



# UNIFIED MODELLING LANGUAGE

# UNIFIED MODELLING LANGUAGE

---

- ❑ Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem informasi atau piranti lunak.
- ❑ UML menawarkan sebuah standar untuk merancang model sebuah sistem.
- ❑ Seperti bahasa-bahasa lainnya, UML mendefinisikan notasi dan *syntax*/semantik.

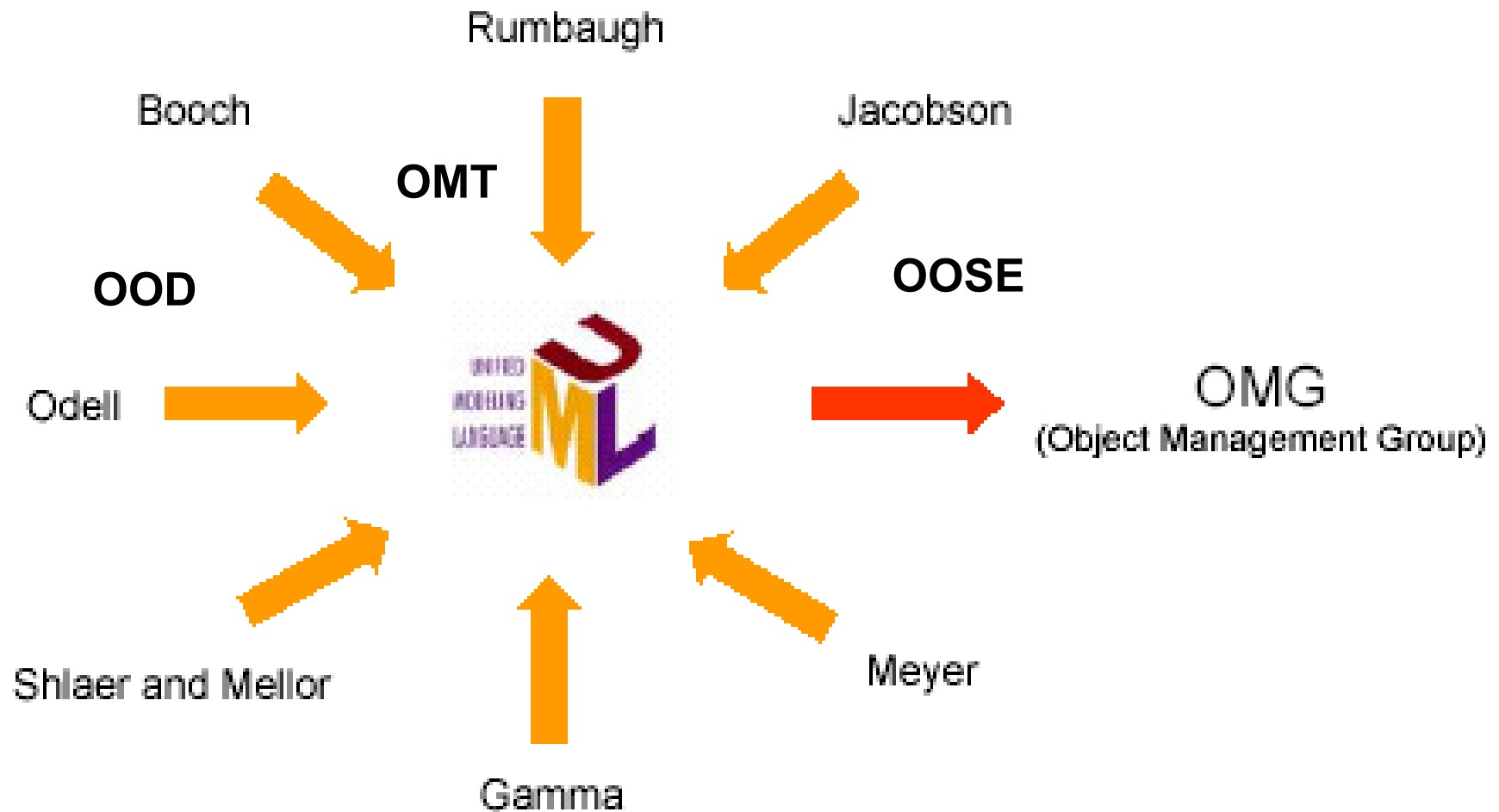
# UNIFIED MODELLING LANGUAGE

---

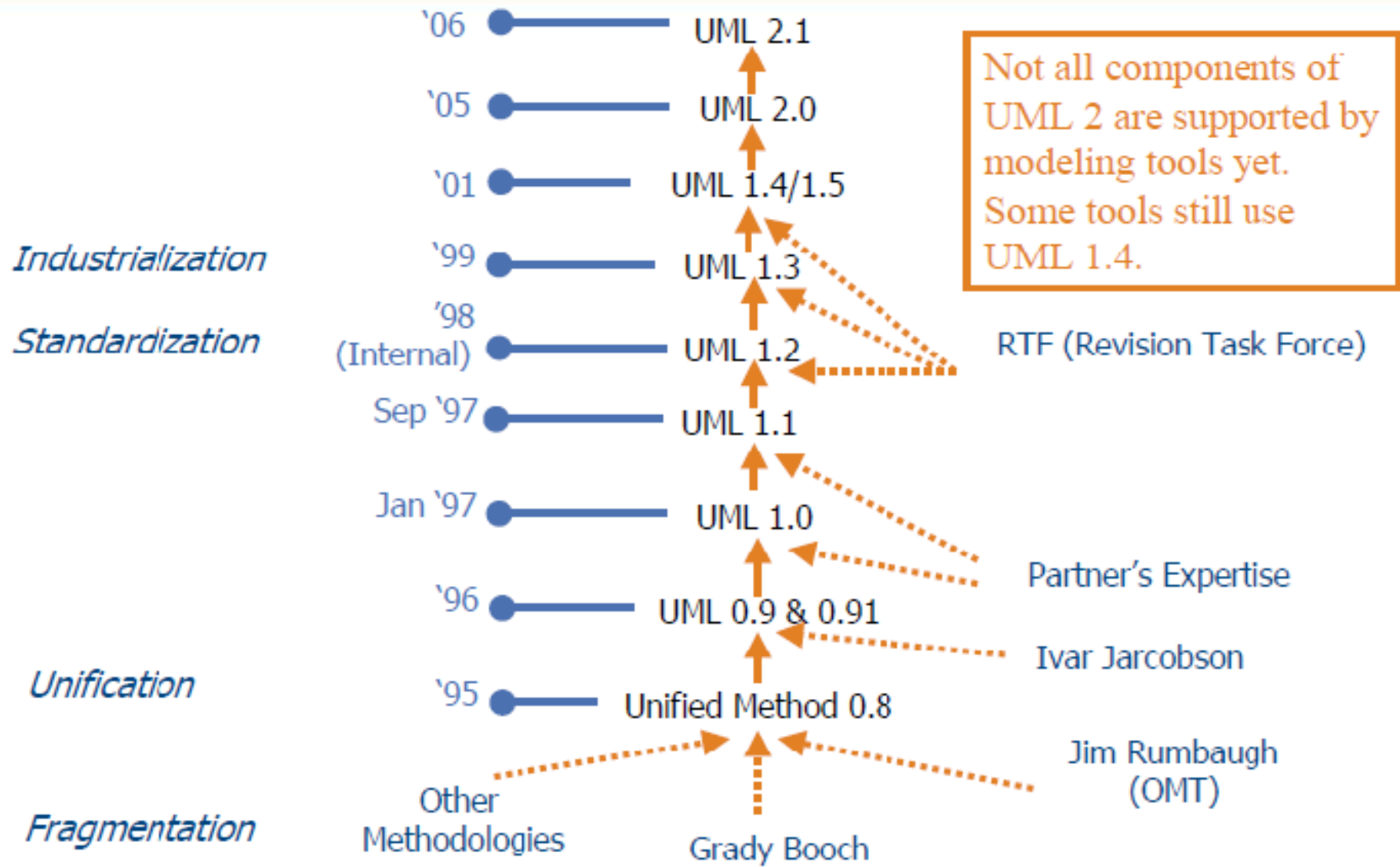
- ❑ Notasi UML merupakan sekumpulan bentuk khusus untuk menggambarkan berbagai diagram piranti lunak.
- ❑ Setiap bentuk memiliki makna tertentu, dan UML *syntax* mendefinisikan bagaimana bentuk-bentuk tersebut dapat dikombinasikan.
- ❑ Notasi UML terutama diturunkan dari 3 notasi yang telah ada sebelumnya: Grady Booch OOD (Object-Oriented Design), Jim Rumbaugh OMT (Object Modeling Technique), dan Ivar Jacobson OOSE (Object-Oriented Software Engineering).

# UML Derivative

---

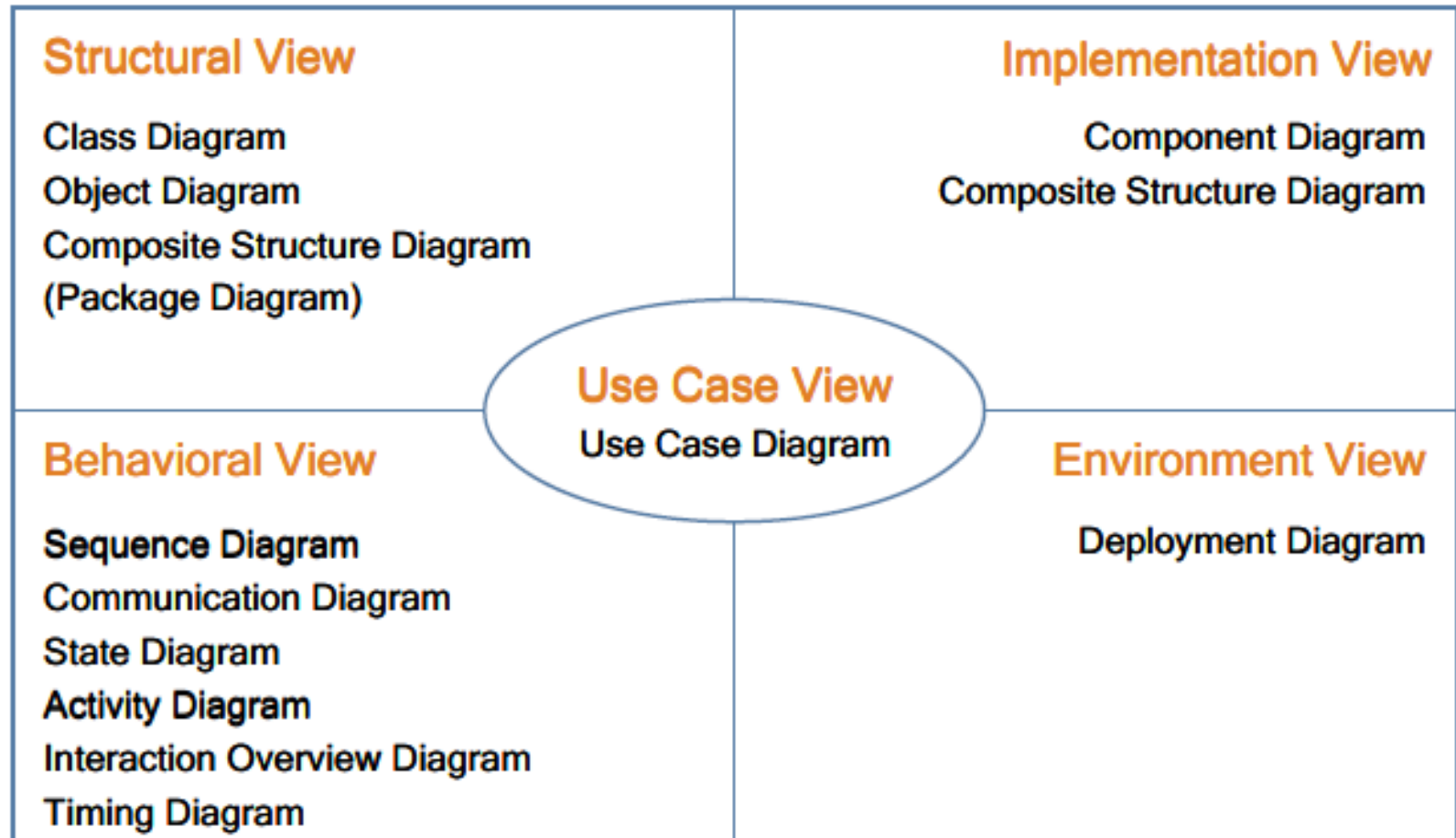


# Long Story of UML

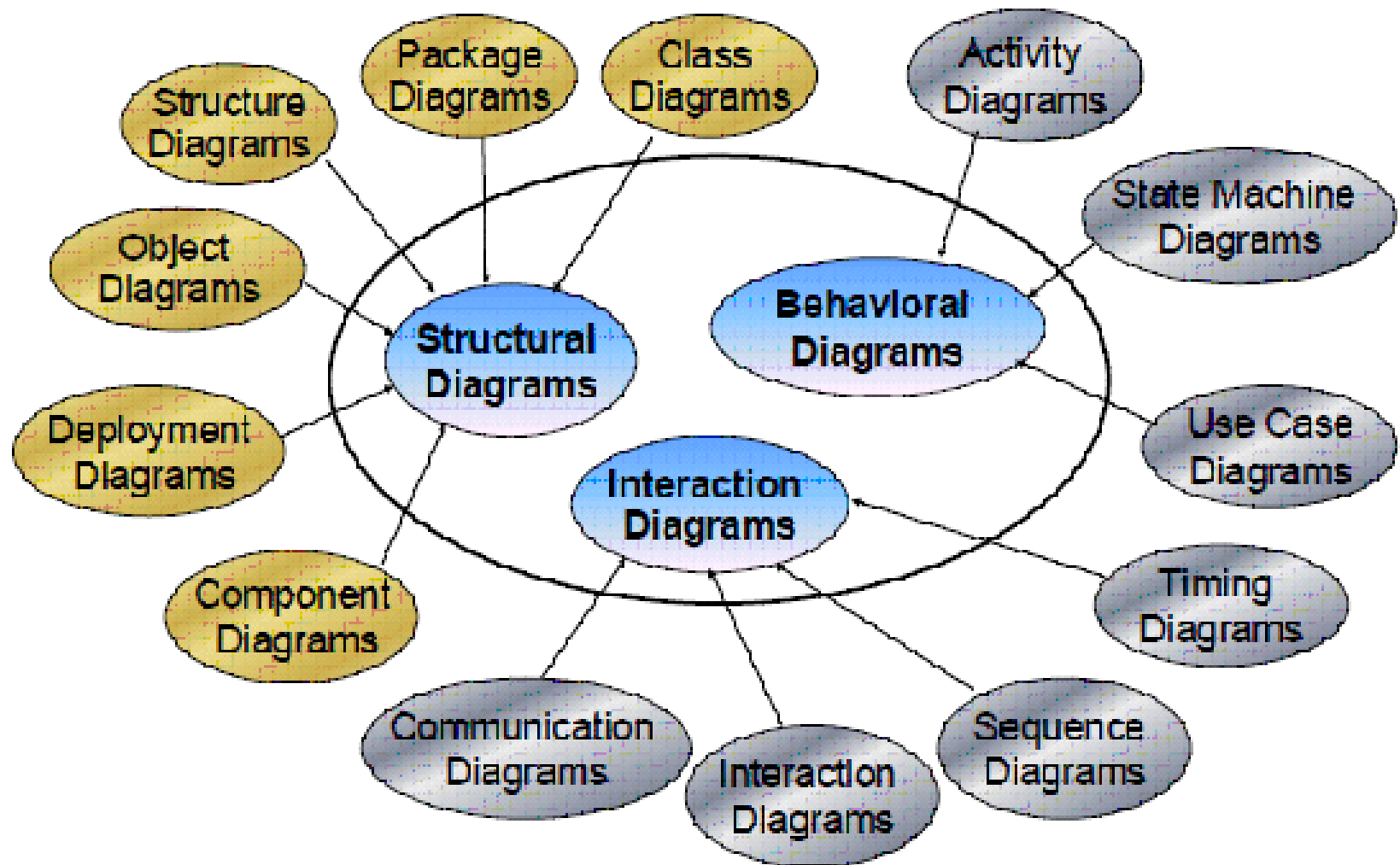


# UML Architectural Views and Diagrams

UML defines 13 diagrams that describe 4+1 architectural views

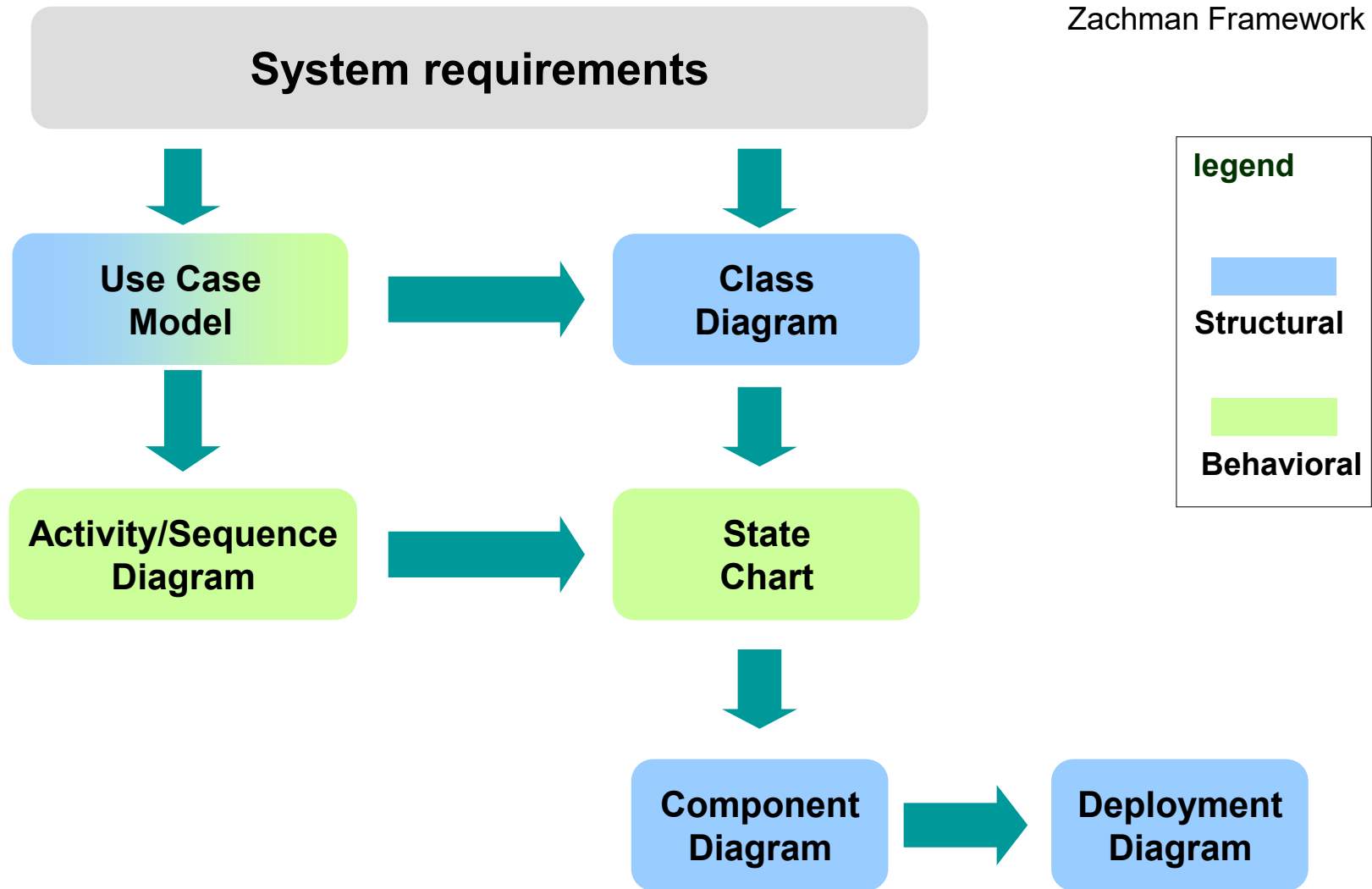


# UML 2.0 Diagrams



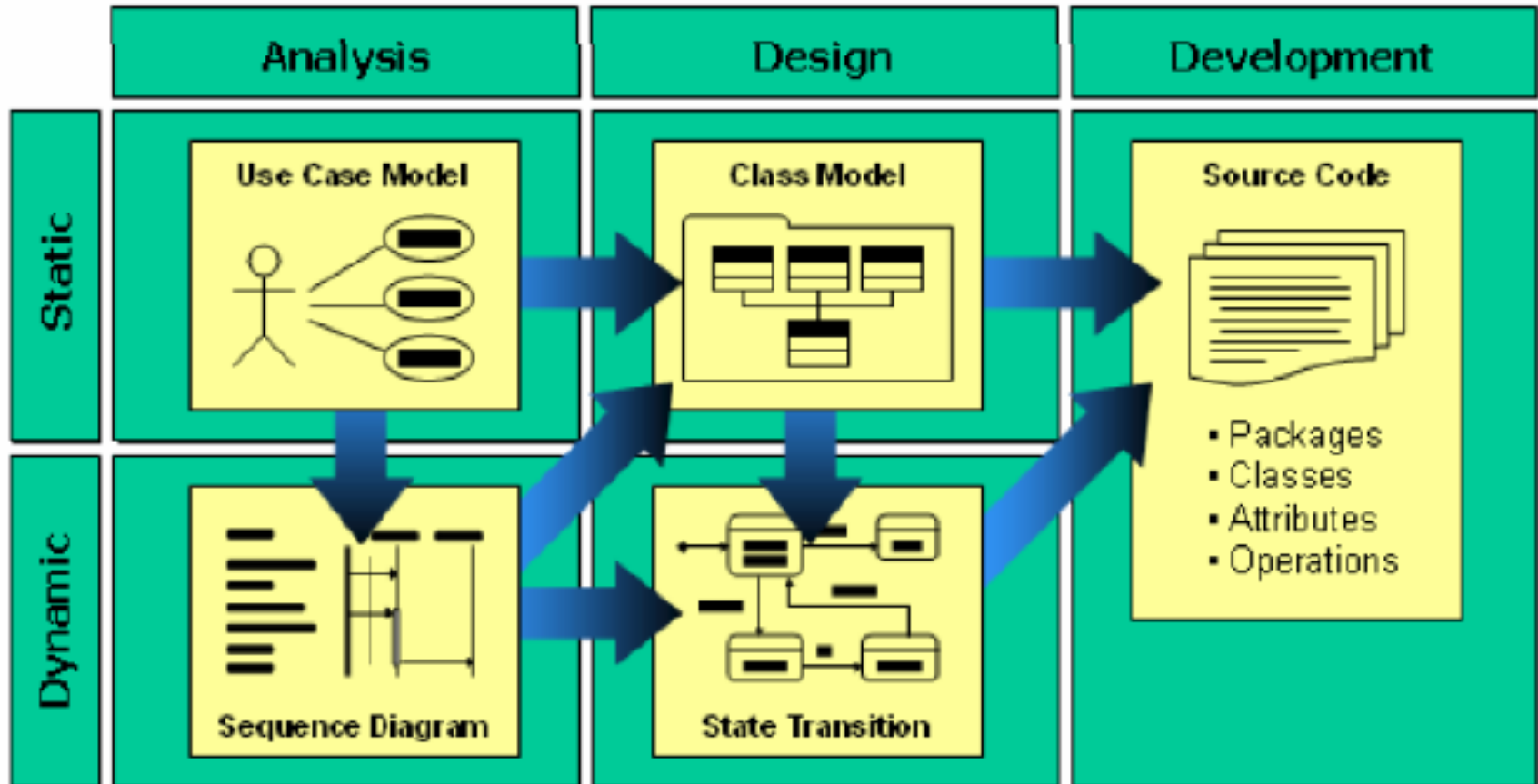
# Analysis and Design Process

Zachman Framework





# System Development

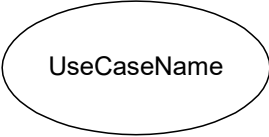
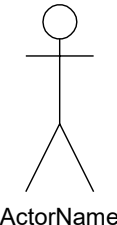
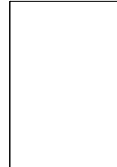


# Use case diagram




---

- ❑ Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
- ❑ Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

# Use Case Modeling: Core Elements

Construct	Description	Syntax
<b>use case</b>	Use case adalah gambaran fungsionalitas (kegunaan) dari suatu sistem (entitas lain) yang berinteraksi dengan aktor sistem.	
<b>actor</b>	Actor menggambarkan orang, system atau external entitas yang menyediakan atau menerima informasi dari system.	
<b>system boundary</b>	Kotak yang menetapkan lingkup sistem untuk use case. Semua use case di luar kotak akan dianggap di luar ruang lingkup sistem itu..	

# Use Case Modeling: Core Relationships

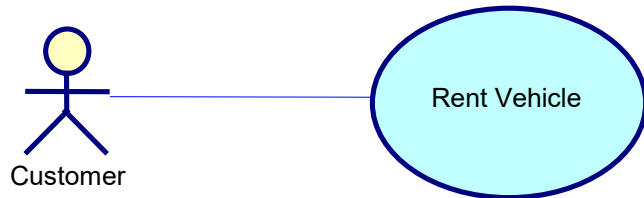
Construct	Description	Syntax
<b>association</b>	Partisipasi suatu aktor dalam use case, yaitu menghubungkan aktor dan instance dari use case berkomunikasi satu sama lain	
<b>generalization</b>	Hubungan taksonomi antara use case yang lebih umum dan use case yang lebih spesifik.	
<b>extend</b>	Hubungan dari use case ekstensi ke use case dasar, menentukan bagaimana perilaku untuk use case ekstensi dapat dimasukkan ke dalam perilaku yang ditentukan untuk use case dasar.	

# Use Case Modeling: Core Relationships (cont'd)

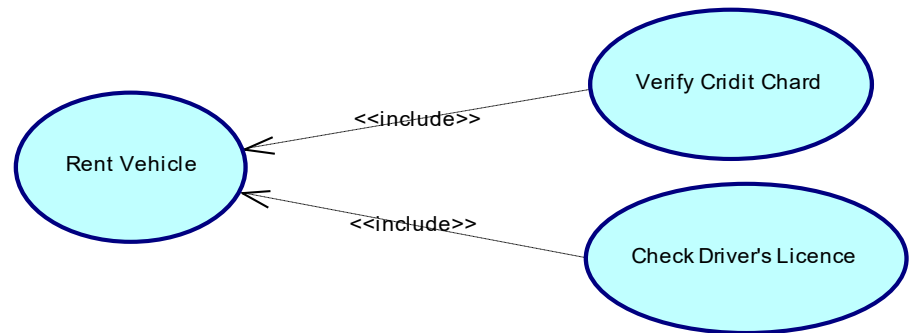
Construct	Description	Syntax
<b>include</b>	Suatu hubungan dari use case dasar ke use case inklusi, menentukan bagaimana perilaku untuk use case inklusi dimasukkan ke dalam perilaku yang ditentukan untuk use case dasar.	<code>&lt;&lt;include&gt;&gt;</code> ----->



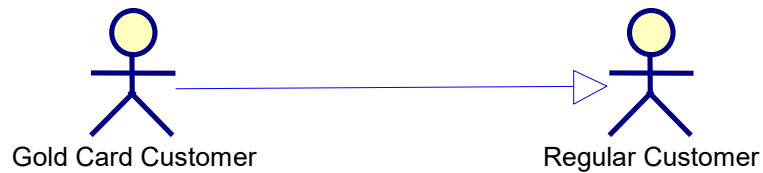
Contoh: kegiatan pasien membuat janji



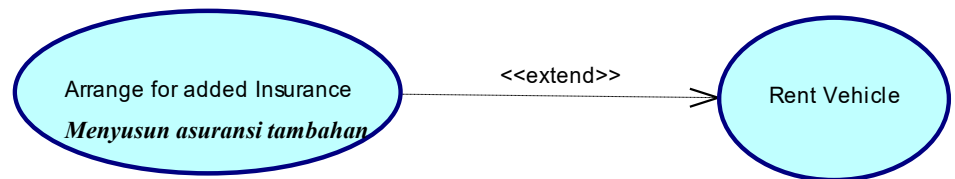
**Communication Relationships**



**Include Relationships**

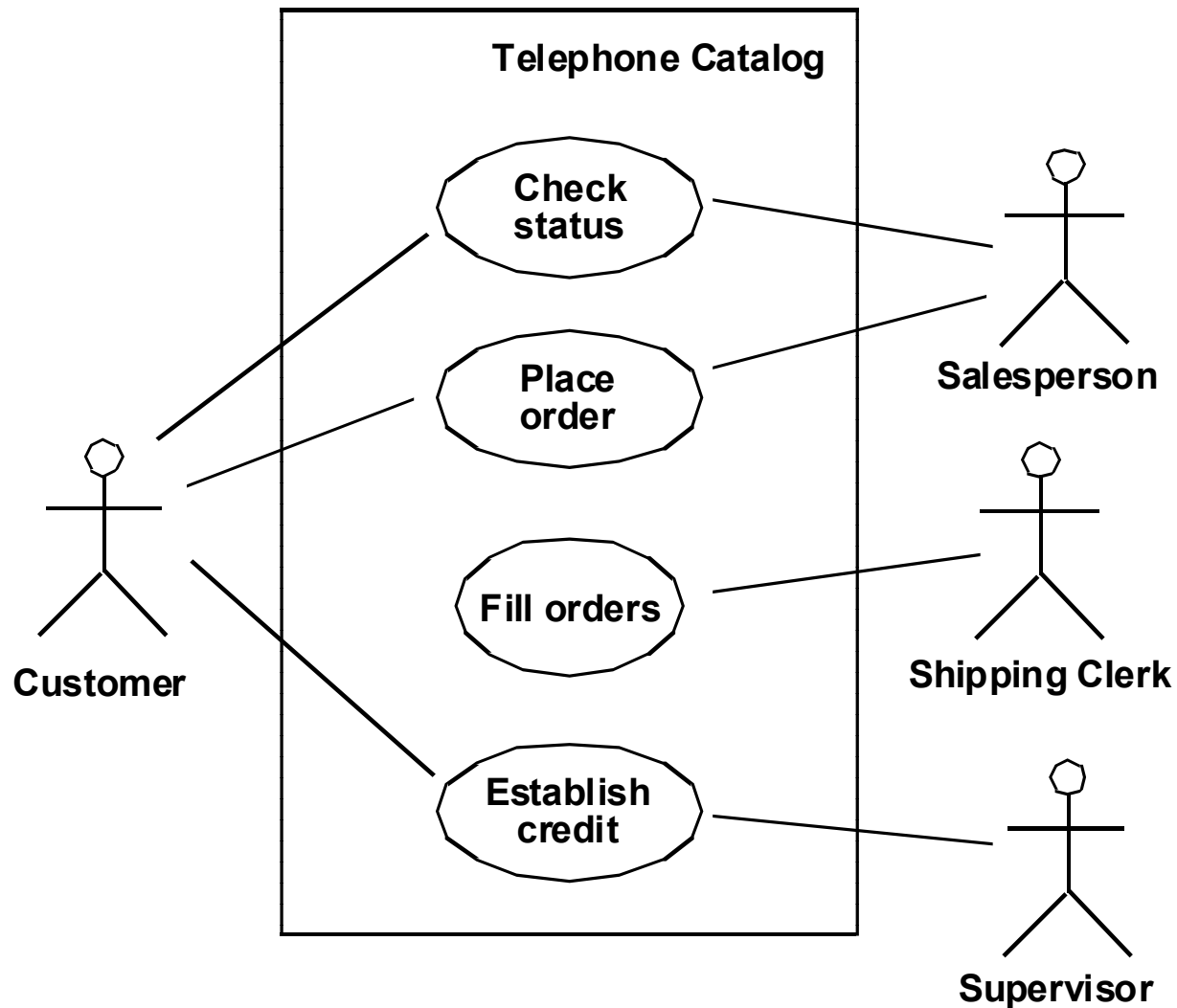


**Generalizes Relationships**



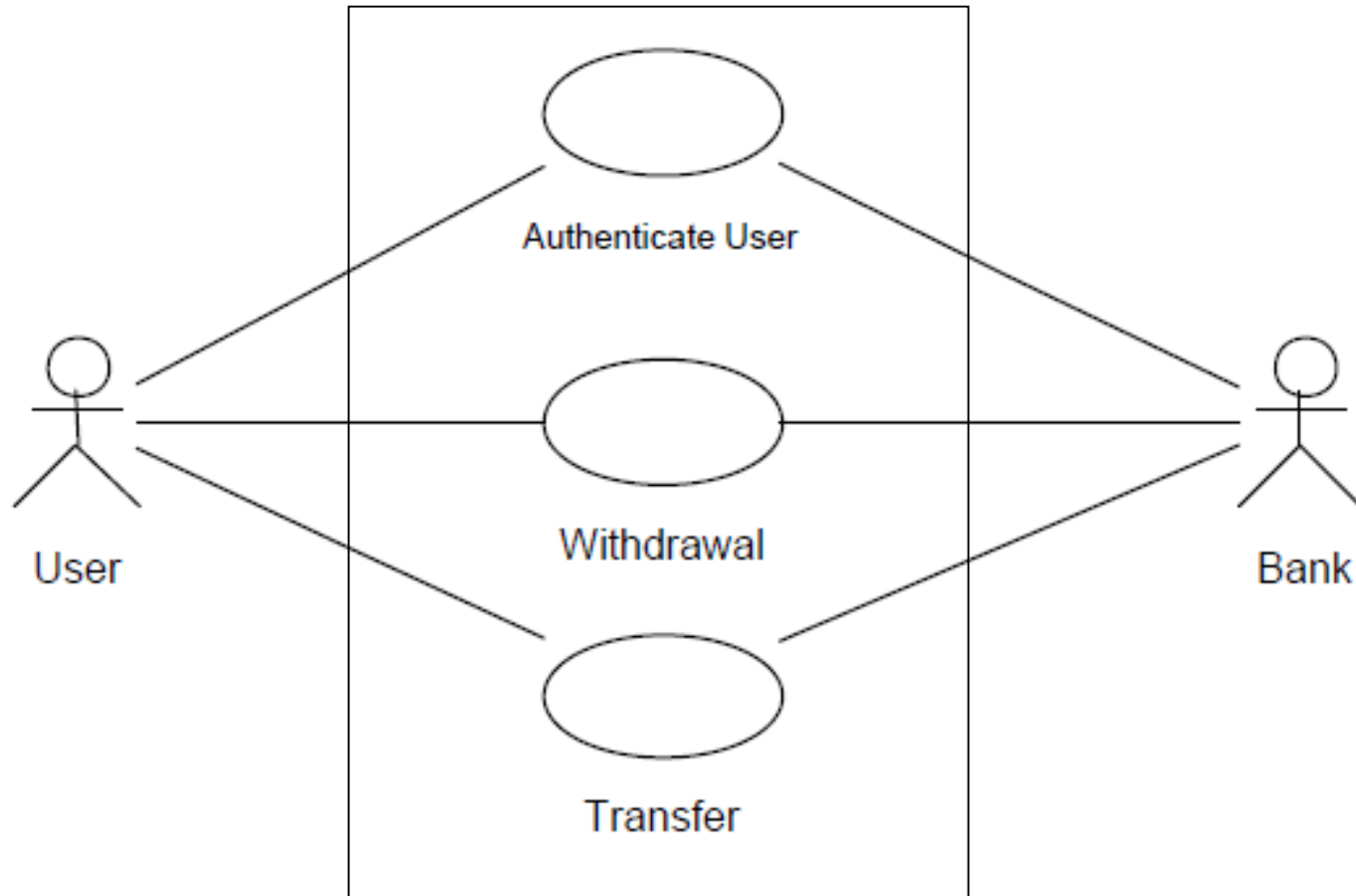
**Extends Relationships**

# Contoh: Use Case Diagram



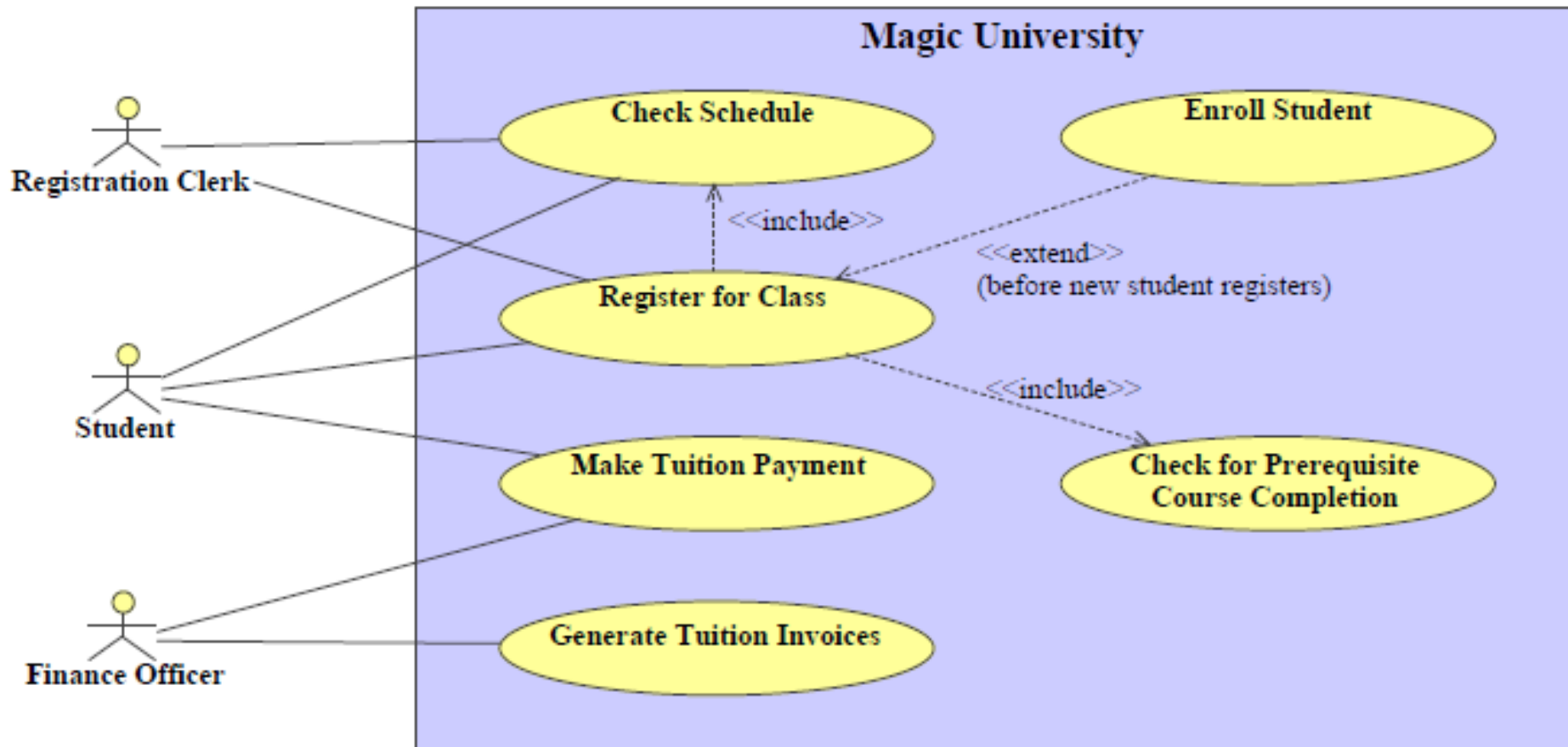
# Contoh: Use Case Diagram

---

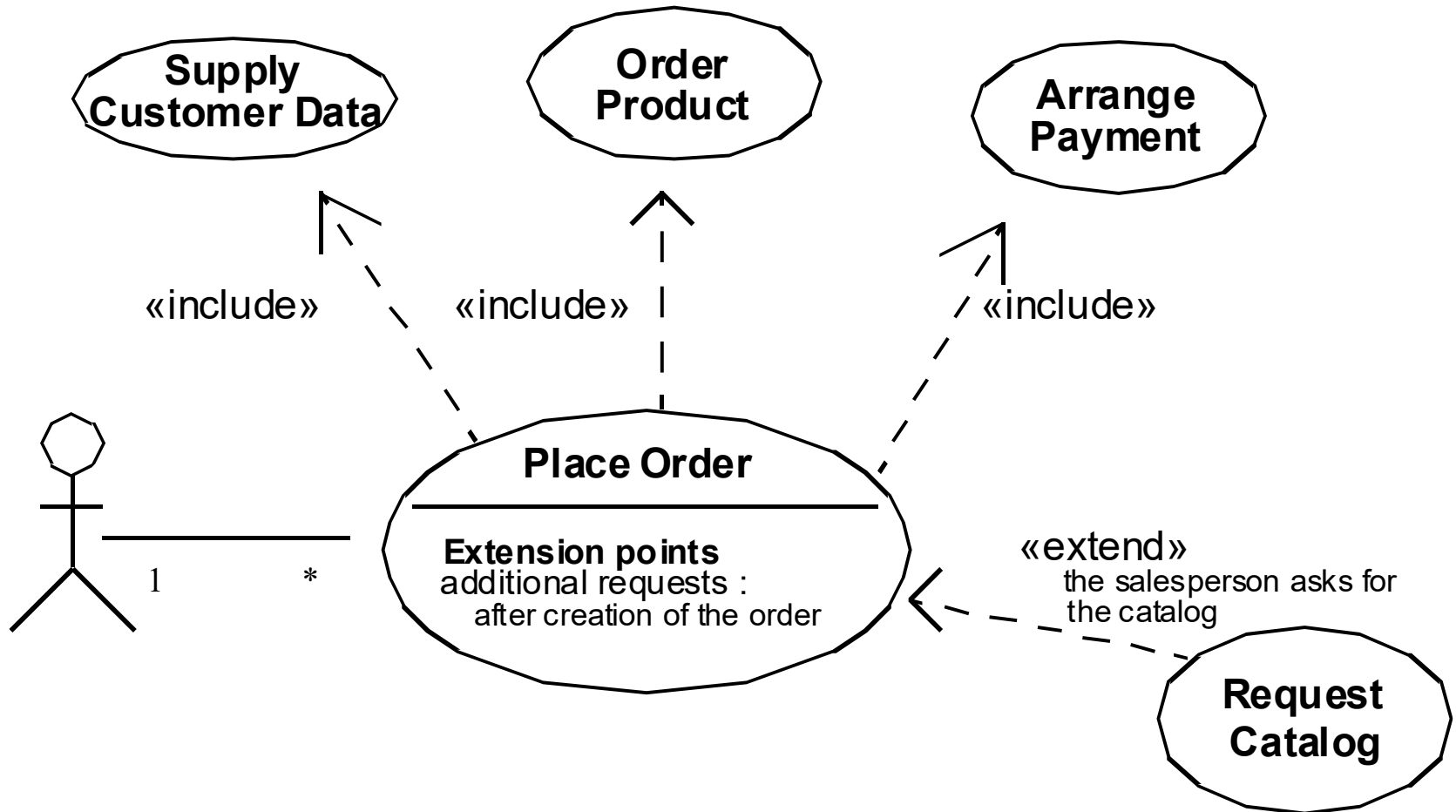




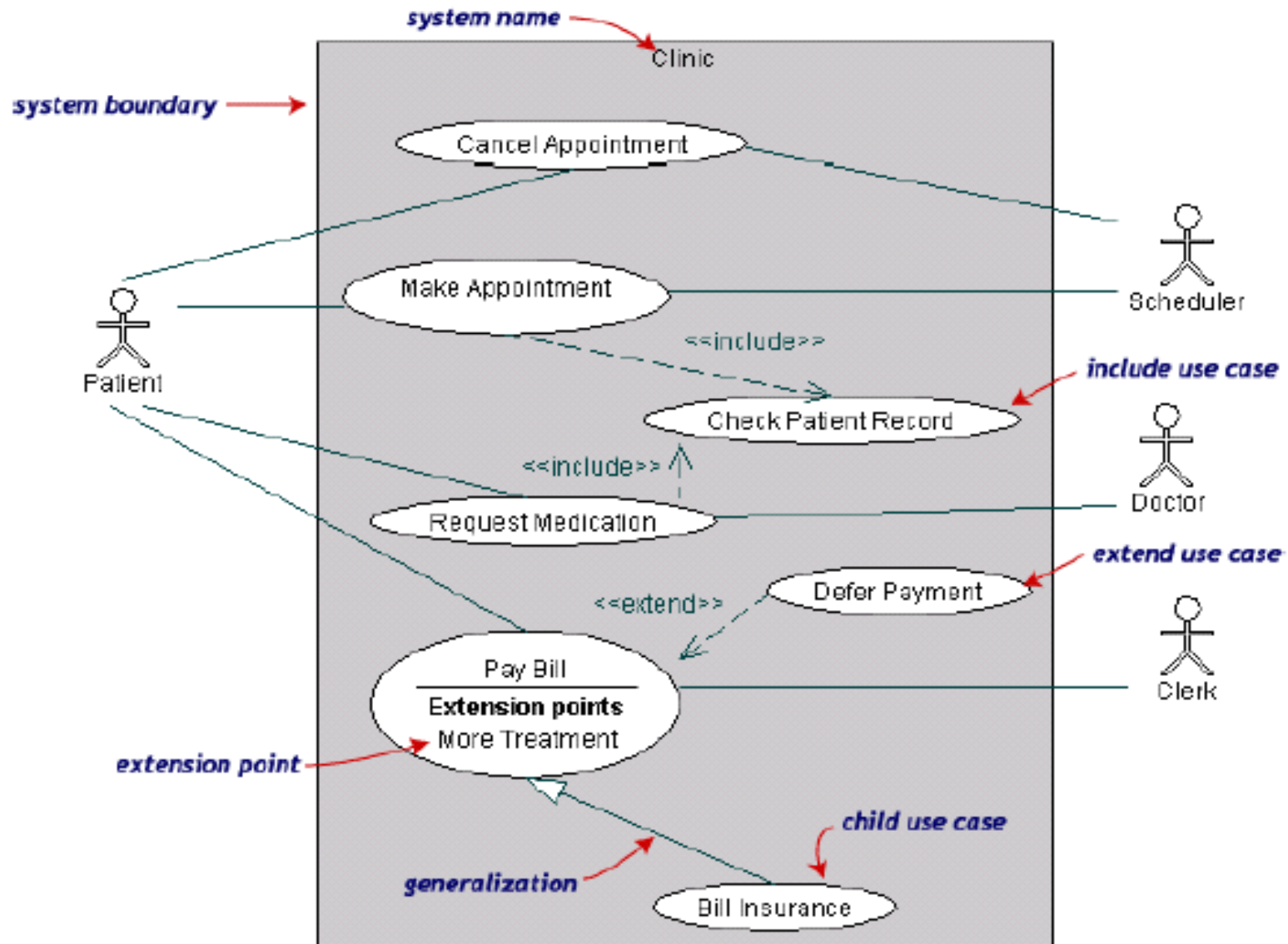
# Contoh: Use Case Diagram



# Use Case Relationships



# Contoh: Use Case Diagram



# Class Diagram

---

- ❑ Class adalah 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.

# Class Diagram

---

Class memiliki tiga area pokok :

1. Nama (dan stereotype)
2. Atribut
3. Metoda

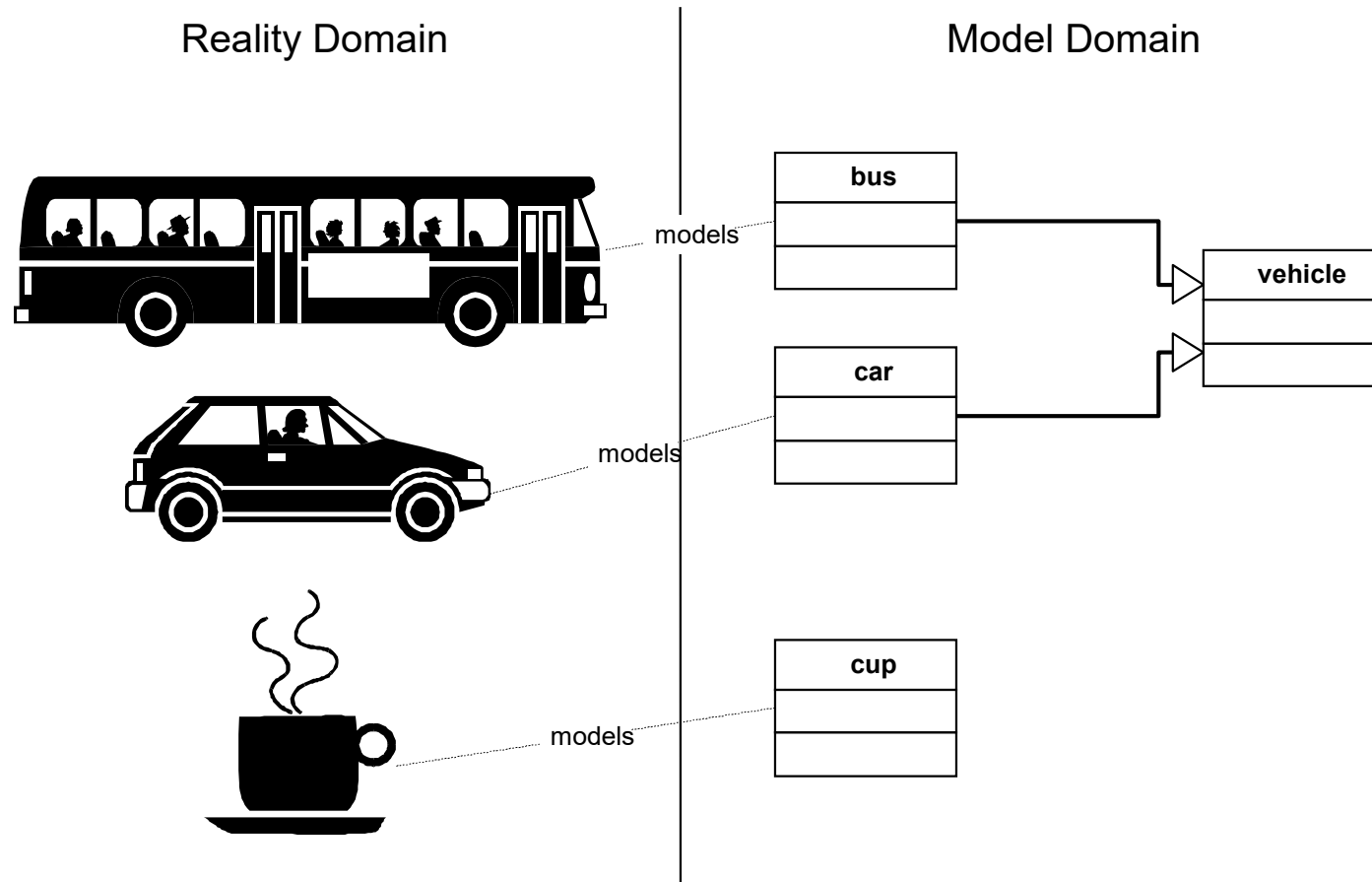
Atribut dan metoda dapat memiliki salah satu sifat berikut :

Private, tidak dapat dipanggil dari luar class yang bersangkutan

Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya

# Object-Oriented Approach

Objects are abstractions of real-world or system entities



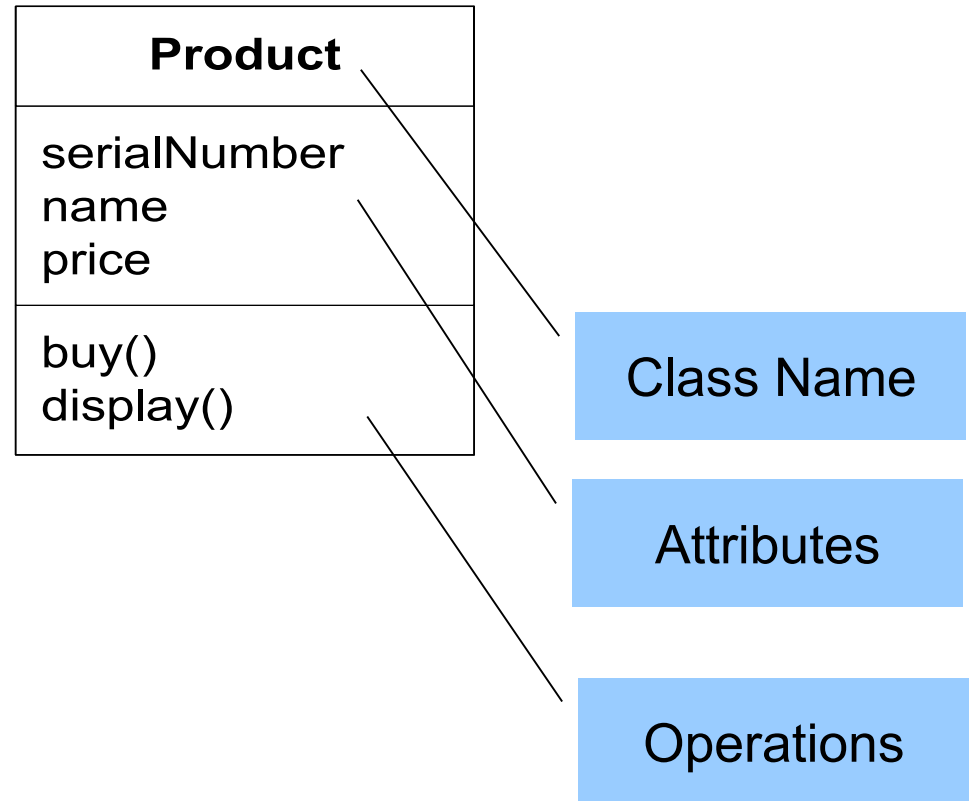
# Object-Oriented Approach

---

- ❑ **Object** adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasanbatasan dan pengertian yang tepat.
- ❑ *Object* bisa mewakili sesuatu yang nyata seperti komputer, mobil dll.
- ❑ *Object* juga dapat berupa konsep seperti proses kimia, transaksi bank, permintaan pembelian, dll.
- ❑ Setiap *object* dalam sistem memiliki tiga karakteristik yaitu *State* (status), *Behaviour* (sifat) dan *Indentity* (identitas).

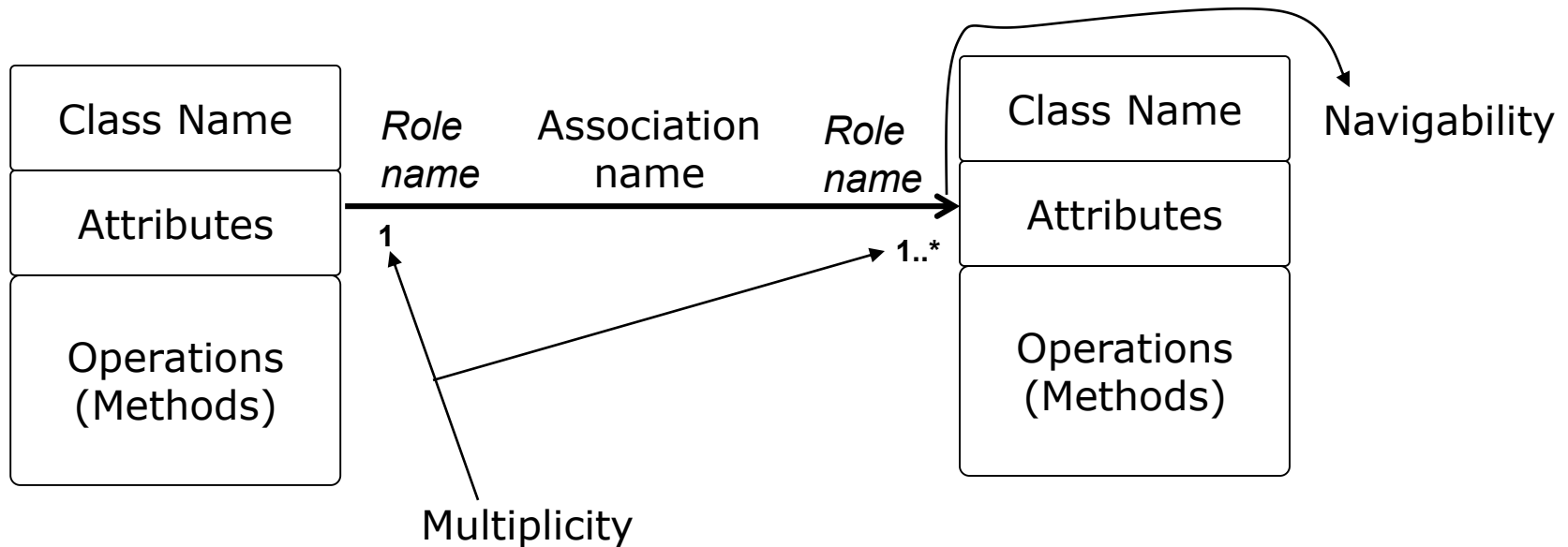
# Classes

Class merepresentasikan sesuatu yang ditangani oleh sistem





# Class Diagram Format and association:



## Multiplicity Notation

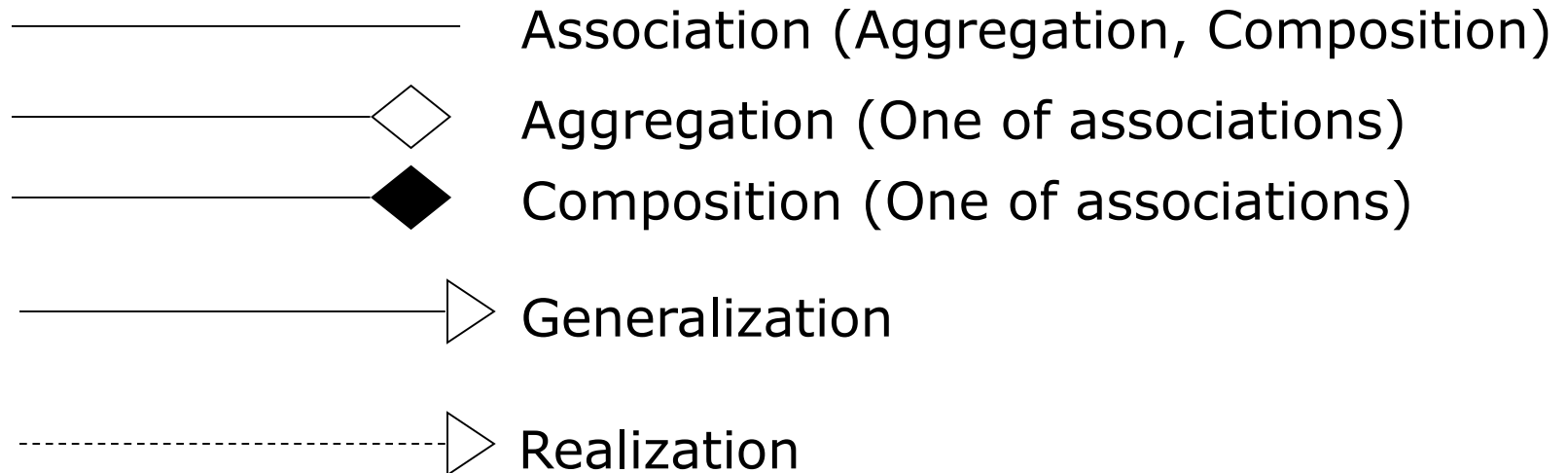
1	: One and only one
0..*	: None or more
1..*	: One or more
0..1	: None or one

# Relationships of Class

---

There three types of relationship :

1. Is-a (Generalization, Realization: Inheritance)
2. Has-a (Association)
3. Others (Association , Dependency)



# Multiplicity of Class

*Multiplicity* dari suatu titik *association* adalah angka kemungkinan bagian dari hubungan kelas dengan single *instance* (bagian) pada titik yang lain. *Multiplicity* berupa single number (angka tunggal) atau range number (angka batasan). Pada contoh, hanya bisa satu 'Customer' untuk setiap 'Order', tapi satu 'Customer' hanya bisa memiliki beberapa 'Order'.

Tabel di bawah mengenai *multiplicity* yang sering digunakan :

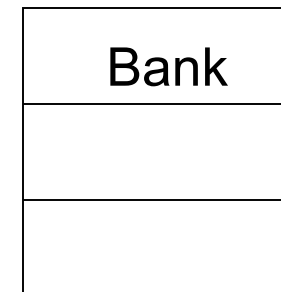
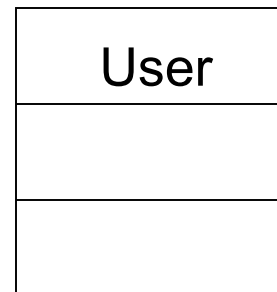
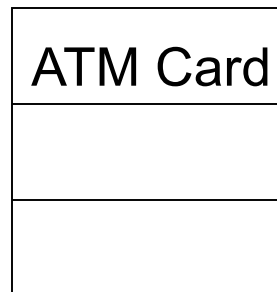
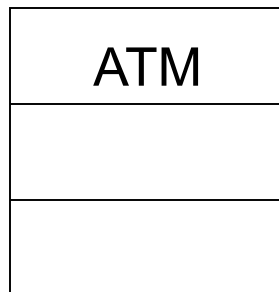
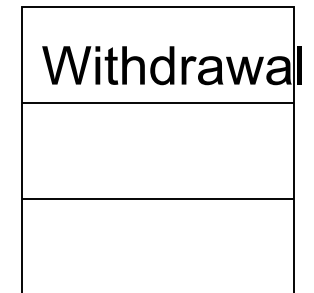
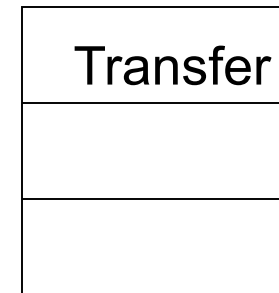
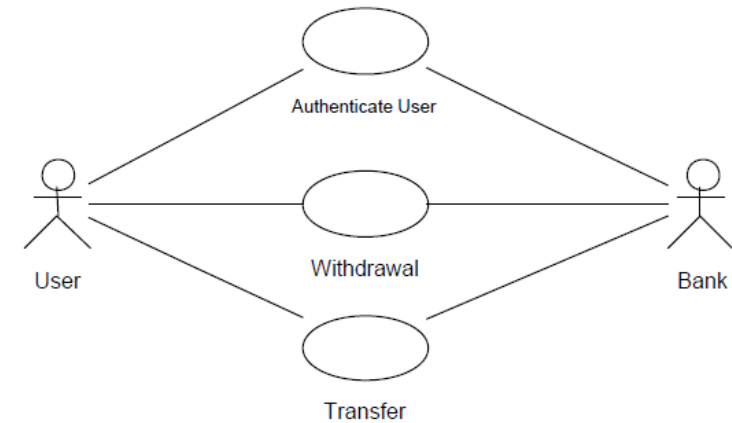
Tabel *Multiplicity*

Multiplicities	Artinya
0..1	Nol atau satu bagian. Notasi $n..m$ menerangkan $n$ sampai $m$ bagian.
0..* or *	Tak hingga pada jangkauan bagian (termasuk kosong).
1	Tepat satu bagian
1..*	Sedikitnya hanya satu bagian

# Pembuatan Class

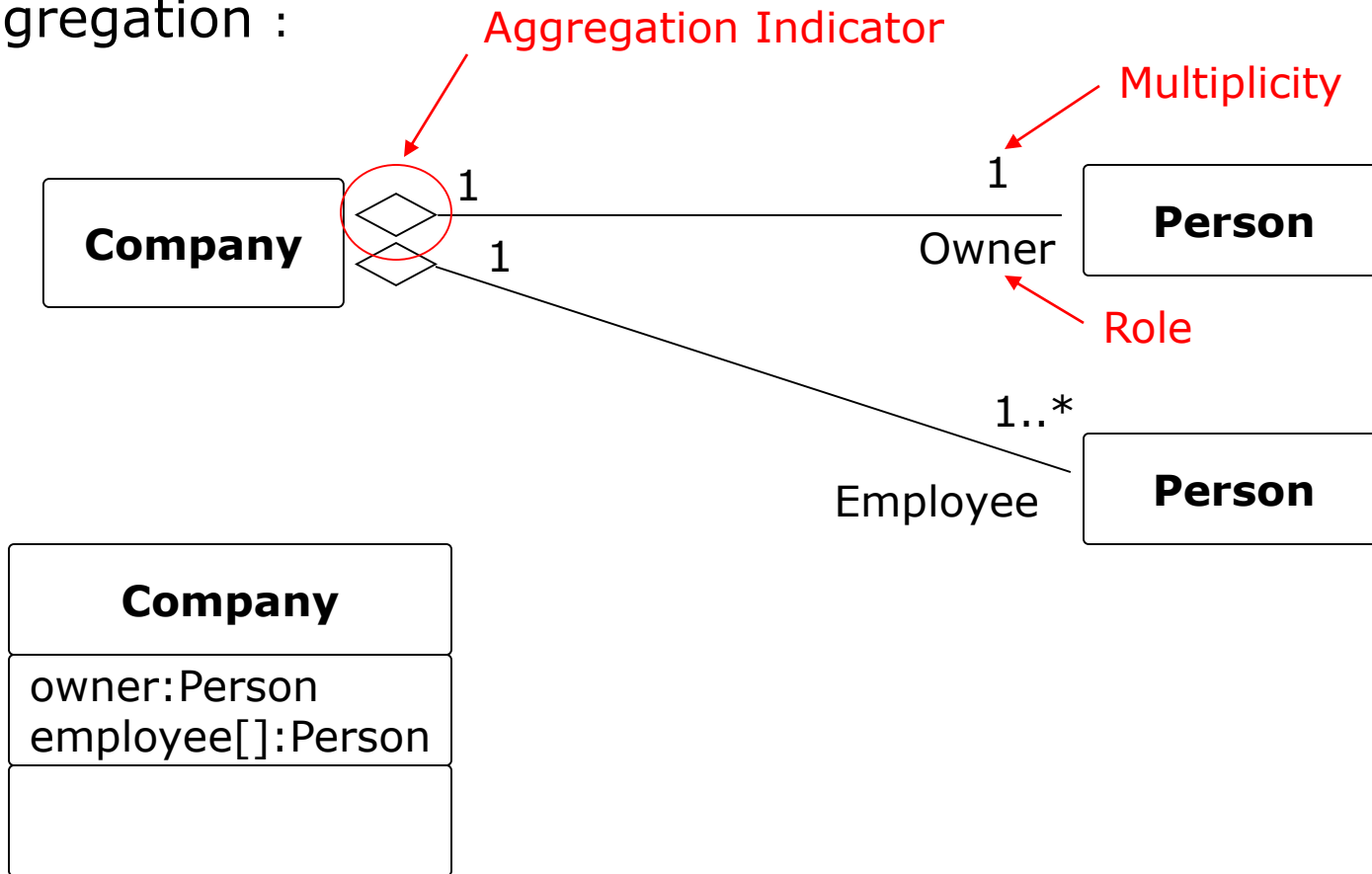
## Candidate Class

No	Kategori Object	Nama Object
1.	Object Fisik	ATM (Mesin), ATM card
2.	Transaksi	Withdrawal, Transfer
3.	Butir yang terlibat pada transaksi	.....
4.	Peran	User(Pemegang ATMCARD) Bank
5.	Piranti	ATM Komputer
6.	Proses	Withdrawal Update
7.	Katalog	Daftar Account



# Contoh Class Diagram

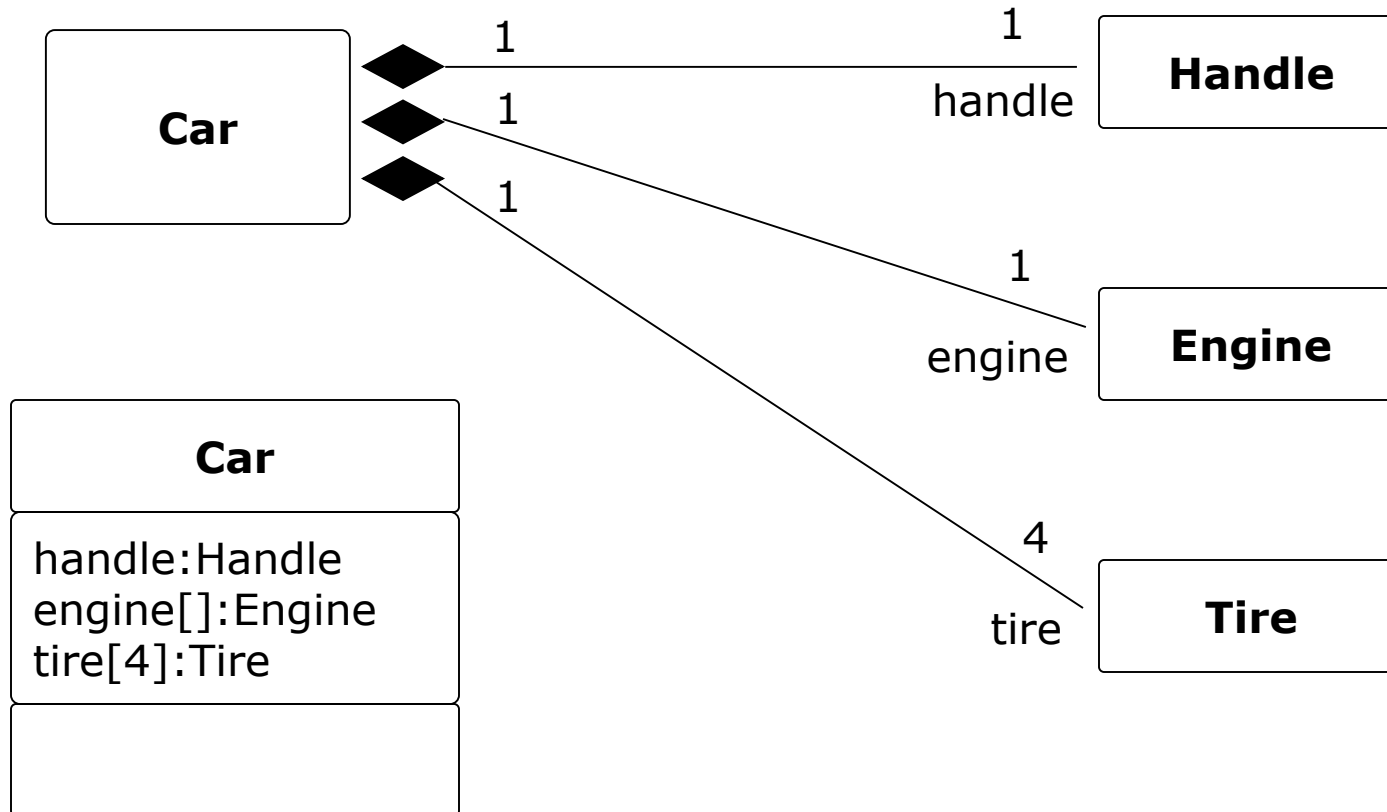
Aggregation :



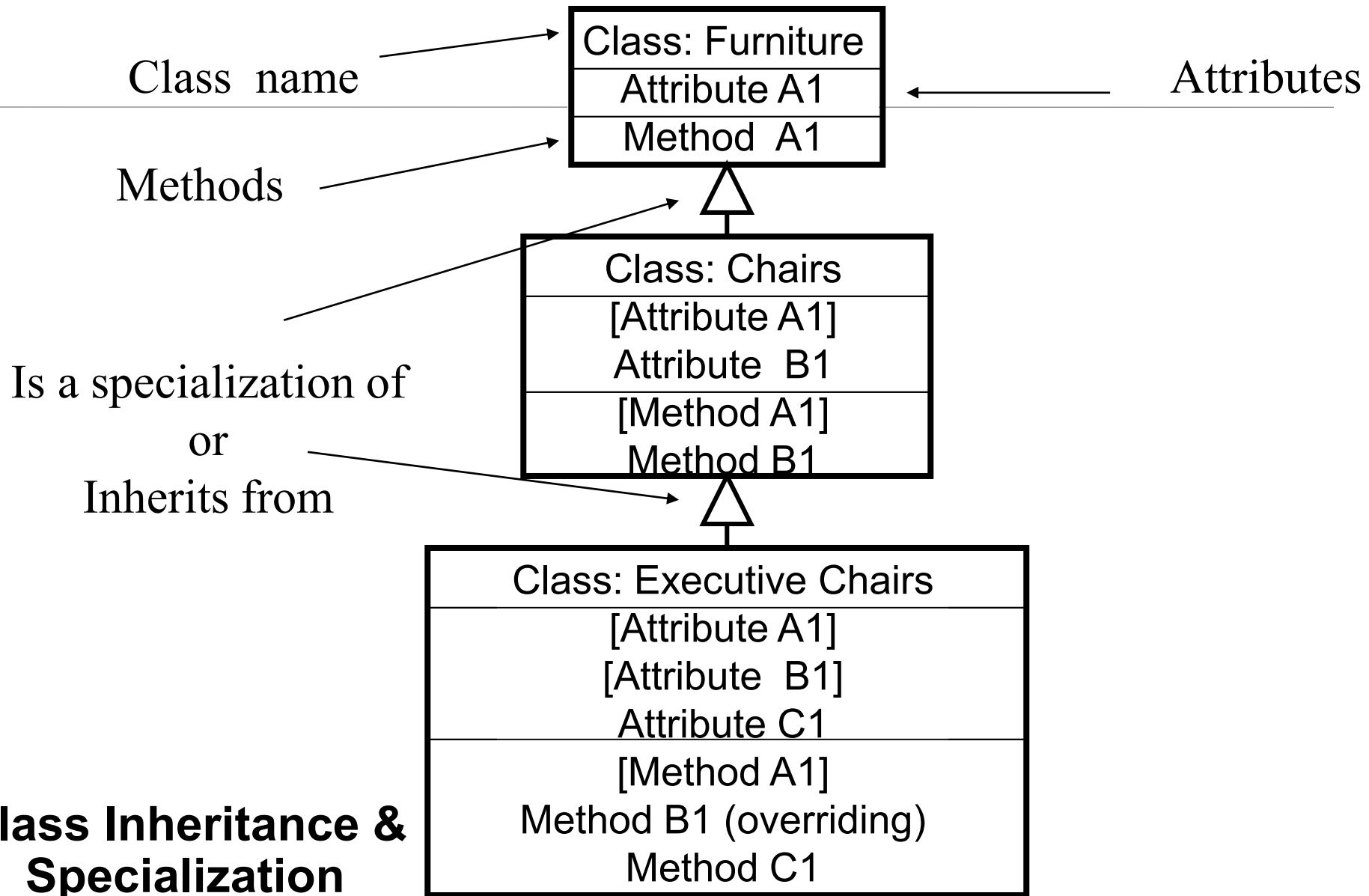
Detail of the class

# Contoh Class Diagram

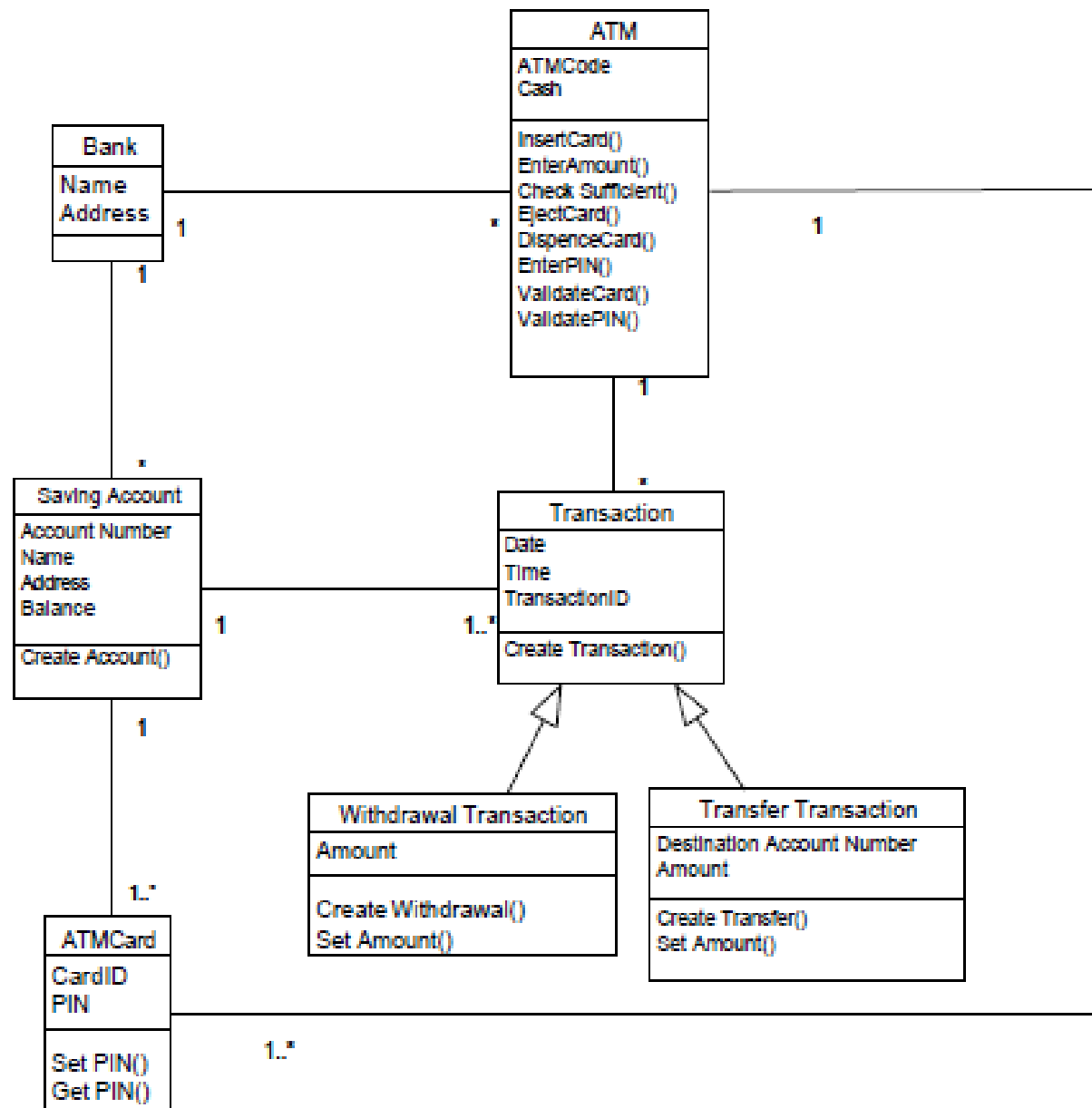
Composition :



Detail of the class

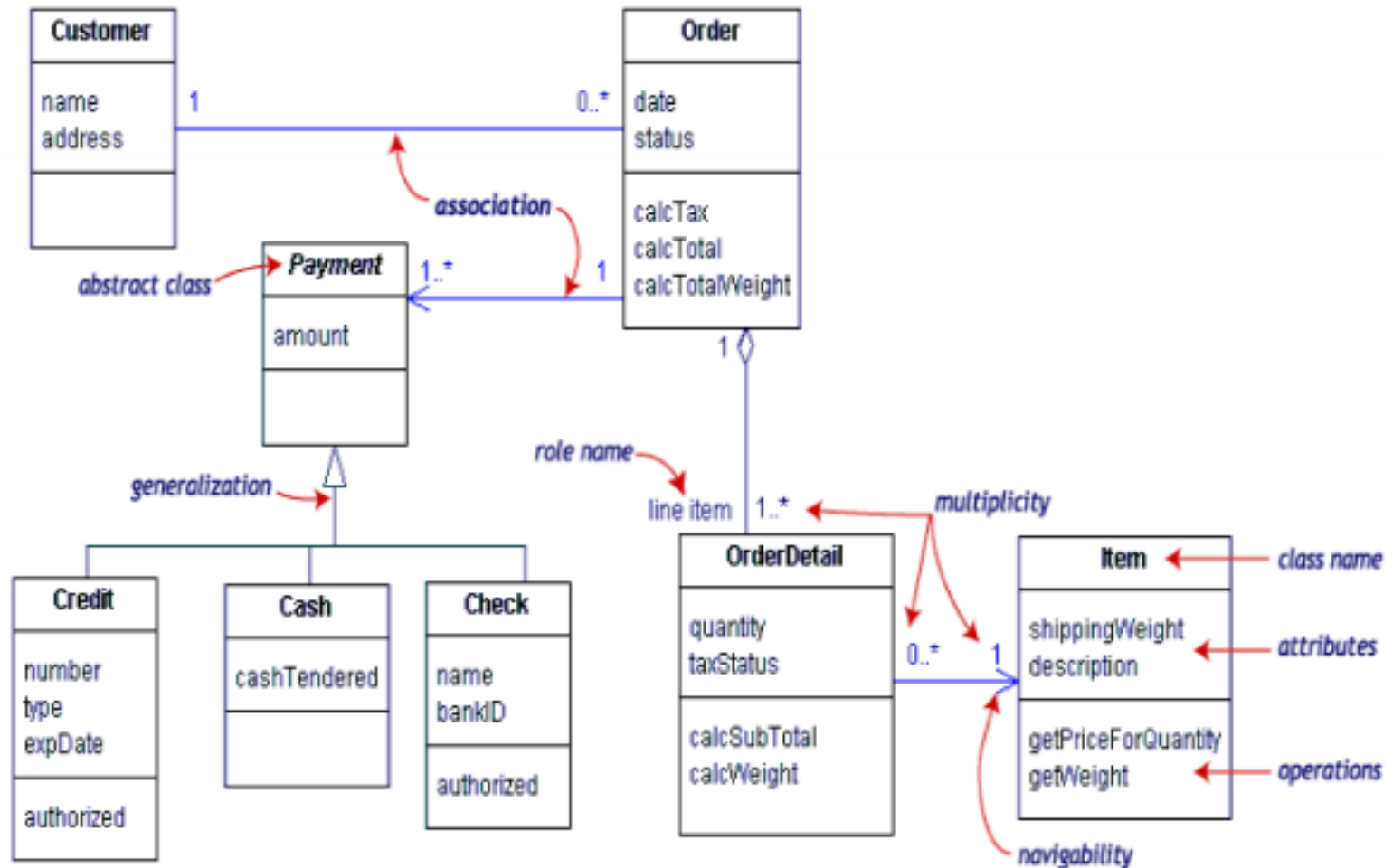


# Contoh Class Diagram (Studi Kasus ATM)





# Contoh Class Diagram (Pembelian Barang)



Contoh Diagram Class transaksi Pembelian barang.

# StateChart Diagram

---

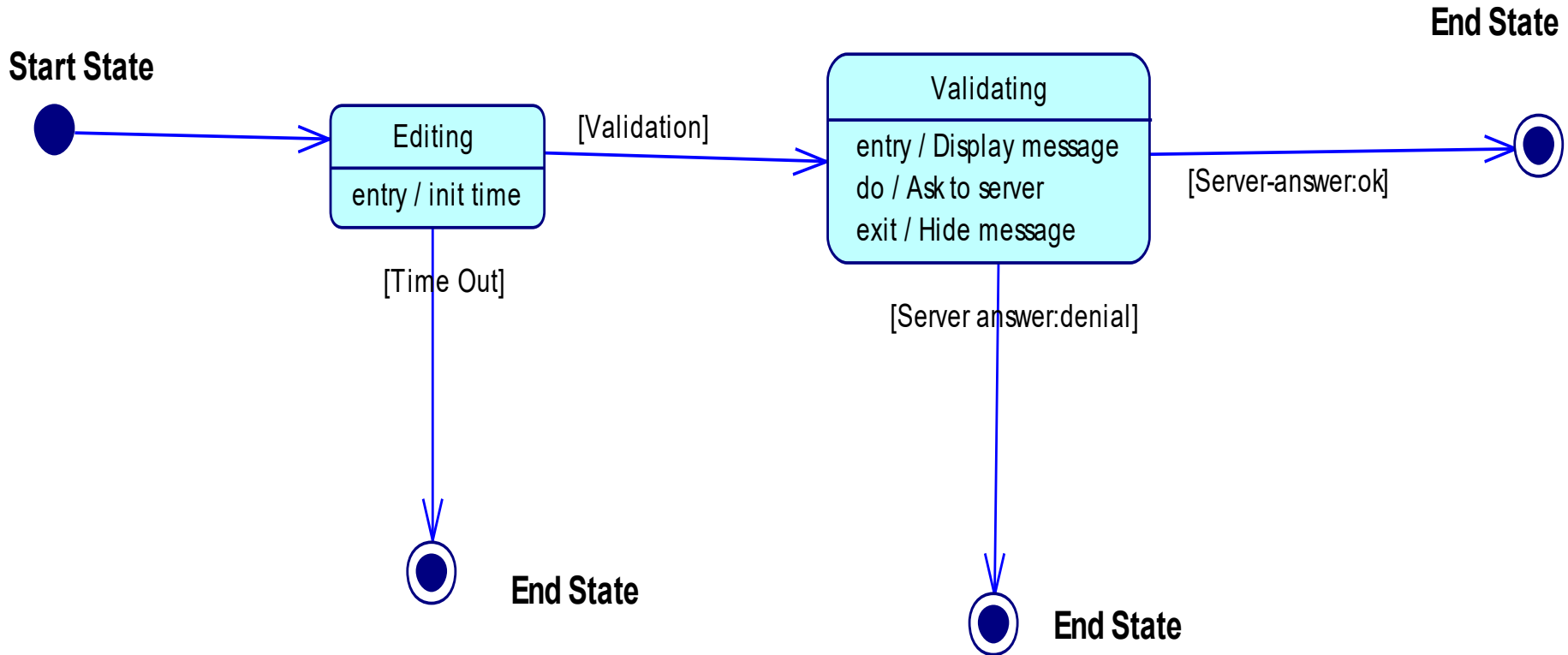
- ❑ Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya)
- ❑ Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

# StateChart Diagram

---

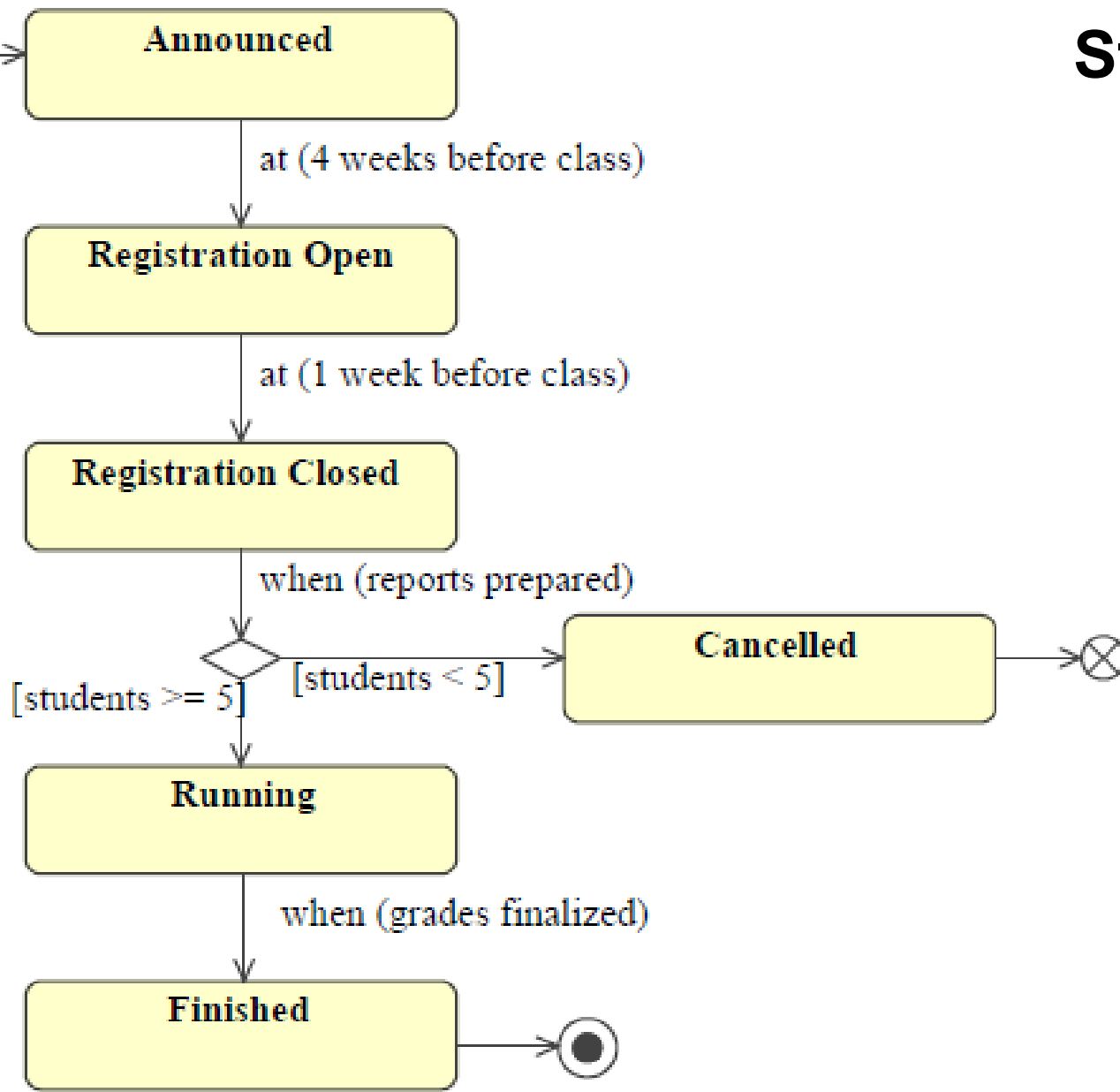
- ❑ Dalam UML, *state* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu.
- ❑ Transisi antar *state* umumnya memiliki kondisi *guard* yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku.
- ❑ *Action* yang dilakukan sebagai akibat dari *event* tertentu dituliskan dengan diawali garis miring.
- ❑ Titik awal dan akhir digambarkan berbentuk lingkaran berwarna penuh dan berwarna setengah.

# State Diagram :Authentication Process



# State Diagram

## Class Open Process

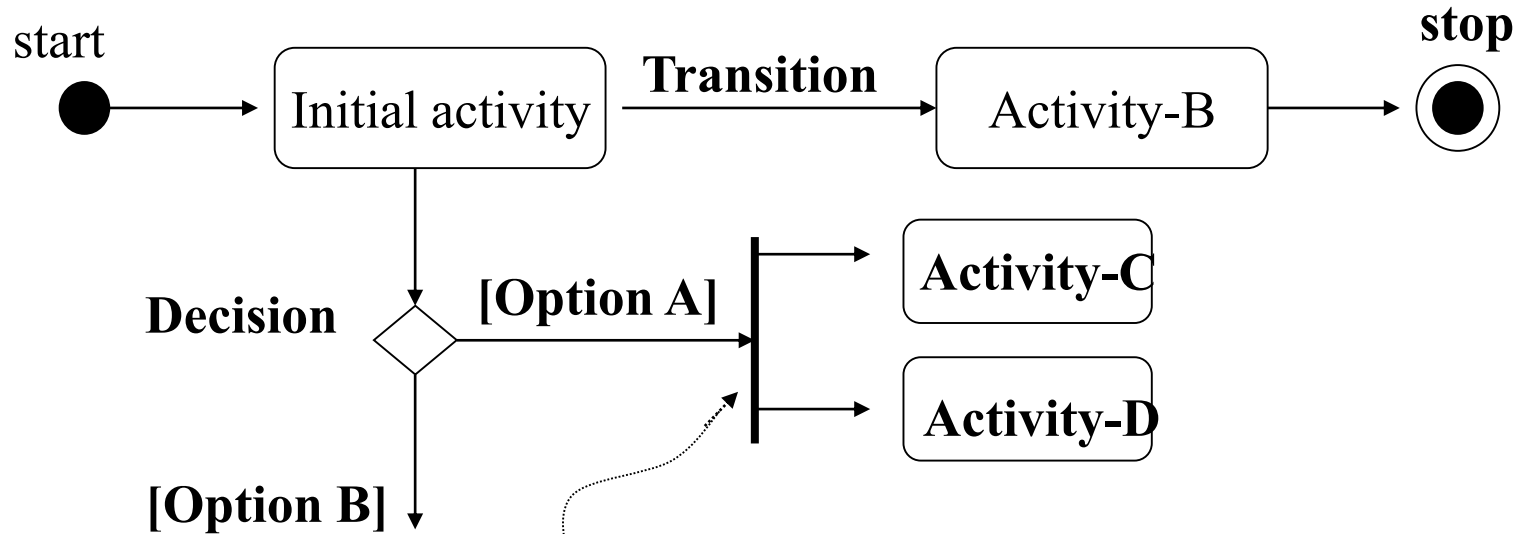


# Activity Diagram

---

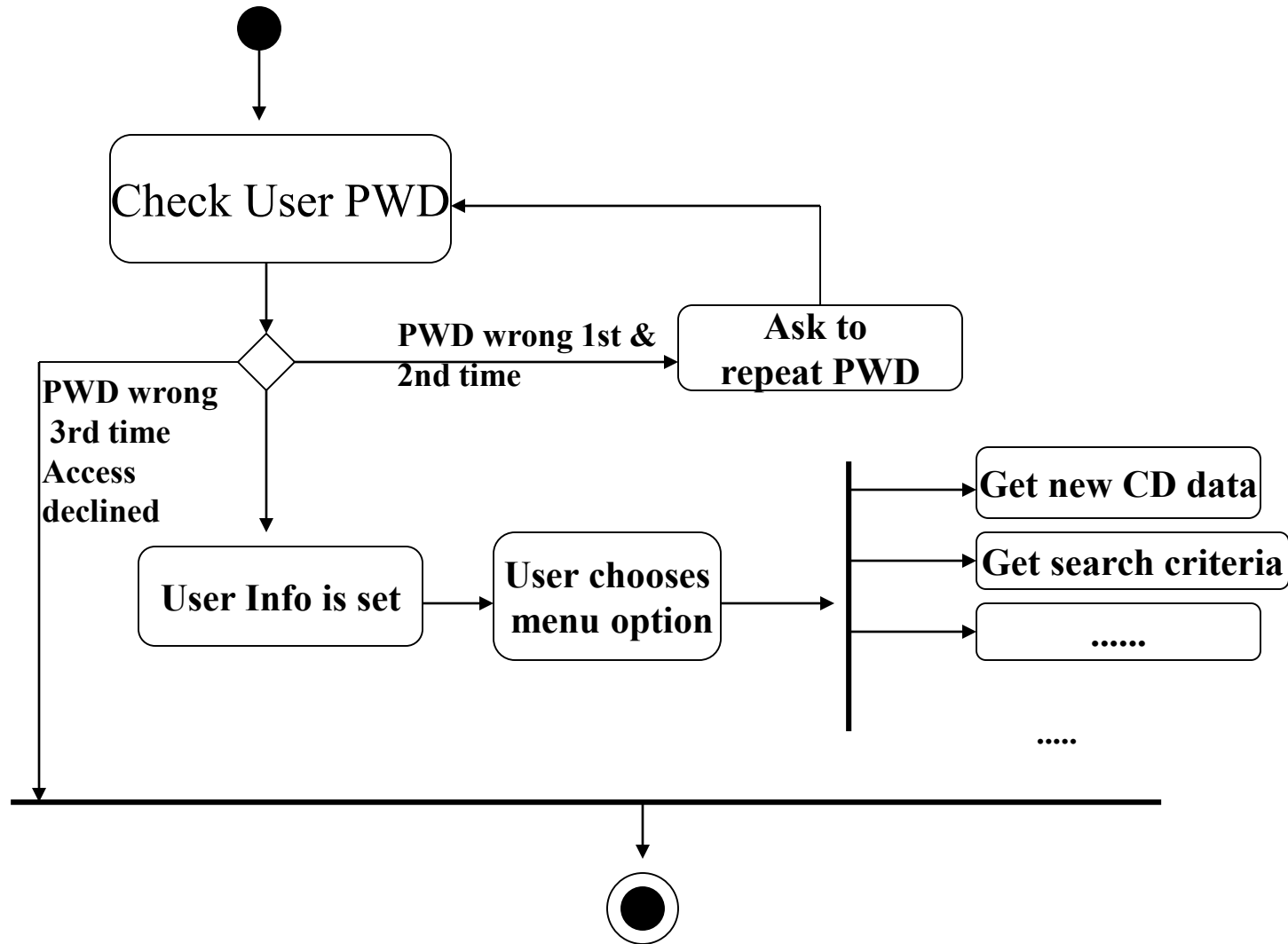
- ❑ Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.
- ❑ Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.
- ❑ Activity diagram merupakan state diagram khusus, di mana sebagian besar state adalah action dan sebagian besar transisi di-trigger oleh selesainya state sebelumnya (internal processing).
- ❑ Oleh karena itu activity diagram tidak menggambarkan behaviour internal sebuah sistem (dan interaksi antar subsistem) secara eksak, tetapi lebih menggambarkan proses-proses dan jalur-jalur aktivitas dari level atas secara umum.

# Activity Diagrams Format



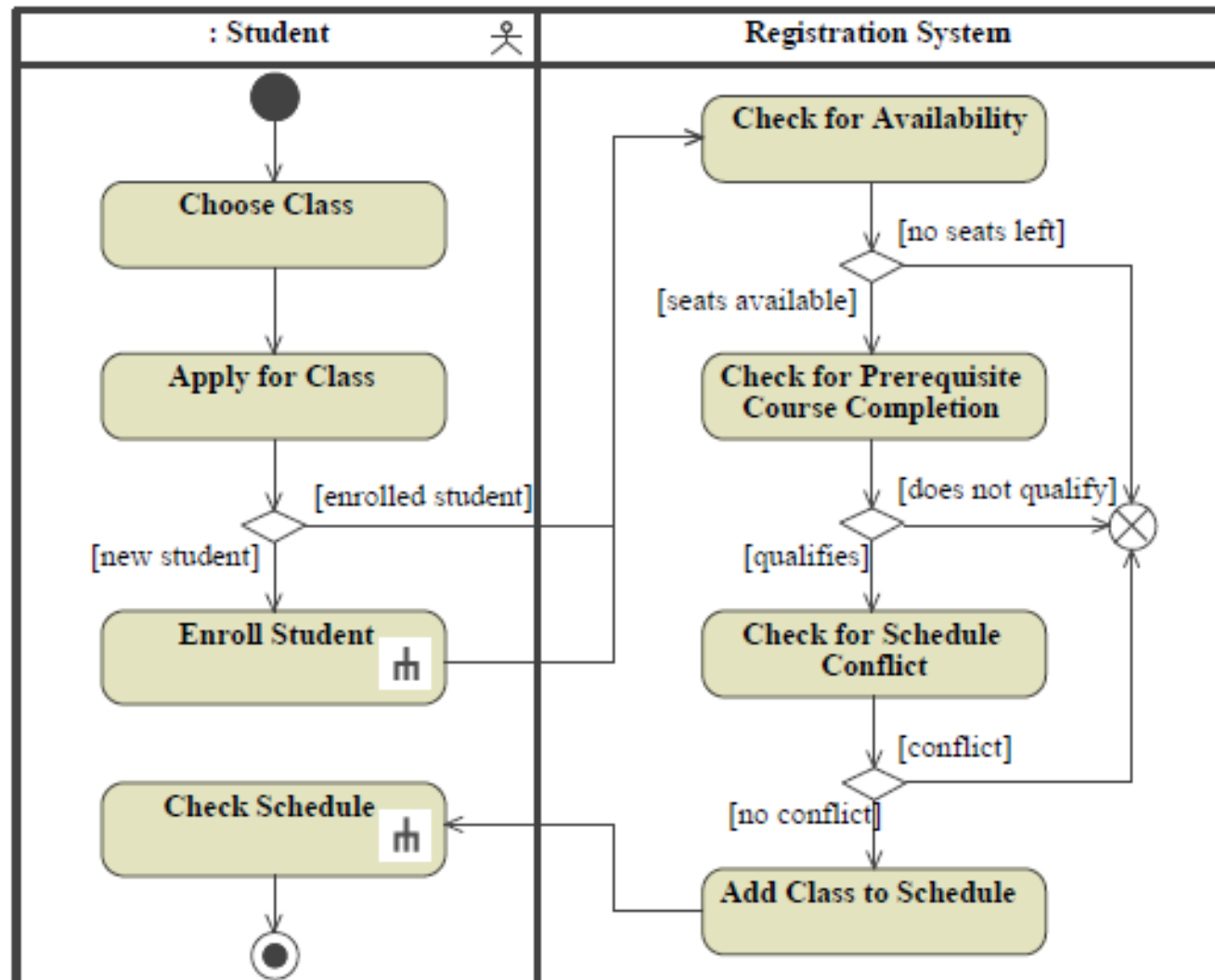
The bar shows that one activity leads to several that occur in parallel or in an unpredictable order.

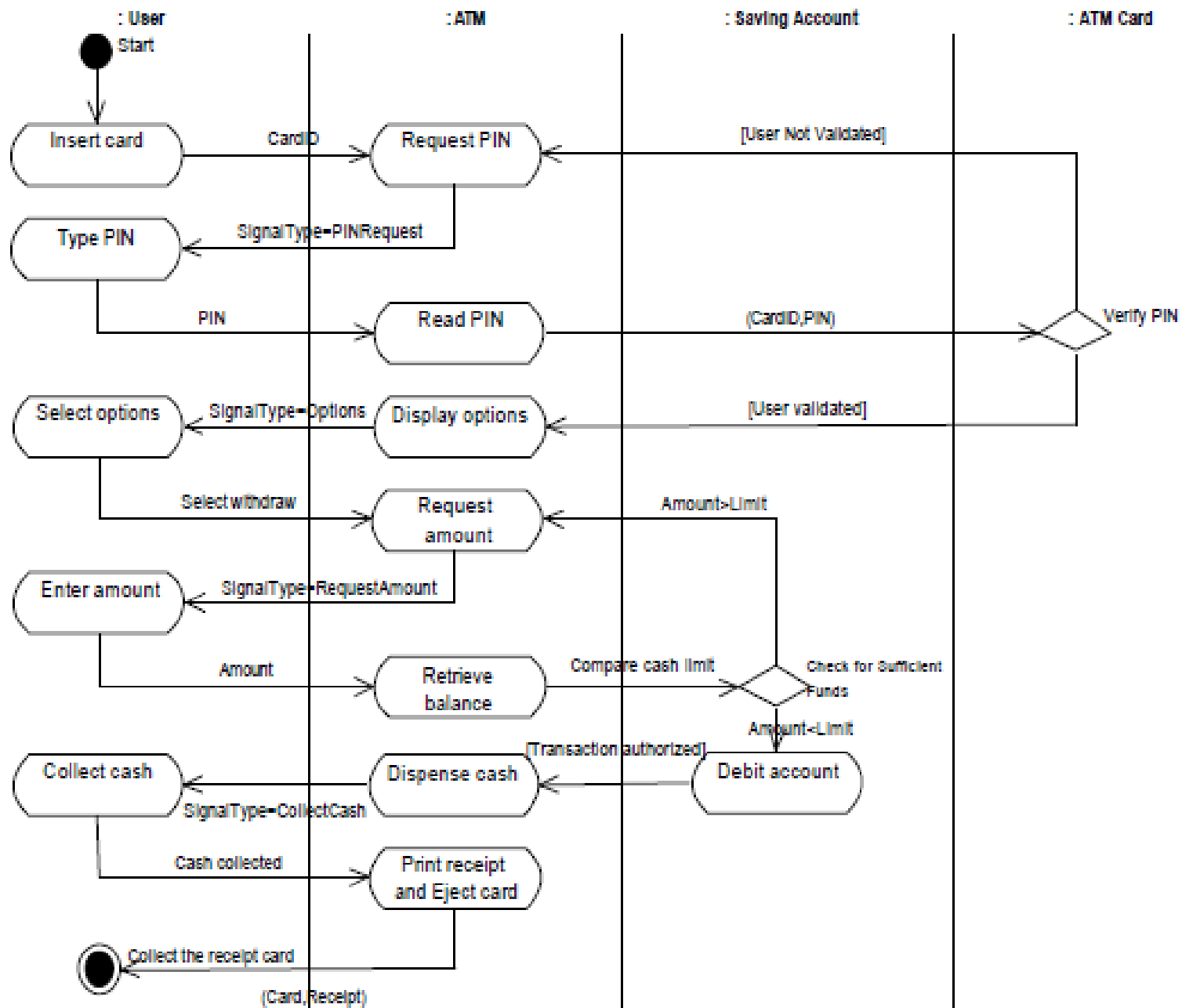
# Activity Diagrams Example

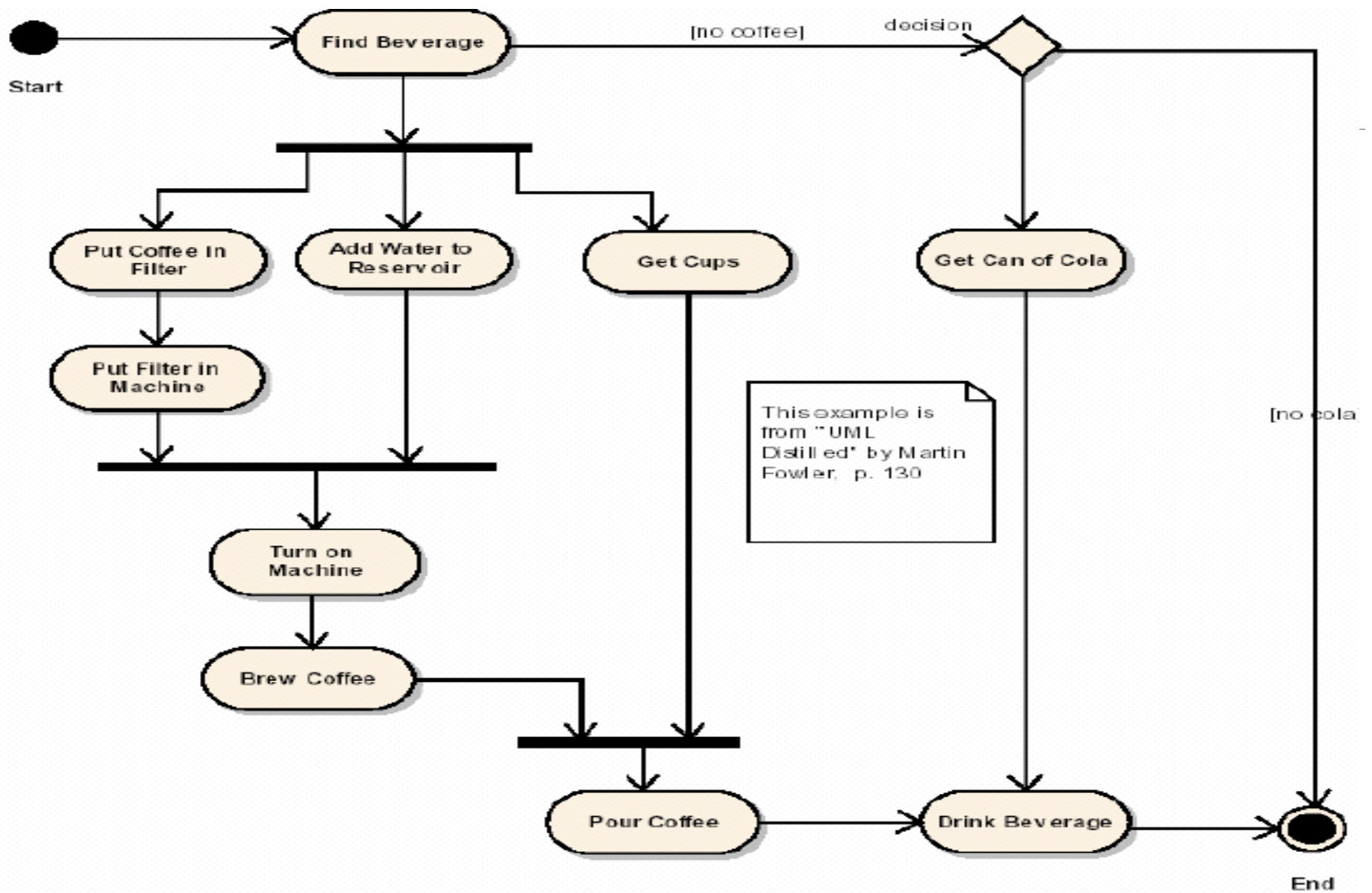




# Activity Diagrams Example






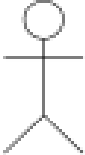



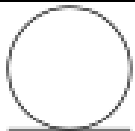


# Sequence Diagram

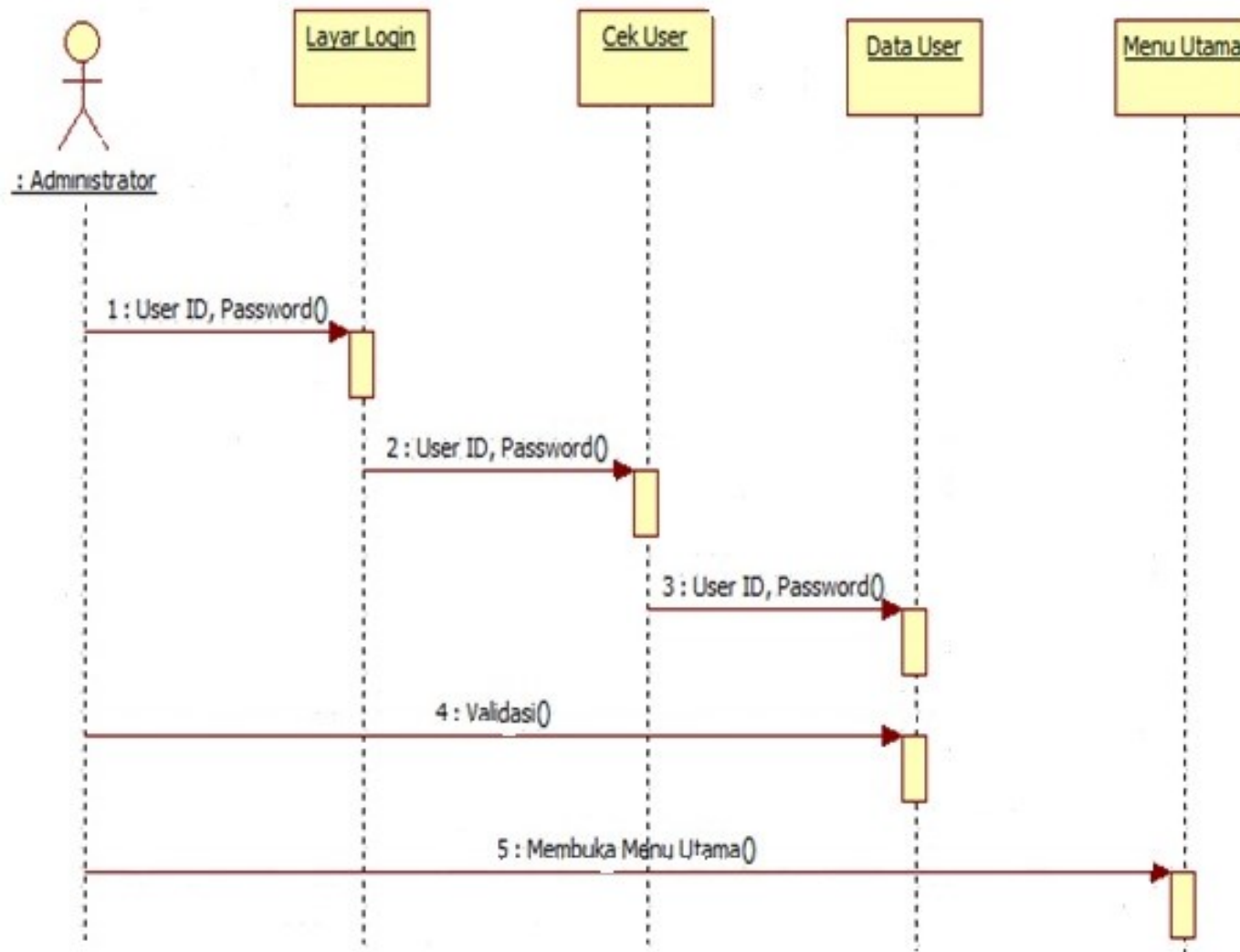
---

- ❑ Sequence diagram menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yang digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).
- ❑ Komponen dasar sequence diagram :
  - ❖ Object - adalah komponen berbentuk kotak yang mewakili sebuah class atau object. Mereka mendemonstrasikan bagaimana sebuah object berperilaku pada sebuah system.
  - ❖ Activation boxes - adalah komponen yang berbentuk persegi panjang yang menggambarkan waktu yang diperlukan sebuah object untuk menyelesaikan tugas. Lebih lama waktu yang diperlukan, maka activation boxes akan lebih panjang.
  - ❖ Actors - adalah komponen yang berbentuk stick figure. Komponen yang mewakili seorang pengguna yang berinteraksi dengan system.
  - ❖ Lifeline - adalah komponen yang berbentuk garis putus - putus. Lifeline biasanya memuat kotak yang berisi nama dari sebuah object. Berfungsi menggambarkan aktifitas dari object.

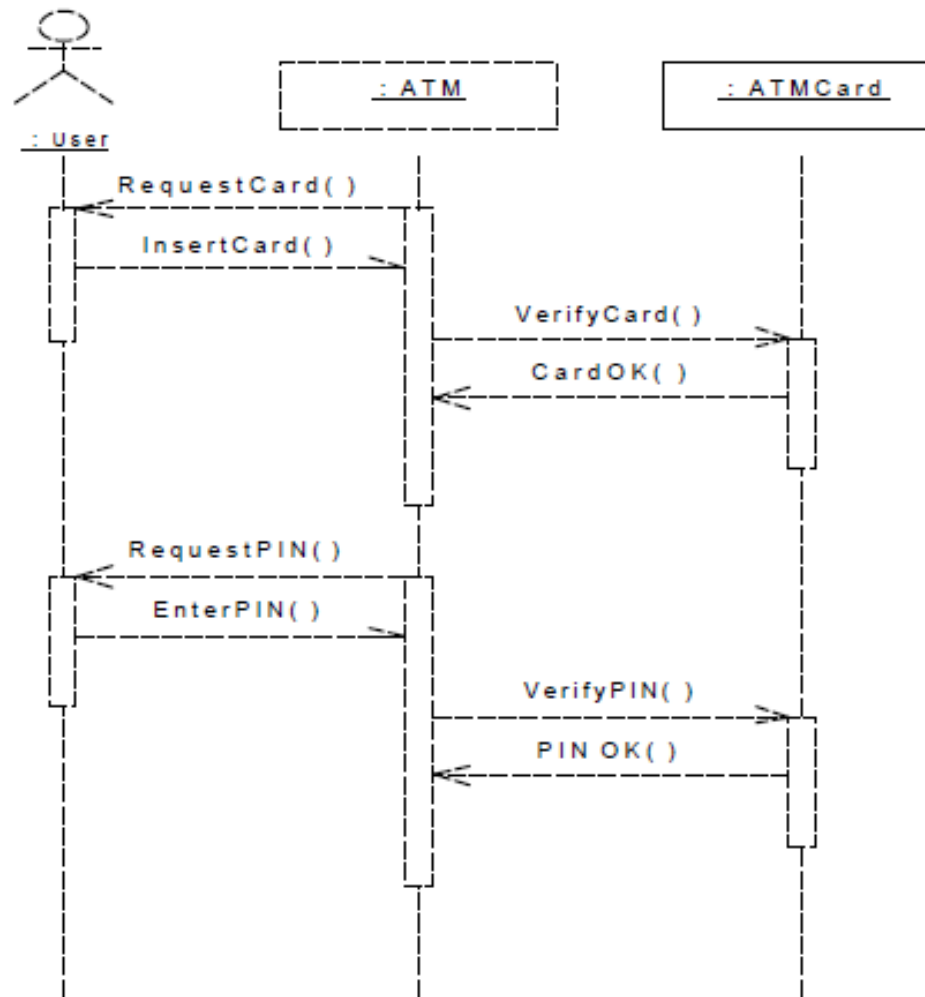
# Notasi Sequence Diagram

NO	GAMBAR	NAMA	KETERANGAN
1		<i>LifeLine</i>	Objek entity, antarmuka yang saling berinteraksi.
		<i>Actor</i>	Digunakan untuk menggambarkan user / pengguna.
2		<i>Message</i>	Spesifikasi dari komunikasi antar objek yang memuat informasi-informasi tentang aktifitas yang terjadi.
3		<i>Boundary</i>	Digunakan untuk menggambarkan sebuah form.
4		<i>Control Class</i>	Digunakan untuk menghubungkan boundary dengan tabel.
5		<i>Entity Clas</i>	Digunakan untuk menggambarkan hubungan kegiatan yang akan dilakukan.

# Contoh Sequence Diagram



# Sequence Diagram Otentifikasi User ATM



# Sequence Diagram Pengambilan Uang di ATM

