case	Deskripsi dari urutan aksi-aksi yang ditampilkan sistem yang menghasilkan suatu hasil yang terukur bagi suatu aktor

Behavior Diagram

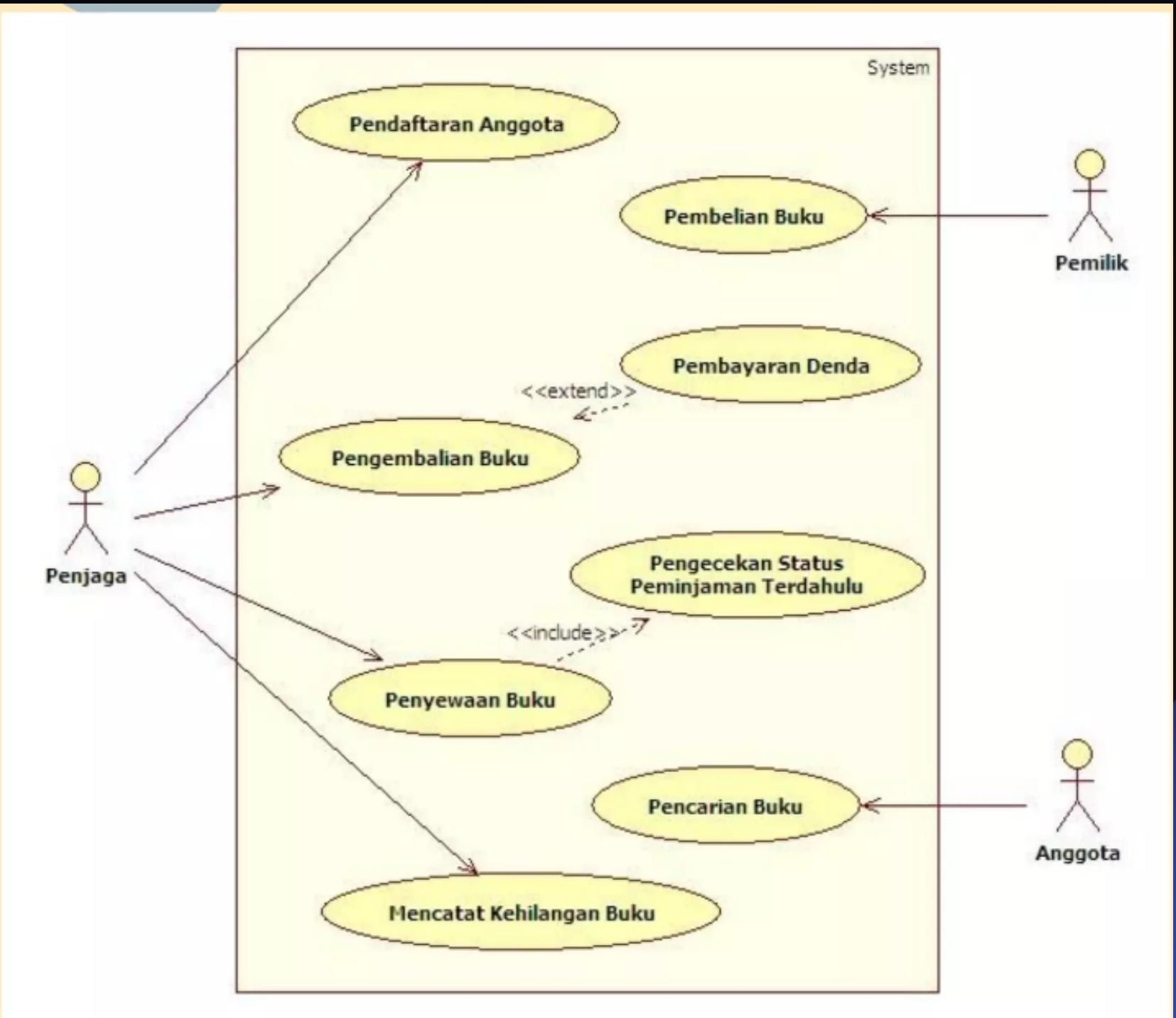
2

Use Case Diagram

Use case diagram adalah gambar dari beberapa atau seluruh aktor dari use case dengan tujuan mengenali interaksi mereka dalam sebuah sistem.

Gambaran fungsionalitas yang harapkan dari sebuah sistem menekankan pada "APA" bukan "BAGAIMANA" yang dibuat sistem.

Menyatakan suatu job/pekerjaan tertentu, misal; login ke sistem, dll



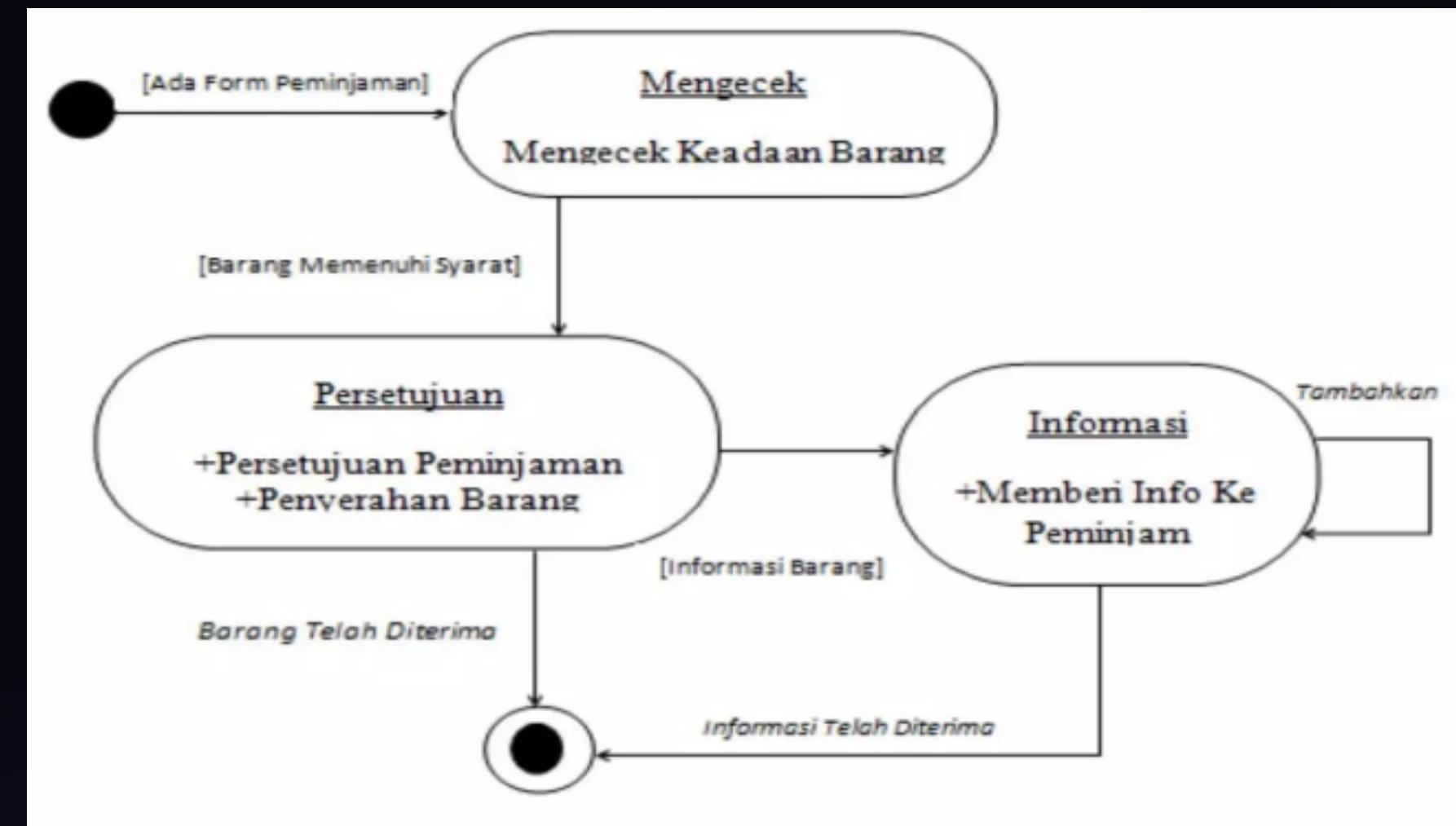
Behavior Diagram

3

State Diagram

State Diagram UML adalah diagram yang menggambarkan perilaku sistem dari waktu ke waktu. State diagram menggunakan simbol-simbol untuk mewakili keadaan, transisi, dan aksi.

- Keadaan adalah kondisi sistem pada waktu tertentu.
- Transisi adalah perubahan dari satu keadaan ke keadaan lain.
- Aksi adalah tindakan yang dilakukan oleh sistem saat terjadi transisi.



Interaction Diagram

untuk menjelaskan interaksi sistem dengan sistem lain maupun antar sistem pada sebuah sistem.

1

Sequence Diagram

Sequence diagram menjelaskan interaksi objek yang disusun berdasarkan urutan waktu. Secara mudahnya sequence diagram adalah gambaran tahap demi tahap yang seharusnya dilakukan untuk melakukan sesuatu yang sesuai dengan use case diagram.

Diagram 2 dimensi, imensi vertikal menunjukan waktu dan dimensi horizontal menunjukan objek-objek.

Notasi Sequence Diagram

1. Objek atau Aktor



Digambarkan dengan persegi panjang dan terdapat garis vertical putus-putus (*lifeline*). Semua aktivitas dilakukan di dalam *lifeline*. Aktivitas ini disebut pesan.

2. Pertukaran Pesan

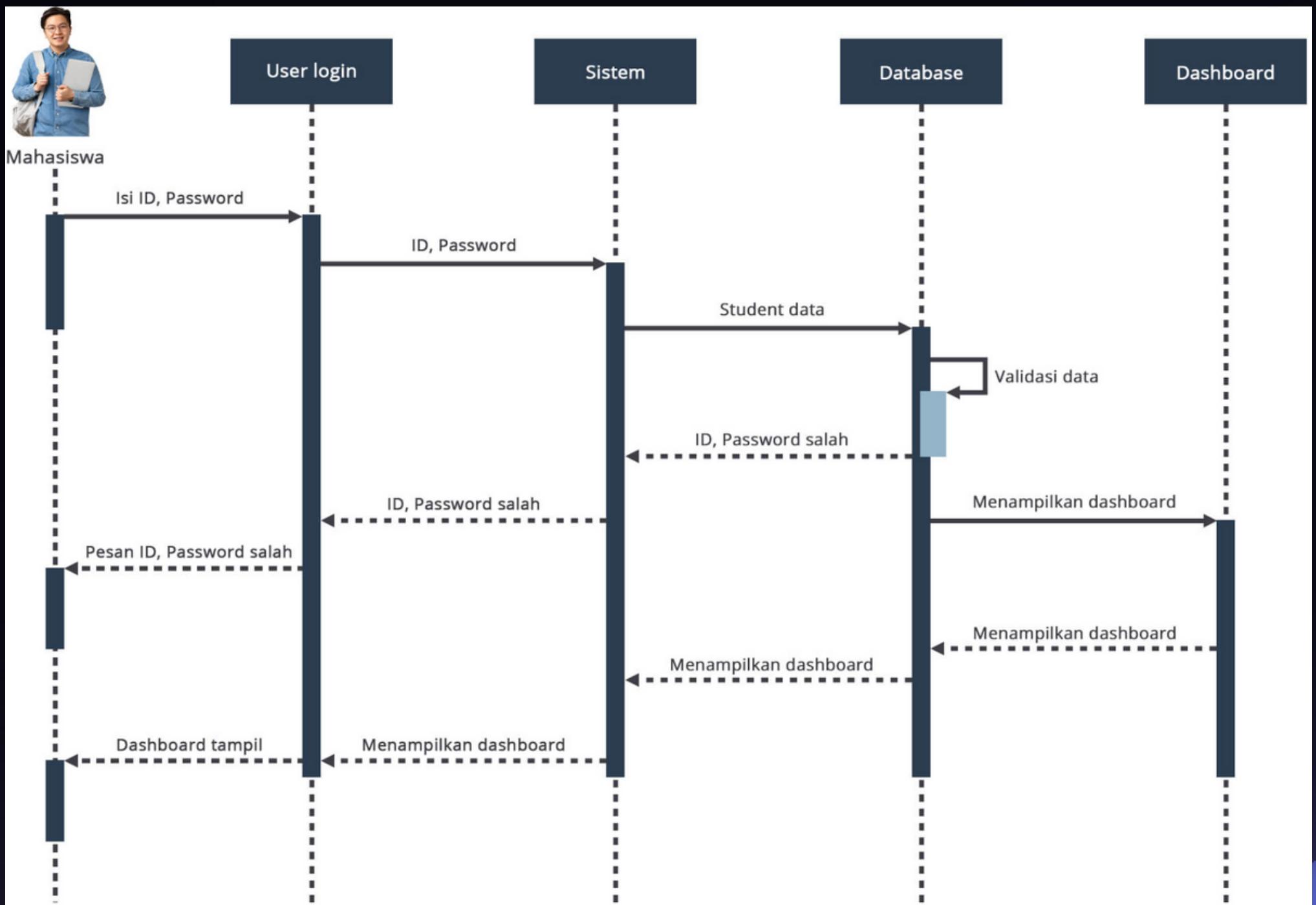


Pesan yang dipertukarkan antar objek digambarkan dengan anak panah kemudian di atasnya diberikan label pesan.

Interaction Diagram

1

Sequence Diagram



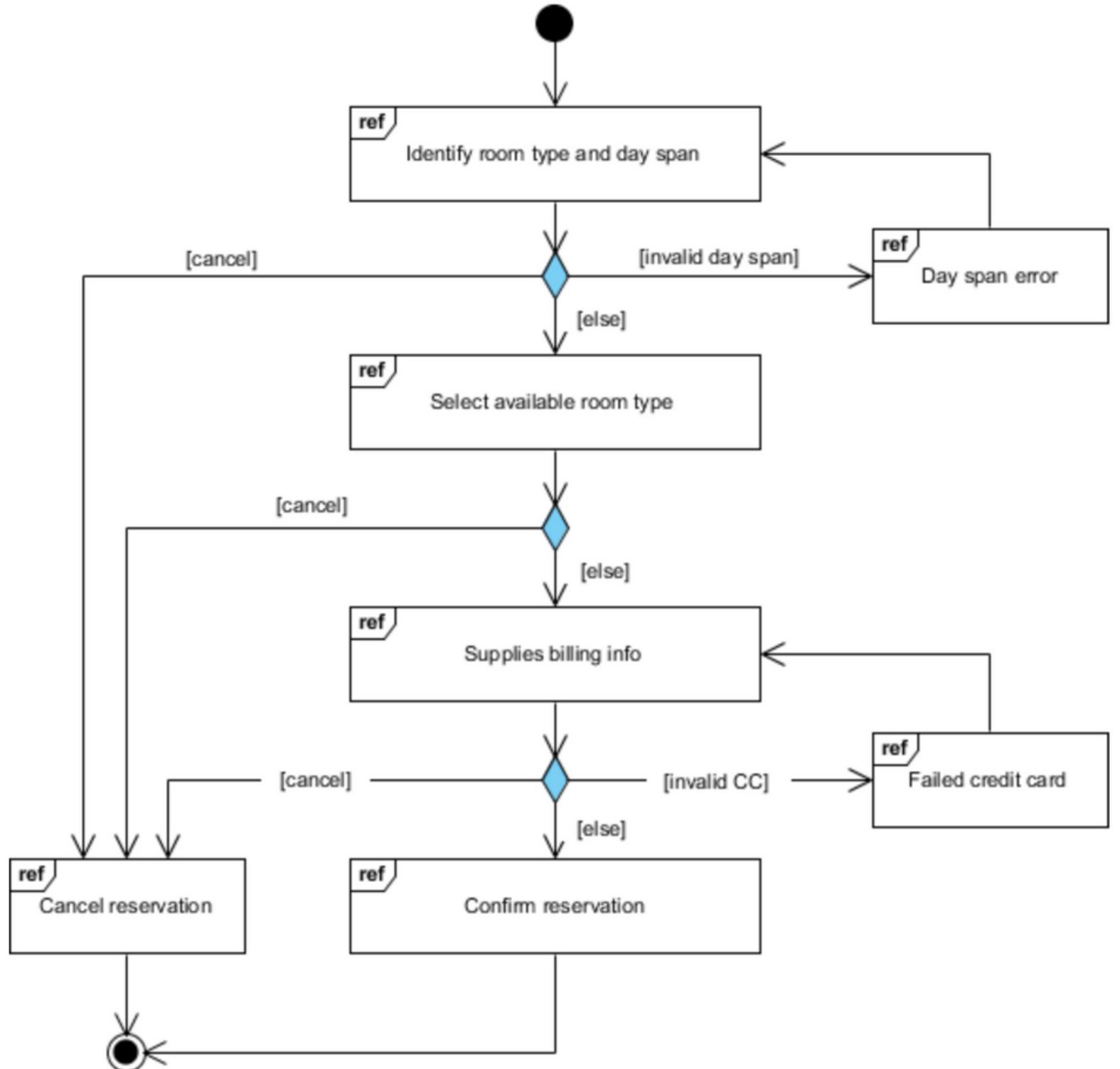
Interaction Diagram

2

Interaction Overview Diagram

Interaction Overview Diagram (IOD) adalah diagram UML yang menggambarkan aliran kontrol antara interaksi. IOD mirip dengan diagram aktivitas, di mana keduanya memvisualisasikan urutan aktivitas. Perbedaannya adalah, untuk IOD, setiap aktivitas individu digambarkan sebagai bingkai yang dapat berisi diagram interaksi

Interaction Diagram Example - Room Reservation



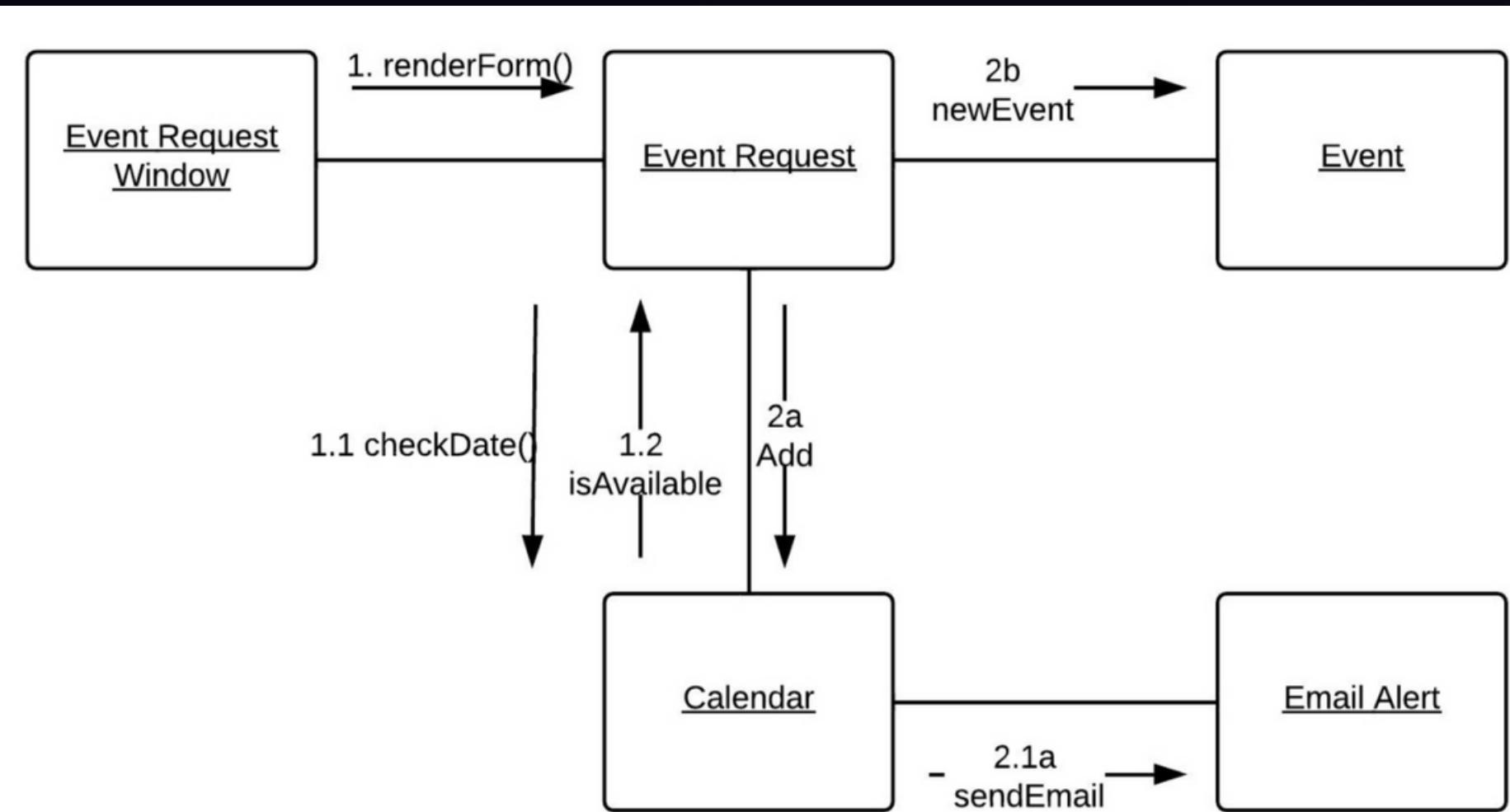
Interaction Diagram

3

Communication Diagram

Communication Diagram adalah diagram UML yang menggambarkan interaksi antara objek-objek dalam suatu sistem. Diagram ini menunjukkan bagaimana objek-objek berkomunikasi satu sama lain dengan mengirimkan pesan.

Hampir sama seperti sequence diagram akan tetapi communication diagram lebih menekankan kepada peranan masing-masing objek pada sistem

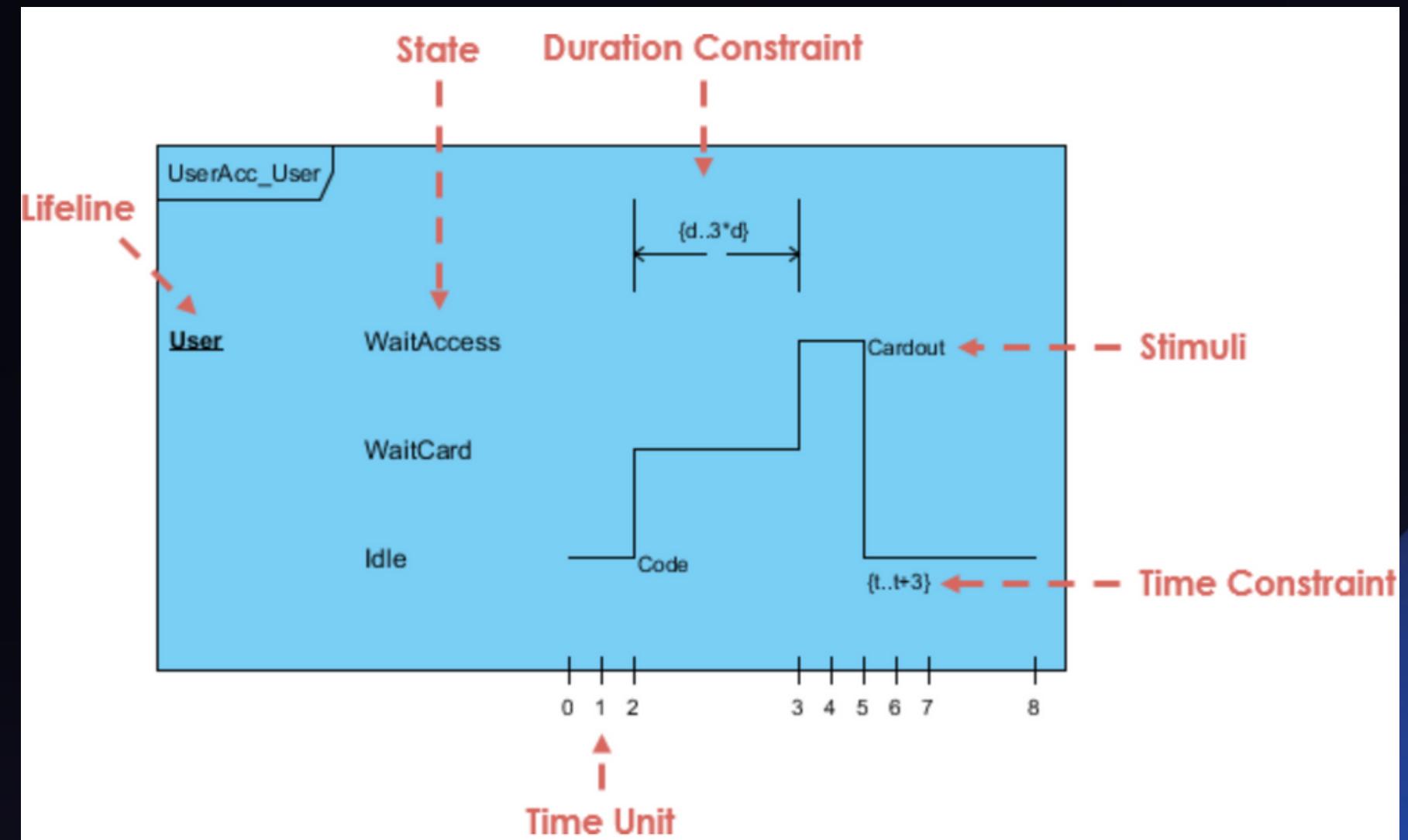


Interaction Diagram

4

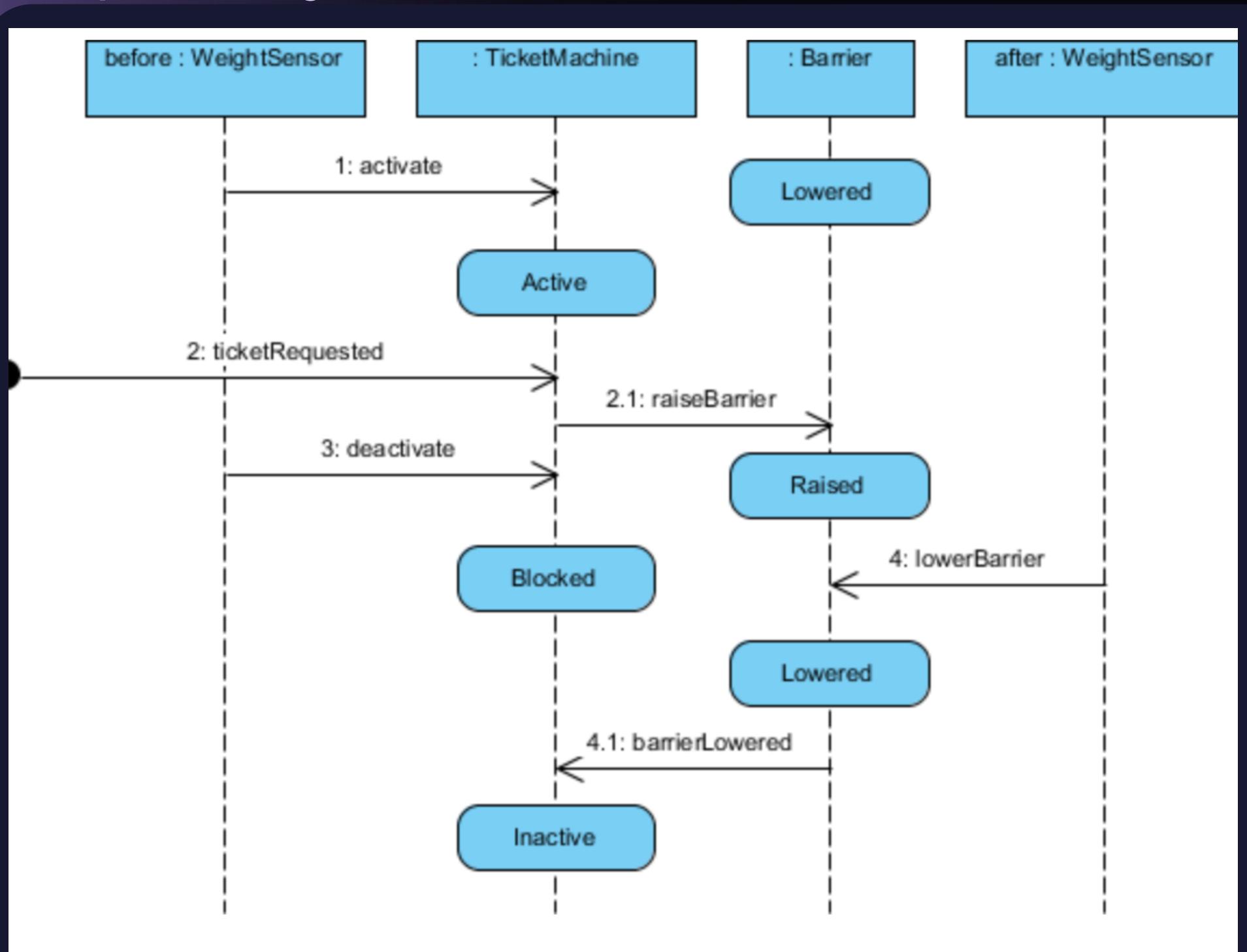
Timing Diagram

Timing diagram adalah bentuk lain dari interaction diagram, dimana fokus utamanya lebih ke waktu. Sebuah timing diagram merupakan bentuk khusus dari sequence diagram. Perbedaan antara timing diagram dan sequence diagram adalah sumbunya dibalik, sehingga waktu meningkat dari kiri ke kanan dan lifeline ditunjukkan dalam kompartemen terpisah yang disusun secara vertikal

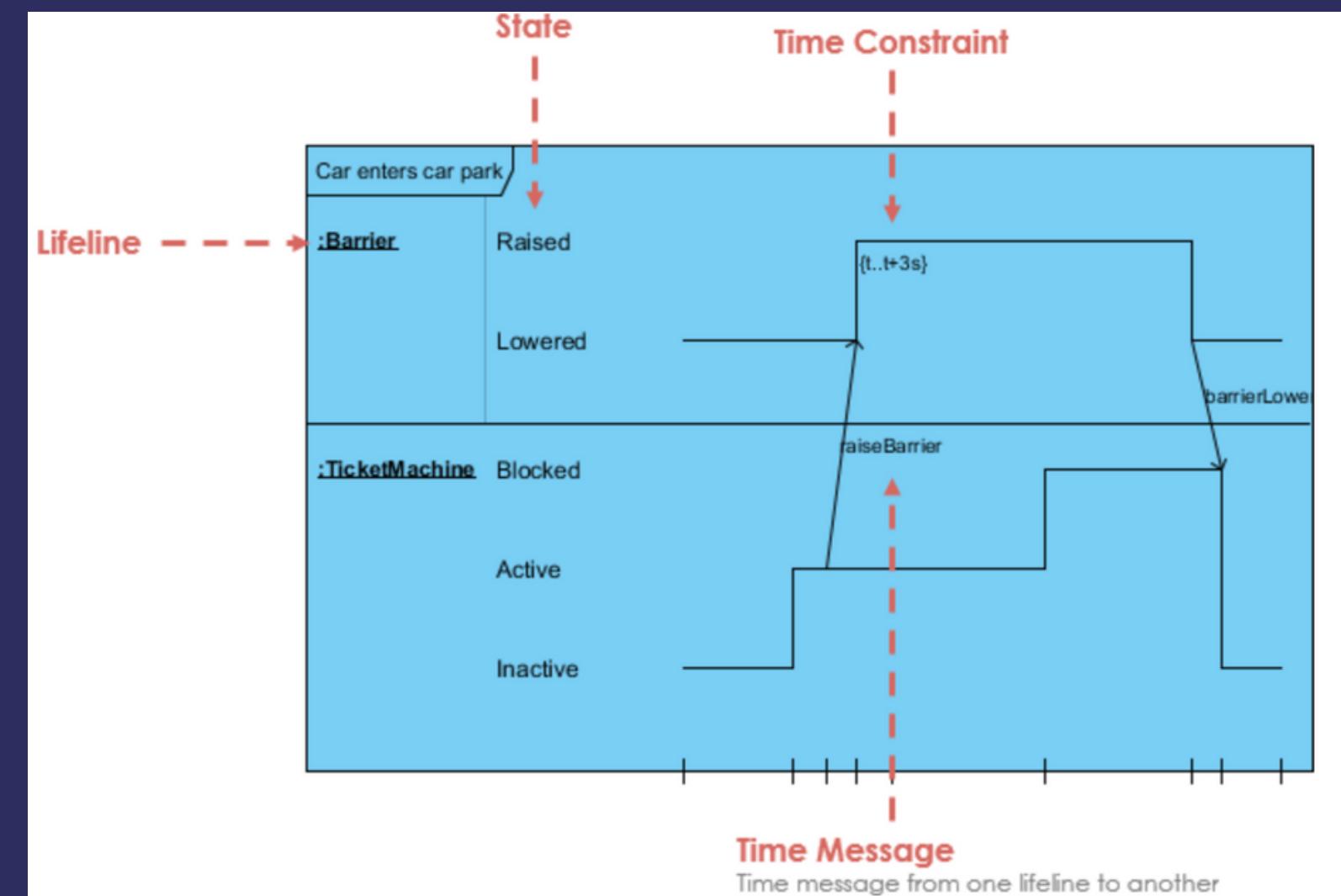


Interaction Diagram

Sequence Diagram



Timing Diagram



Berbagai bagian dari diagram waktu mengacu pada konten diagram urutan.

Any Question



Kesimpulan

UML, atau Unified Modeling Language, adalah bahasa visual yang krusial dalam pengembangan perangkat lunak karena memberikan cara standar untuk merancang, memodelkan, dan mendokumentasikan sistem perangkat lunak. Dalam penggunaannya, UML memungkinkan para pengembang untuk menggambarkan struktur, fungsi, dan interaksi sistem dengan jelas dan terstruktur. Dengan demikian, UML tidak hanya memfasilitasi komunikasi yang efektif antara pengembang dan pemangku kepentingan, tetapi juga memungkinkan pengembangan perangkat lunak yang lebih terstruktur, terorganisir, dan efisien. Terdapat berbagai versi UML yang berkembang seiring waktu, namun inti dari bahasa ini tetap membantu dalam menghadapi kompleksitas proyek-proyek IT dengan berbagai jenis diagram yang masing-masing menggambarkan karakteristik unik dari sistem yang sedang dirancang.