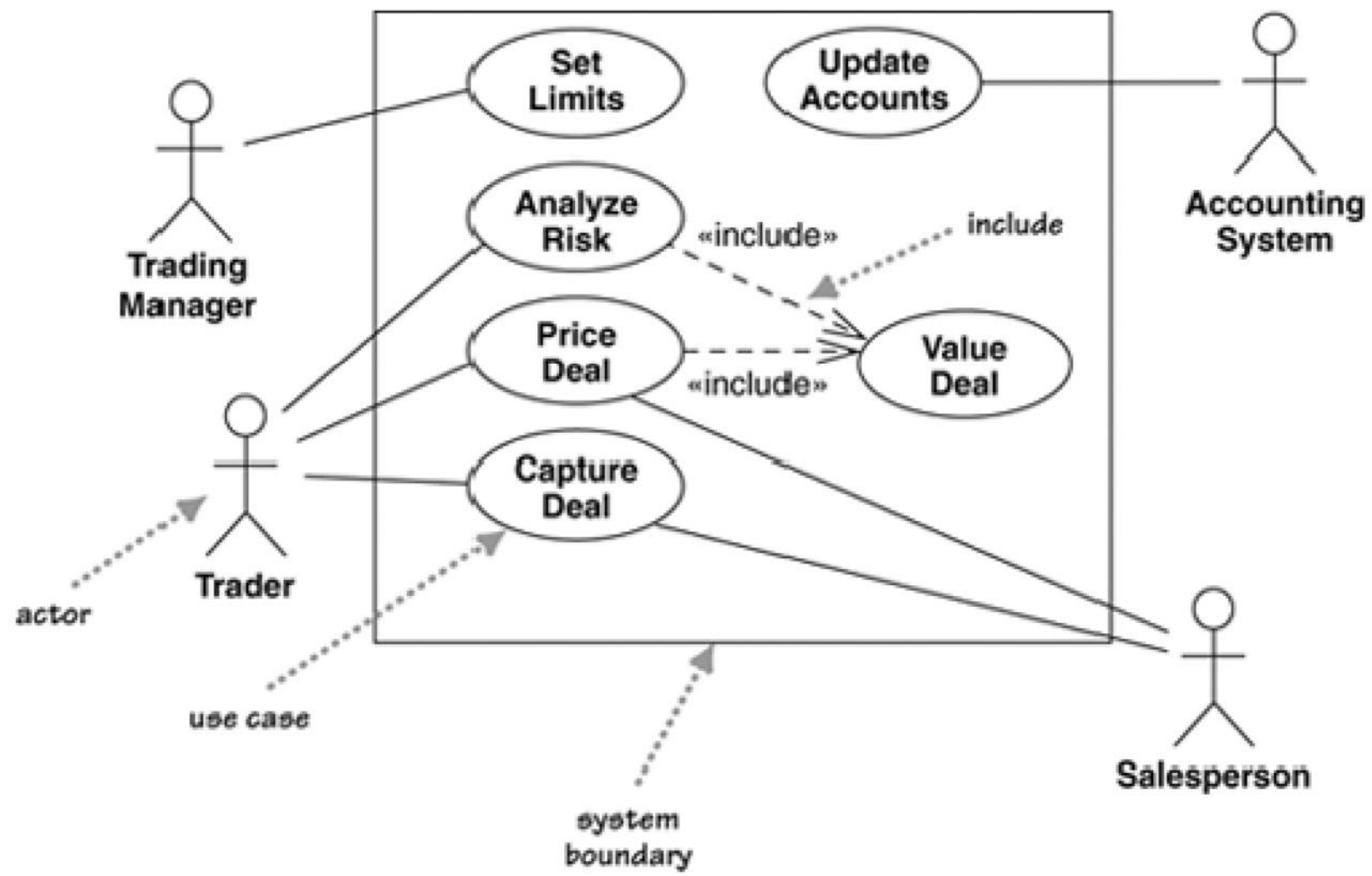


# NOTASI UML

CITRA N., S.SI, MT  
SISFO - UNIKOM

# Level Use Case

- Use Case memiliki dua istilah :
  - System use case : Interaksi dengan sistem.
  - Business use case : Bisnis interaksi dengan konsumen atau kejadian.
- Cookburn menyarankan pembedaan level :
  - Sea level : Interaksi sistem dengan actor utama.
  - Fish level : Use case yang ada karena include dari use case sea-level.
  - Kite level : Menggambarkan sea-level use case untuk interaksi bisnis yang lebih luas.



# Informasi pada Use Case Text

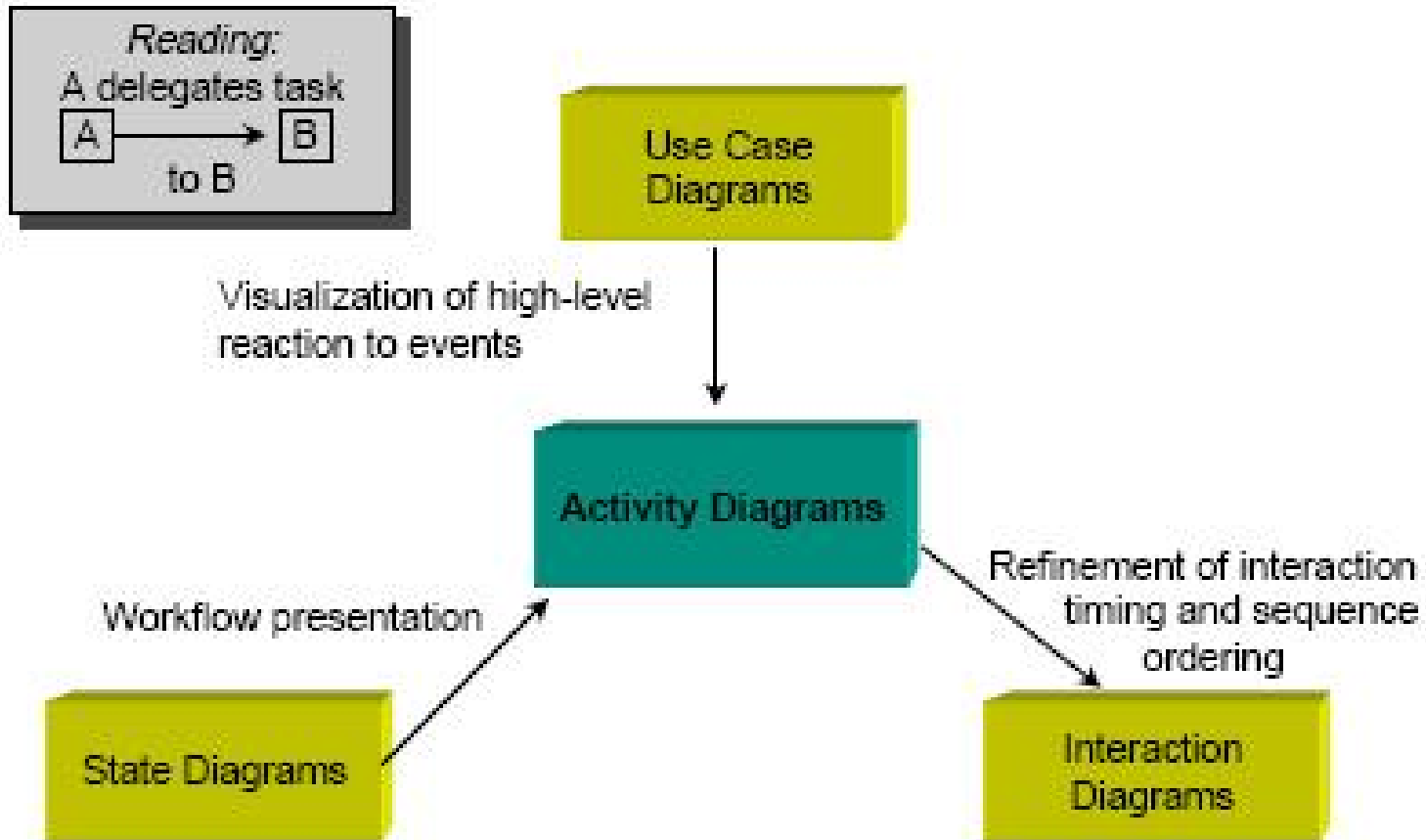
1. Objective/Goal : Tujuan dari use case.
2. Actors : Pelaku.
3. Pre-condition : Kondisi yang harus dipenuhi sebelum use case dimulai.
4. Guarantee/result : Kondisi yang harus dipenuhi setelah use case selesai.
5. Trigger : Kejadian yang mampu menjadi pemicu terjadinya sebuah use case.
6. Relationship : Hubungan dengan use case lain.
7. Scenario : Langkah-langkah.

<b>Use Case:</b>	Use Case Name
<b>Short Description:</b>	A Brief Description of the Use Case
<b>Pre-Conditions:</b>	A description of the conditions that must be satisfied before the use case is invoked
<b>Post-Conditions :</b>	A description of what has happened at the end of the use case
<b>Main Flow:</b>	A list of the system interactions that take place under the most common scenario. For example, for "withdraw money", this would be "enter card, enter pin, etc..."
<b>Alternate Flow(s):</b>	A description of possible alternative interactions.
<b>Exception Flow(s):</b>	A description of possible scenarios where unexpected or unpredicted events have taken place

Use Case Name	Withdraw Name
Primary Actor	Customer
Supporting Actor(s)	Bank Accounting System
Summary	Customer withdraws cash from the ATM system by inserting his or her card, entering the correcting PIN, selecting an account, and entering an amount. The ATM system validates the card, PIN, account and amount with the Bank Accounting System.
Pre - Conditions	<ol style="list-style-type: none"> <li>1. ATM has money and supplies.</li> <li>2. Bank accounting system is working.</li> </ol>
Normal Flow of Events	<ol style="list-style-type: none"> <li>1. User inserts ATM card.</li> <li>2. ATM reads and validates bank ID and account number with bank account system.</li> <li>3. User enters PIN number.</li> <li>4. ATM validates PIN with bank accounting system.</li> <li>5. User selects account.</li> <li>6. User enters amount to withdraw.</li> <li>7. ATM validates amount with bank accounting system.</li> <li>8. ATM dispenses cash and receipt.</li> <li>9. ATM logs transactions.</li> <li>10. User takes card, cash and receipt.</li> </ol>

Extensions	<ol style="list-style-type: none"> <li>1. Non-ATM card entered.ATM card inserted incorrectly.</li> <li>2. ATM card inserted incorrectly.</li> <li>3. Bank ID or account invalid.</li> <li>4. Card is from ineligible bank.</li> <li>5. Card is stolen.</li> <li>6. Customer does not enter PIN in time.</li> <li>7. PIN is invalid.</li> </ol> <p>5.1. Account is invalid.</p> <p>6.1. Amount is invalid or over maximum allowed.</p> <p>7.1. Insufficient funds in account.</p>
Post – Conditions	<ol style="list-style-type: none"> <li>1. User's account balance is adjusted.</li> <li>2. ATM's money inventory is adjusted.</li> <li>3. ATM's supply inventory is adjusted.</li> </ol>

# Activity Diagram: Peran di UML





# Activity Diagram dipakai dengan cara :

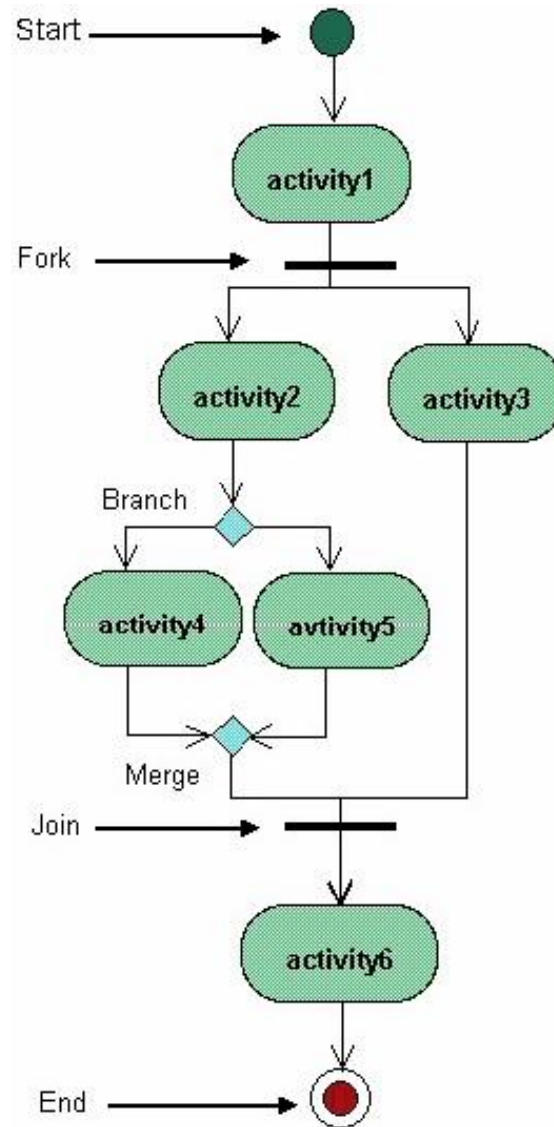
- Memodelkan workflow

Fokus pada aktivitas seperti yang dilihat oleh aktor pada use case diagram









- Memodelkan operasi

Mirip cara kerja flowchart untuk memperjelas use case Text (Skenario)

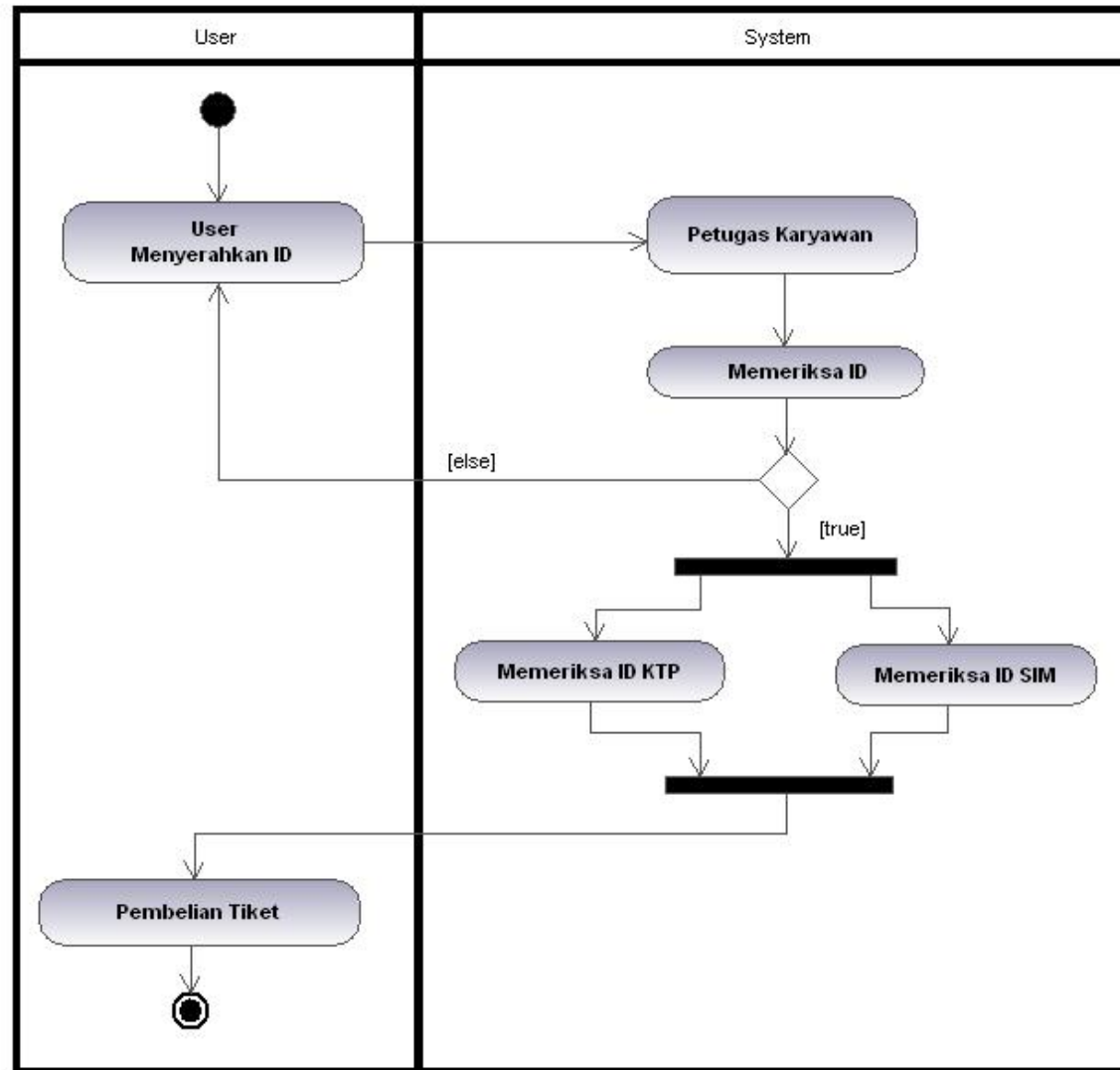
# Diagram Aktivitas : How To Draw



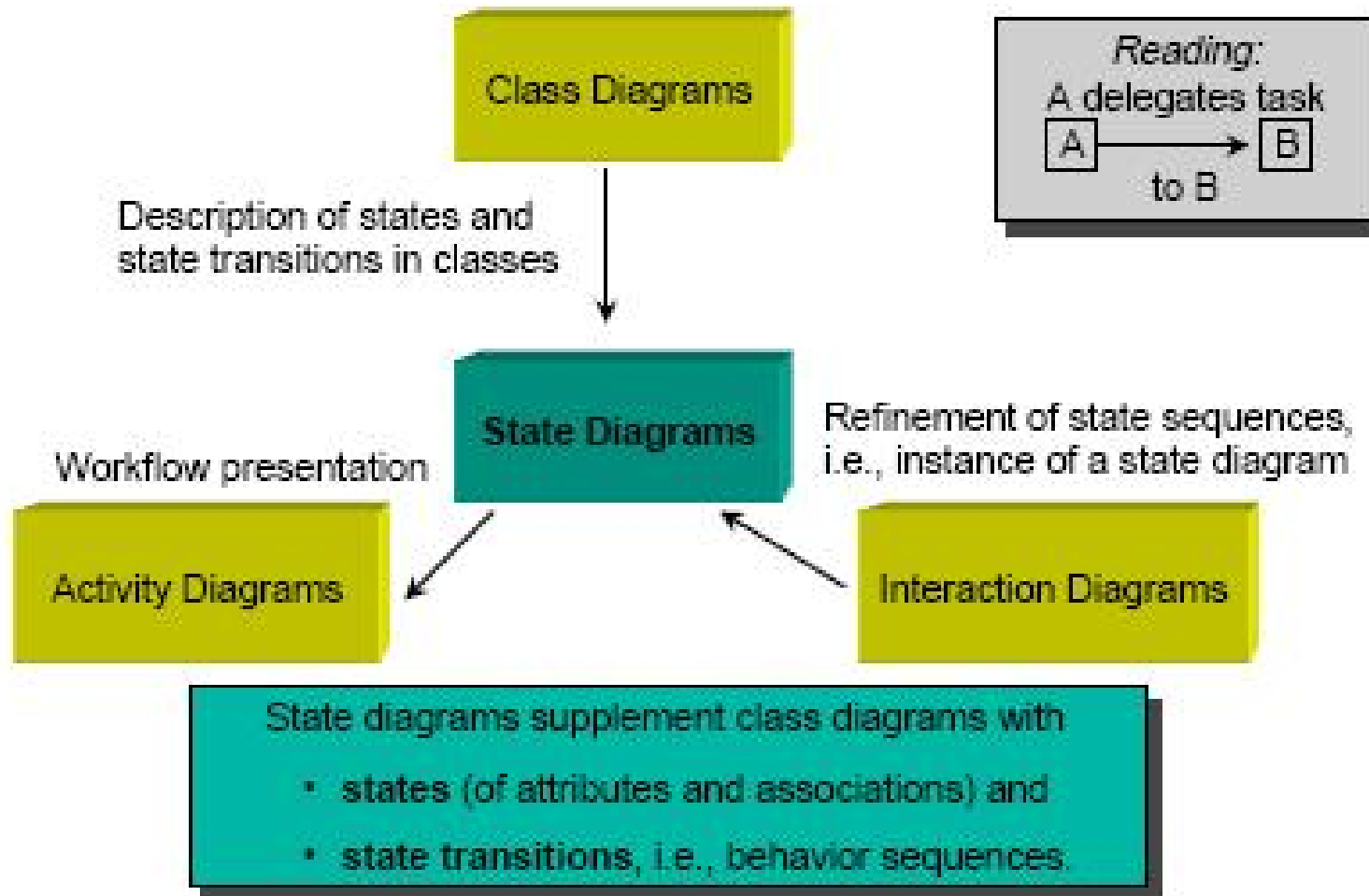
# Notasi

Simbol	Keterangan
	InitialNode (Titik Awal)
	FinalNode (Titik Akhir)
	Action (Aktivity)
	Decision/Merger (Pilihan)
	ForkNode/JoinNode (Kegiatan Pararel)
	AcceptEventAction (Tanda waktu)
	AcceptEventAction (Tanda Penerimaan)
	FlowFinal (Aliran Akhir)

# Contoh Activity dengan swimlane :

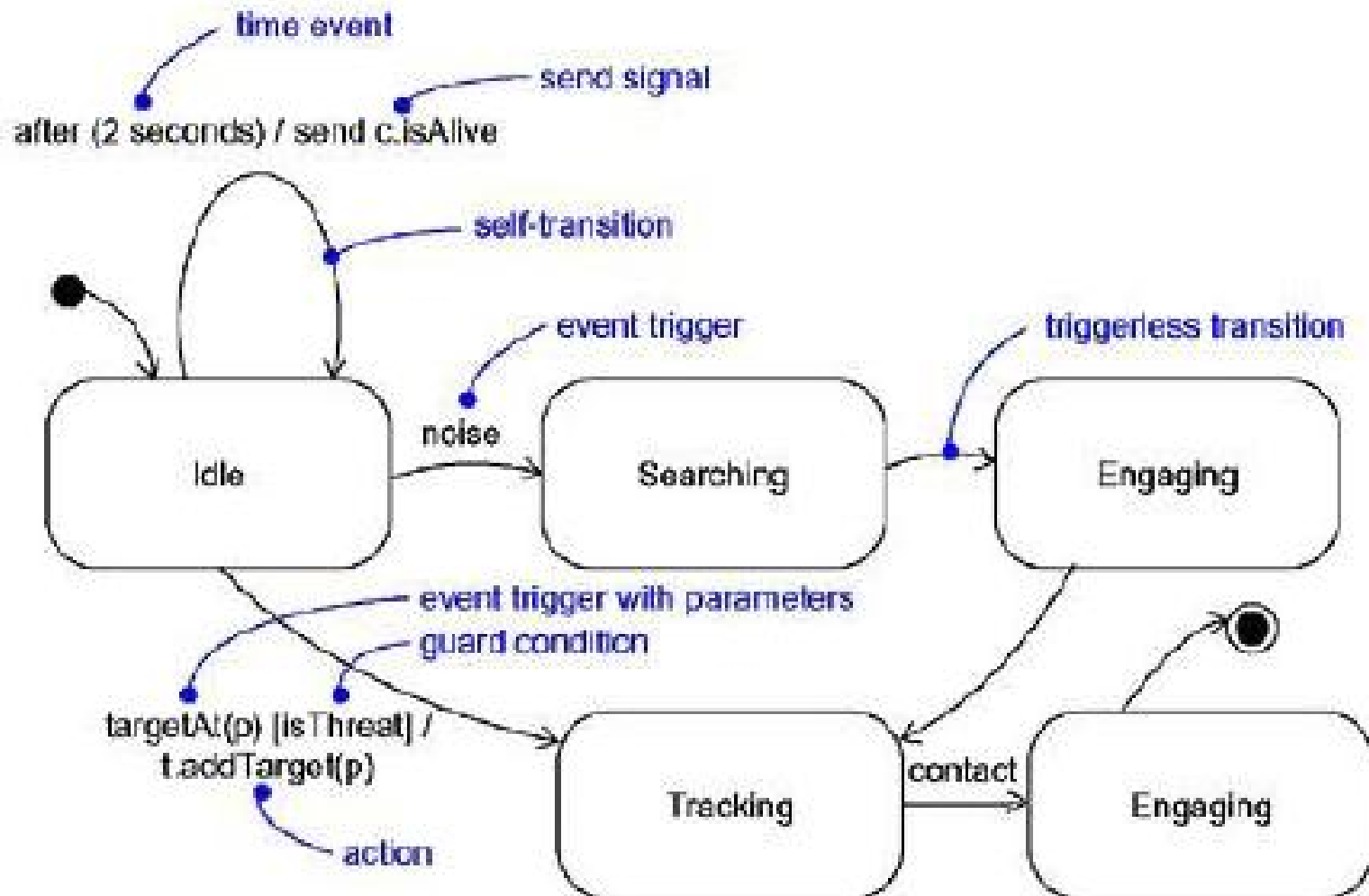


# Diagram State : Peran di UML



# State Transitions

- Transisi – Hubungan antara dua state yang mengindikasikan bahwa suatu objek telah selesai melakukan suatu aksi dan akan berpindah pada aksi berikutnya



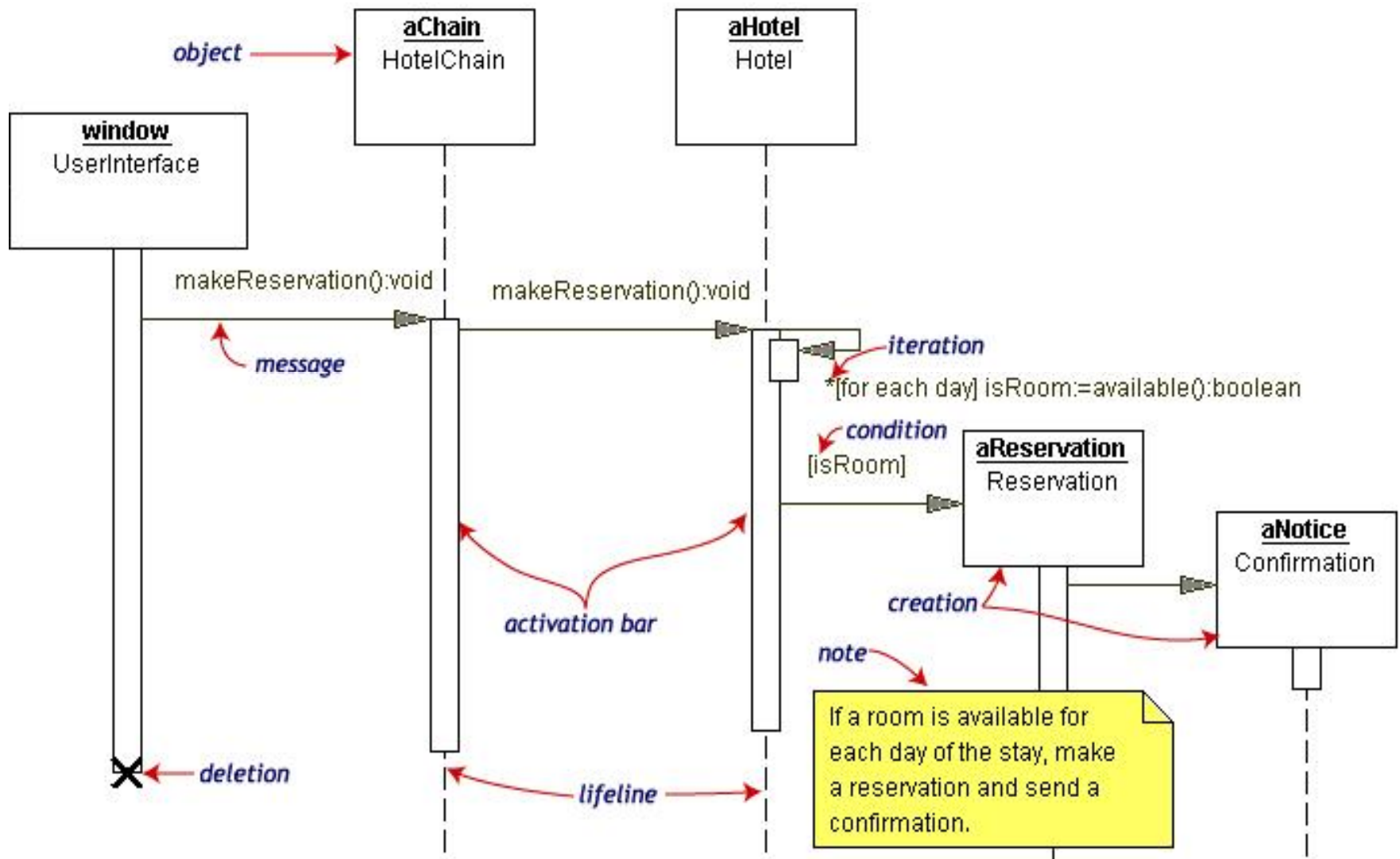
# Definisi Diagram Sequence

- Sequence diagram adalah suatu diagram yang menggambarkan interaksi antar obyek dan mengindikasikan komunikasi diantara obyek-obyek tersebut.
- Diagram ini juga menunjukkan serangkaian pesan yang dipertukarkan oleh obyek-obyek yang melakukan suatu tugas atau aksi tertentu. Obyek-obyek tersebut kemudian diurutkan dari kiri ke kanan, aktor yang menginisiasi interaksi biasanya ditaruh di paling kiri dari diagram.

- Partisipan : obyek atau entitas yang bertindak dalam sequence diagram
- Message : komunikasi antar obyek partisipan
- Terdapat 2 tipe garis yaitu vertikal dan horisontal
  - Vertikal : waktu → maju berdasarkan waktu
  - Horisontal : obyek mana yang beraksi



# Penggunaan Notasi



# Message

- Interaksi antara 2 objek yang di bentuk sebagai pesan yang dikirim dari satu objek ke objek lain.
  - Biasanya di implementasikan dengan operasi pemanggilan yang sederhana
- Pesan digambarkan sebagai anak panah antara life line dari 2 objek
  - Pemanggilan diri sendiri di perbolehkan
  - Waktu yang dibutuhkan penerima pesan untuk memproses pesan dinyatakan dengan *activation-box*
- Pesan dilabeli secara minimal dengan nama pesan
  - Argument dan informasi kontrol (kondisi,iterasi) kemungkinan di masukan
  - Lebih memilih menggunakan deskripsi tekstual bila actor adalah sumber/target pesan.

# Type Message

Synchronous



Asynchronous



Simple



Create

`<<create>>`



Destroy

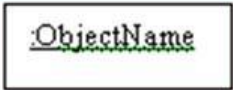

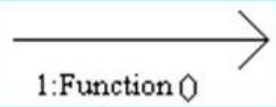
`<<destroy>>`



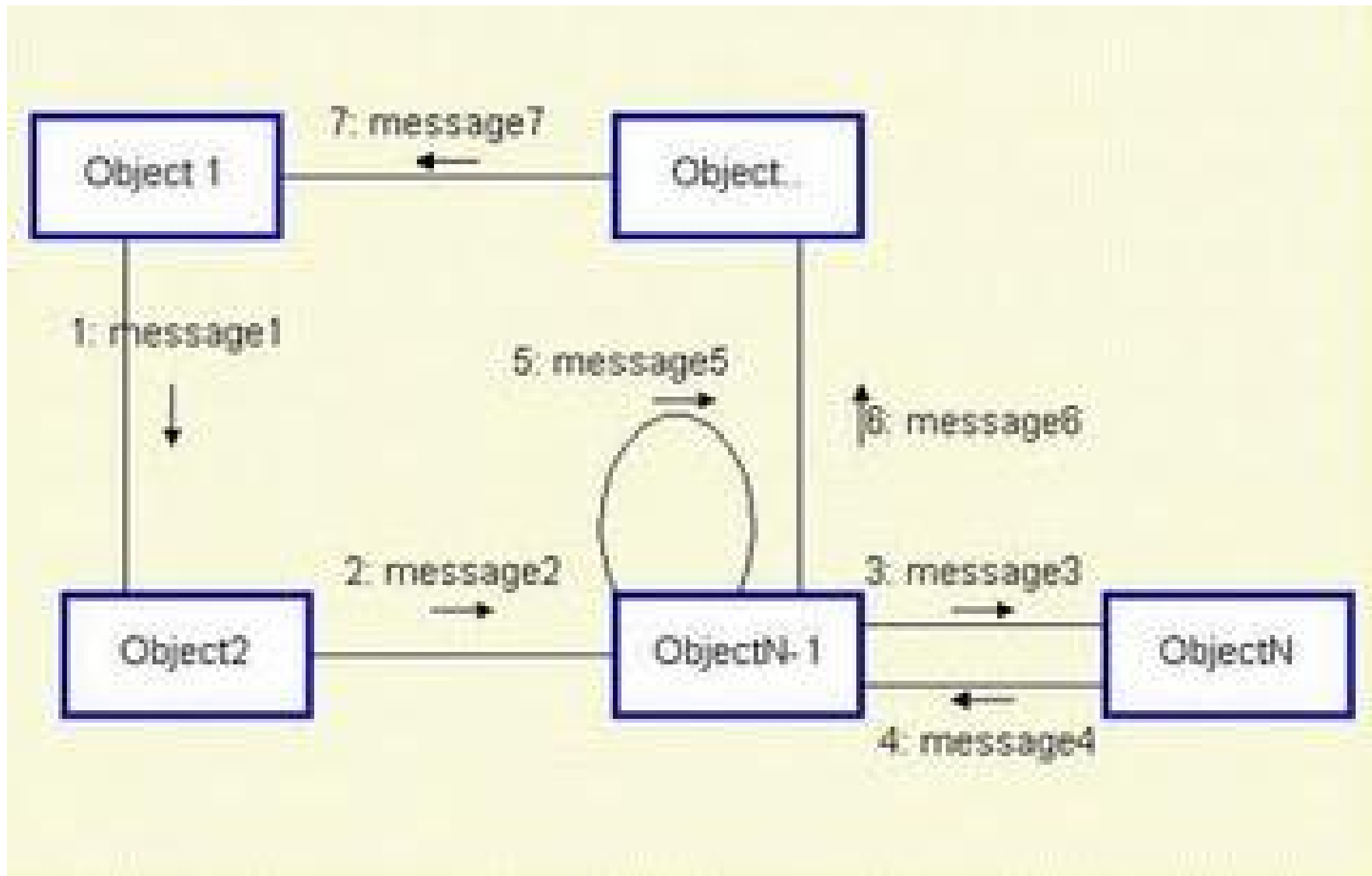
# Definisi Diagram Collaboration

- Collaboration diagram menggambarkan interaksi antar objek seperti *sequence diagram*, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian *message*. Setiap *message* memiliki *sequence number*, di mana *message* dari level tertinggi memiliki nomor 1.
- Sequence diagram dan collaboration diagram mempunyai tipe yang sama, yakni merepresentasikan informasi yang sama, dan sequence diagram dapat ditransformasikan ke collaboration diagram atau sebaliknya. Dalam sequence diagram lebih menekankan pada urutan-urutan waktu proses atau interaksi antar objek-objek.

# Notasi

Elemen dan keterangan	Lambang
Object: Objek-objek yang berinteraksi satu dengan yang lain dalam sebuah sistem. Digambarkan dengan sebuah segi empat dengan nama objeknya di dalamnya, di dahului dengan titik dua dan digarisbawahi.	
Relation/Association: Suatu keterkaitan yang menghubungkan objek-objek yang satu dengan yang lain. Tanda lain dapat di tambahkan pada ujung-ujungnya untuk memberikan gambaran tingkatan	
Messages: Anak panah dari objek awal ke objek tujuan yang menunjukkan interaksi antara objek-objek tersebut. Angka diberikan untuk merepresentasikan urutan dari interaksinya.	

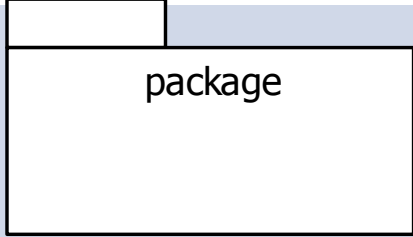
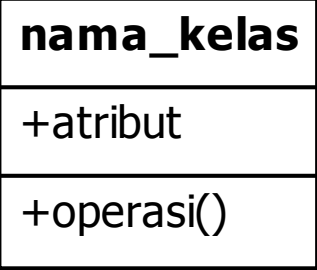
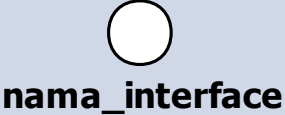

# Pengiriman message



# Definisi Diagram Class



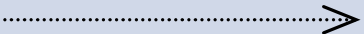

- Diagram class sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.
- Class menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).
- Class diagram menggambarkan struktur dan deskripsi class, package dan objek beserta hubungan satu sama lain seperti containment, pewarisan, asosiasi, dan lain-lain.

# NOTASI

Deskripsi	Notasi
<i>package</i> merupakan sebuah bungkusan dari satu atau lebih kelas	
kelas pada struktur sistem	
sama dengan konsep <i>interface</i> dalam pemrograman berorientasi objek	
relasi antar kelas dengan makna umum, asosiasi biasanya juga disertai dengan <i>multiplicity</i>	

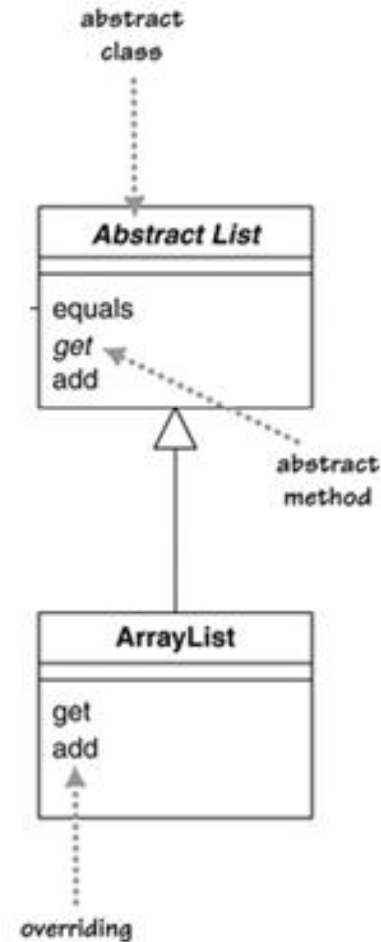


# NOTASI

Deskripsi	Notasi
relasi antar kelas dengan makna kelas yang satu digunakan oleh kelas yang lain, asosiasi biasanya juga disertai dengan <i>multiplicity</i>	
relasi antar kelas dengan makna generalisasi-spesialisasi (umum khusus)	
relasi antar kelas dengan makna kebergantungan antar kelas	
relasi antar kelas dengan makna semua-bagian ( <i>whole-part</i> )	

# Abstract Class

Abstract Class digunakan pada class yang tidak bisa diinstantiasi, harus diturunkan kedalam class non-abstract. Memiliki satu atau lebih metode abstract sedangkan metode abstract tidak memiliki implementasi. Implementasi dilakukan oleh class yang menurunkan. Dinotasikan italics pada nama.



# Sifat Atribut dan Metoda (Visibility)

- ***Private (-)***

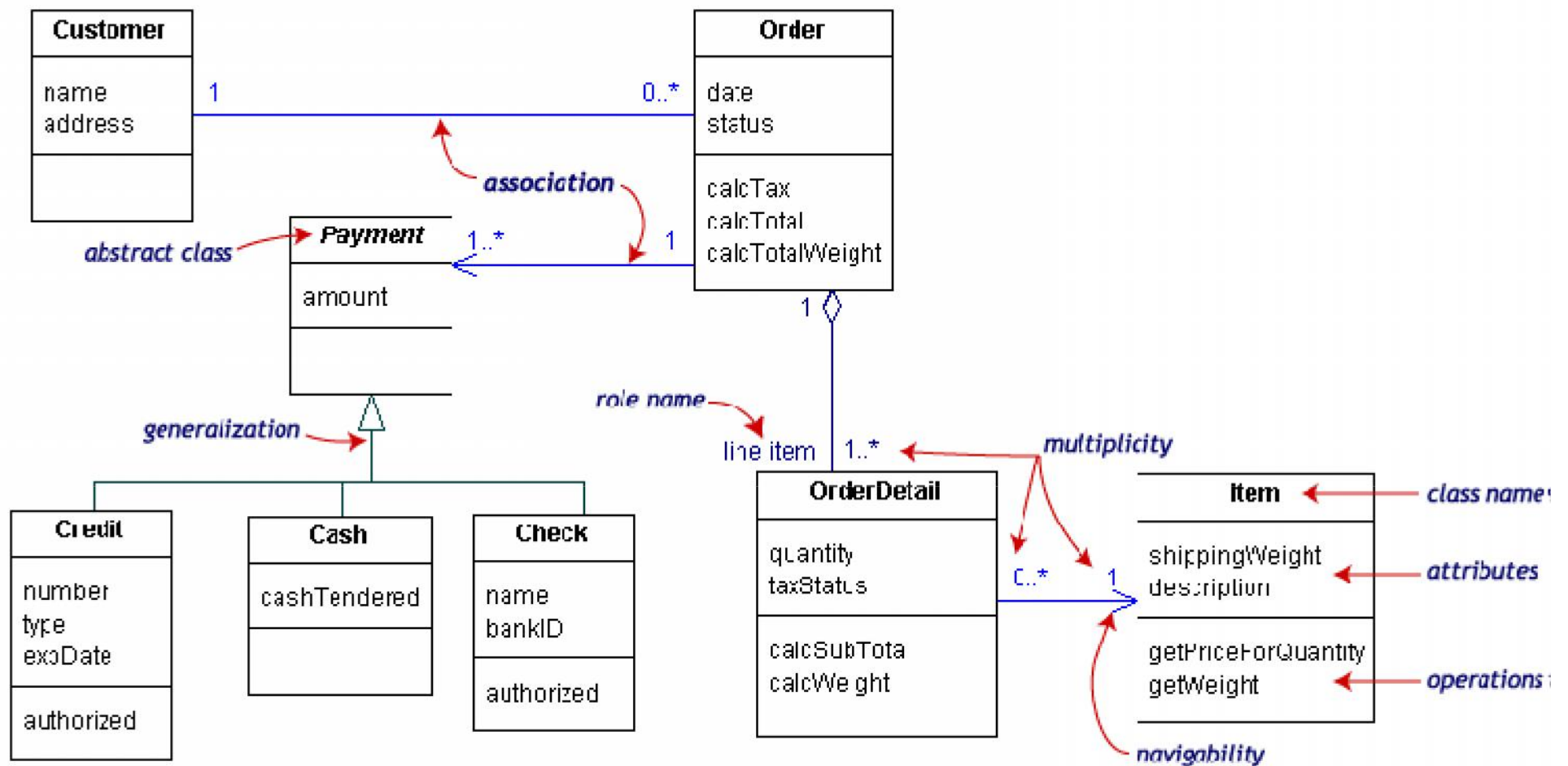
Tidak dapat dipanggil dari luar class yang bersangkutan

- ***Protected (#)***

Hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

- ***Public (+)***

Dapat dipanggil oleh siapa saja

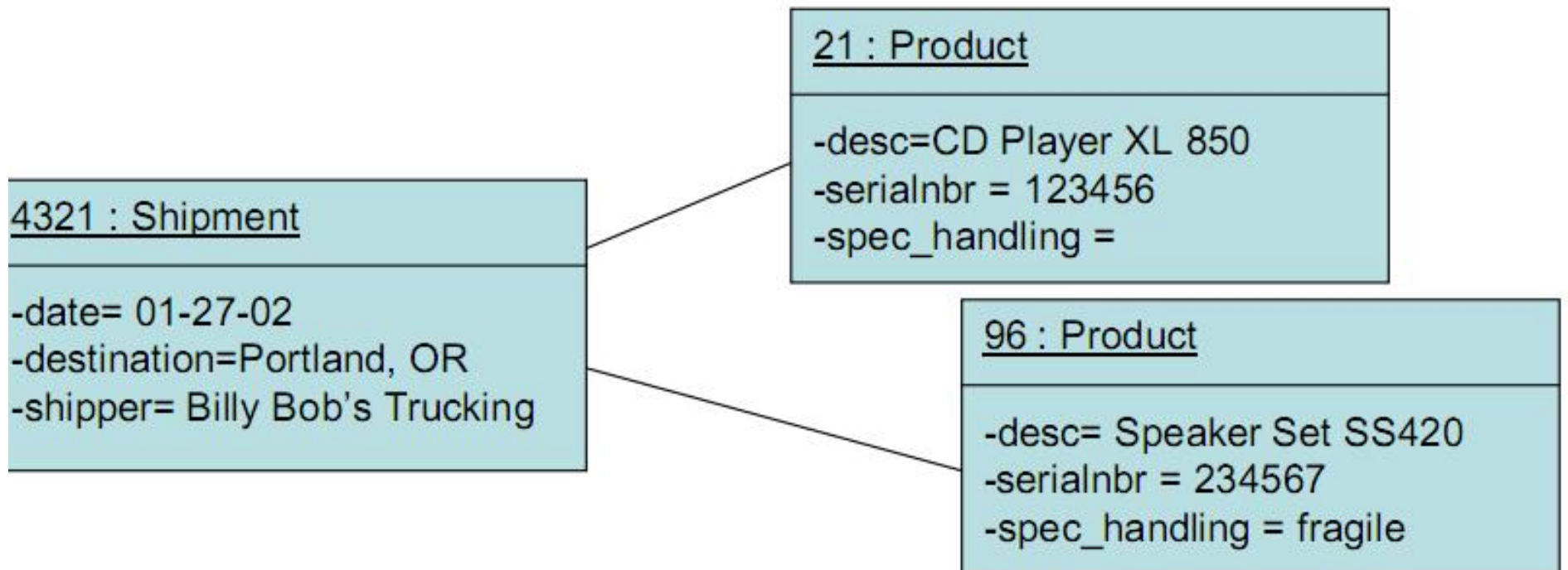
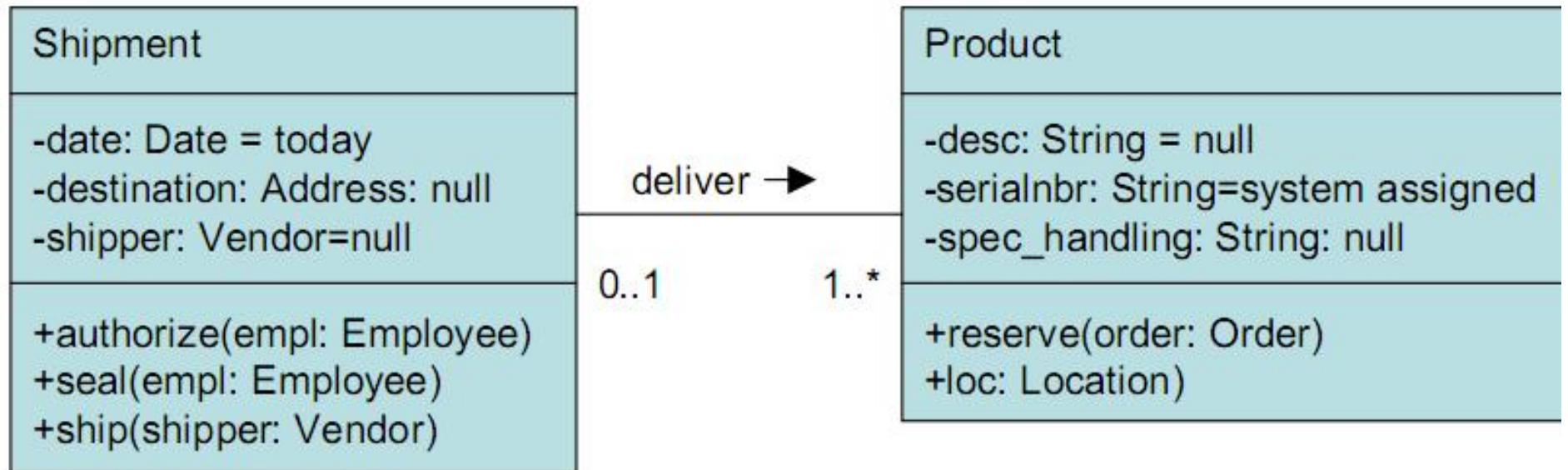


# Diagram Objek

- Objek diagram berasal dari kelas objek diagram diagram sehingga tergantung pada diagram kelas.
- Konsep-konsep dasar serupa untuk kelas objek diagram dan diagram. Obyek diagram juga mewakili pandangan statis dari sebuah sistem tetapi pandangan statis ini merupakan sebuah snapshot dari sistem pada saat tertentu.
- Object diagram digunakan untuk membuat satu set benda dan hubungan mereka sebagai contoh.

# NOTASI



Deskripsi	Notasi
objek dari kelas yang berjalan saat sistem dijalankan	<div><u>nama objek : nama kelas</u> atribut = nilai</div>
relasi antar objek	_____

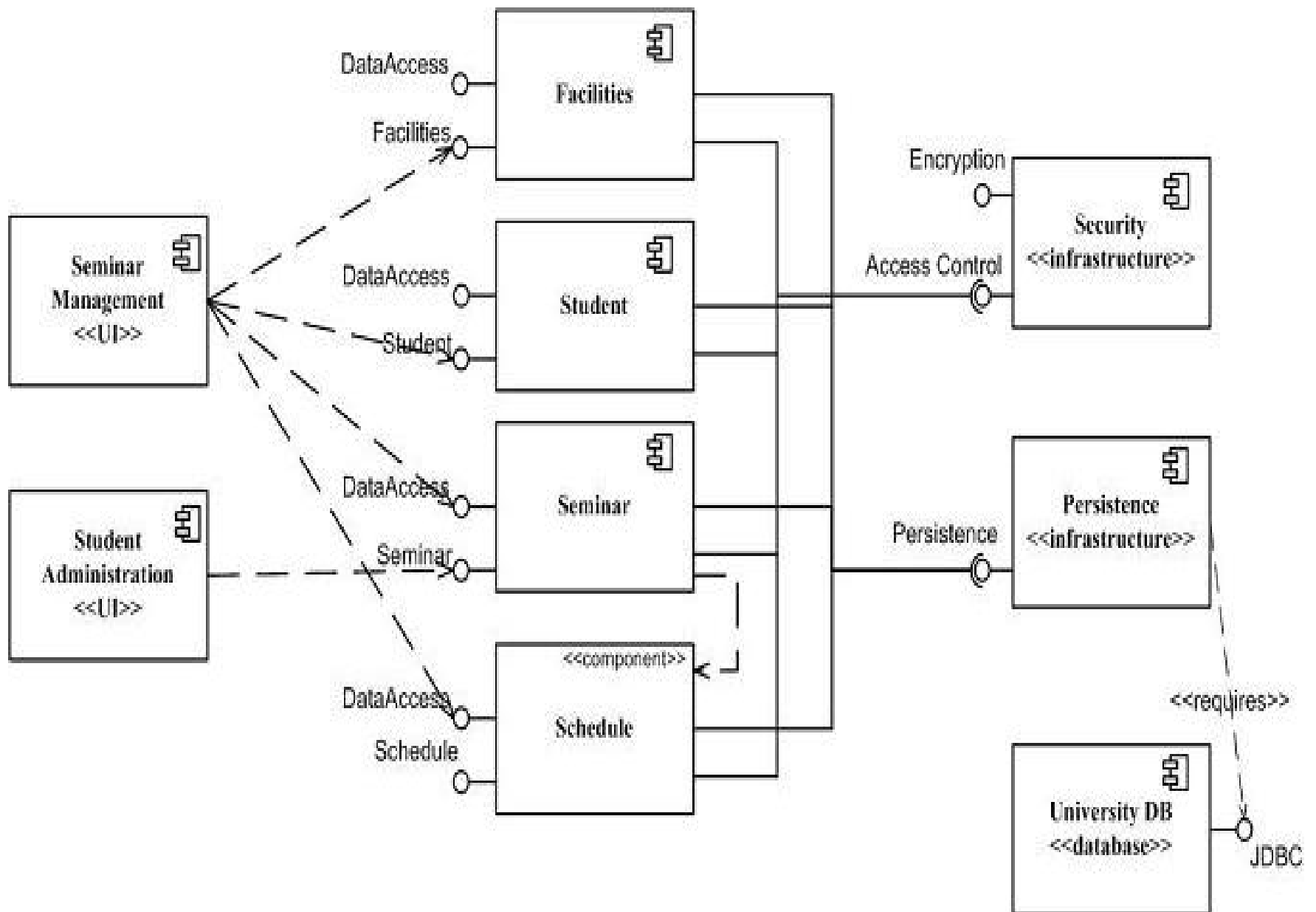


# Definisi

- Component diagram menggambarkan struktur dan hubungan antar komponen peranti lunak, termasuk ketergantungan (dependency) diantaranya.
- Komponen peranti lunak adalah modul berisi code, baik berisi source code maupun binary code, baik library maupun executable, baik yang muncul pada compile time, link time maupun run time.
- Pada umumnya komponen terbentuk dari beberapa class dan/atau package, tapi dapat juga dari komponen-komponen yang lebih kecil.
- Komponen dapat juga berupa interface, yaitu kumpulan layanan yang disediakan sebuah komponen untuk komponen lain.



Elemen dan descriptionnya	Simbol
<p>Komponen adalah sebuah blok bangunan fisik dari sistem. Hal ini digambarkan sebagai persegi panjang dengan tab.</p>	
<p>Interface Sebuah antarmuka menggambarkan sekelompok operasi digunakan atau dibuat oleh komponen.</p>	

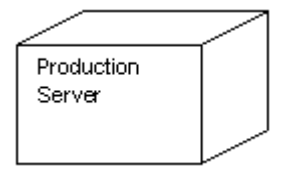


# Fungsi

Deployment/physical diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal .

Sebuah node adalah server, workstation, atau piranti keras lain yang digunakan untuk men-deploy komponen dalam lingkungan sebenarnya. Hubungan antar node (misalnya TCP/IP) dan requirement dapat juga didefinisikan dalam diagram ini.



Elemen dan descriptionnya	Simbol
<b>Node:</b> Elemen yang menyediakan lingkungan eksekusi untuk komponen-komponen sistem. Digambarkan oleh kubus dengan nama obyek di dalamnya, didahului oleh titik dua, dan digarisbawahi	
<b>Koneksi:</b> Serupa dengan relasi / asosiasi yang digunakan dalam diagram kelas untuk menentukan interkoneksi antar node.	