



Unified Modelling Language

S2 Teknik Informatika
PENS



UNIFIED MODELLING LANGUAGE

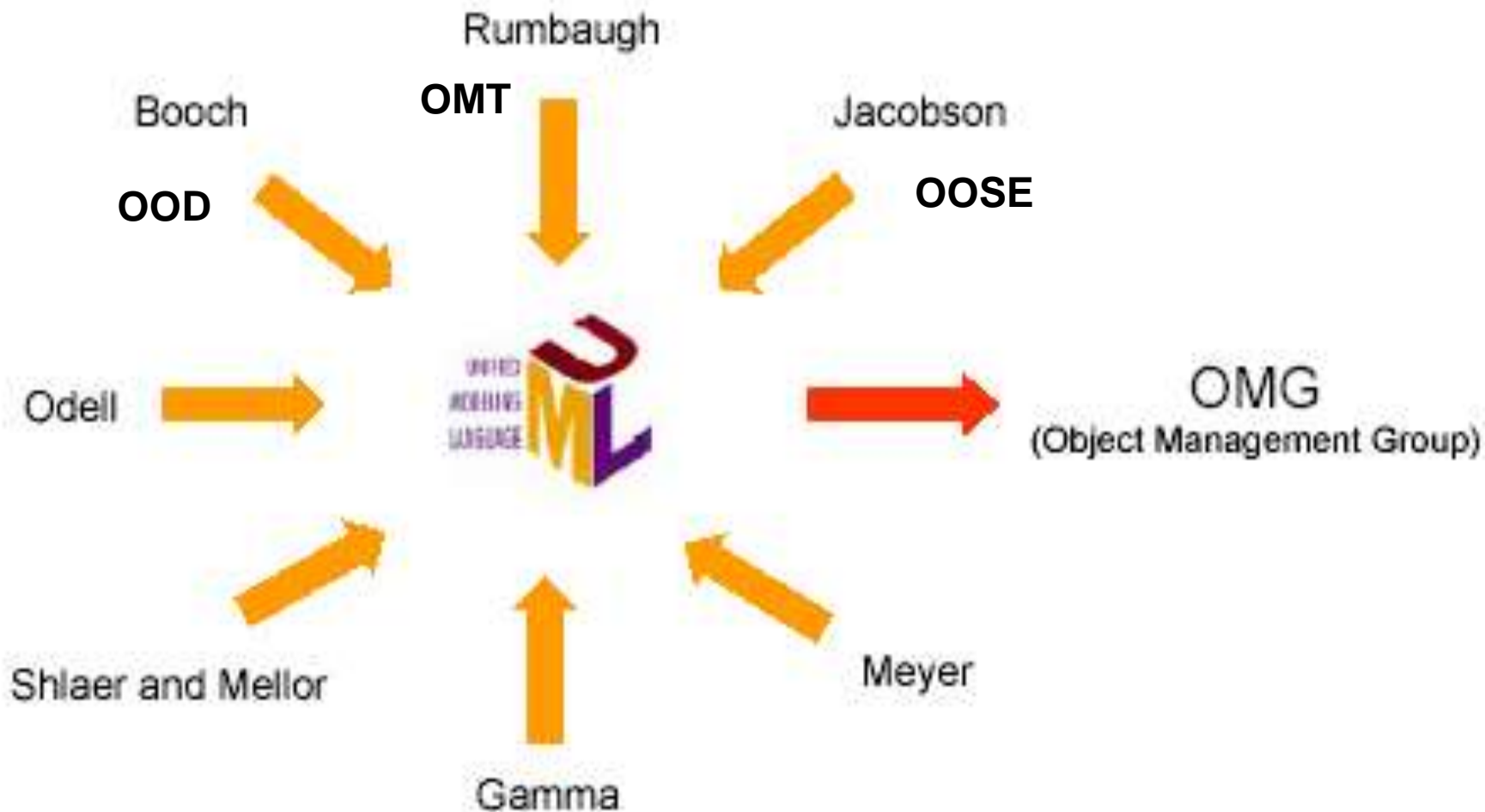
- Unified Modelling Language (UML) adalah sebuah "bahasa" yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem informasi atau piranti lunak.
- UML menawarkan sebuah standar untuk merancang model sebuah sistem.
- Seperti bahasa-bahasa lainnya, UML mendefinisikan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



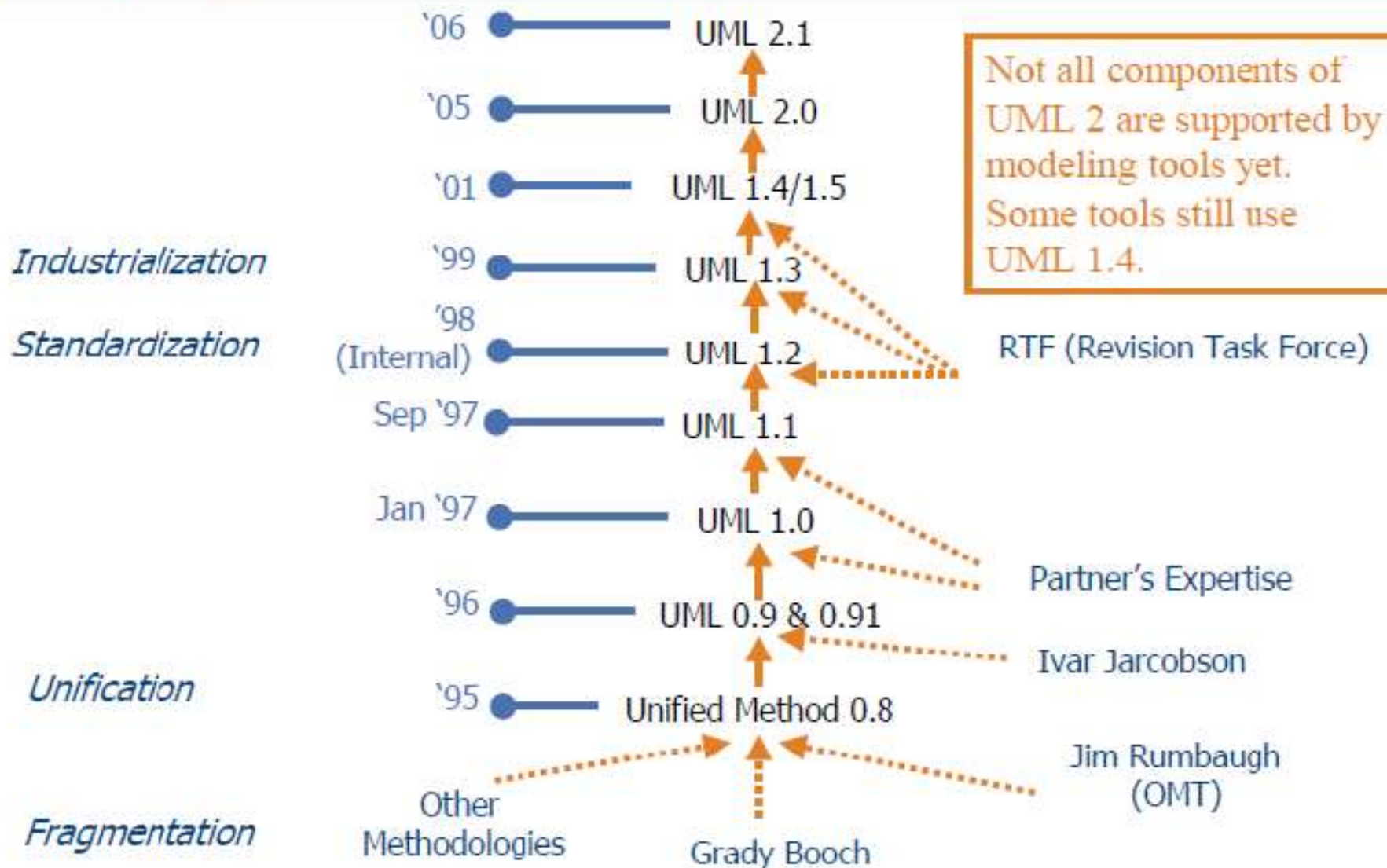
UNIFIED MODELLING LANGUAGE

- 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

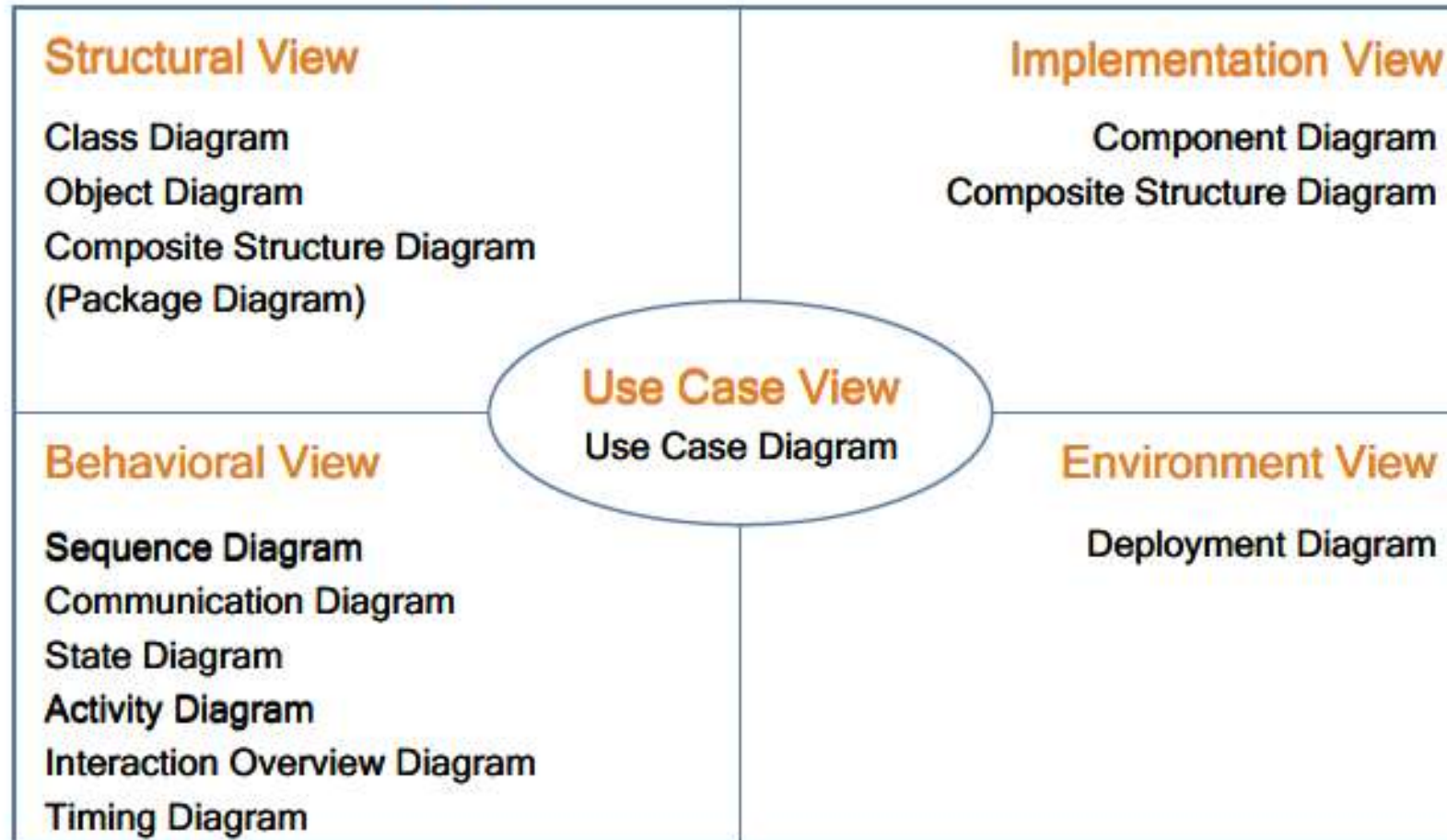


Long Story of UML

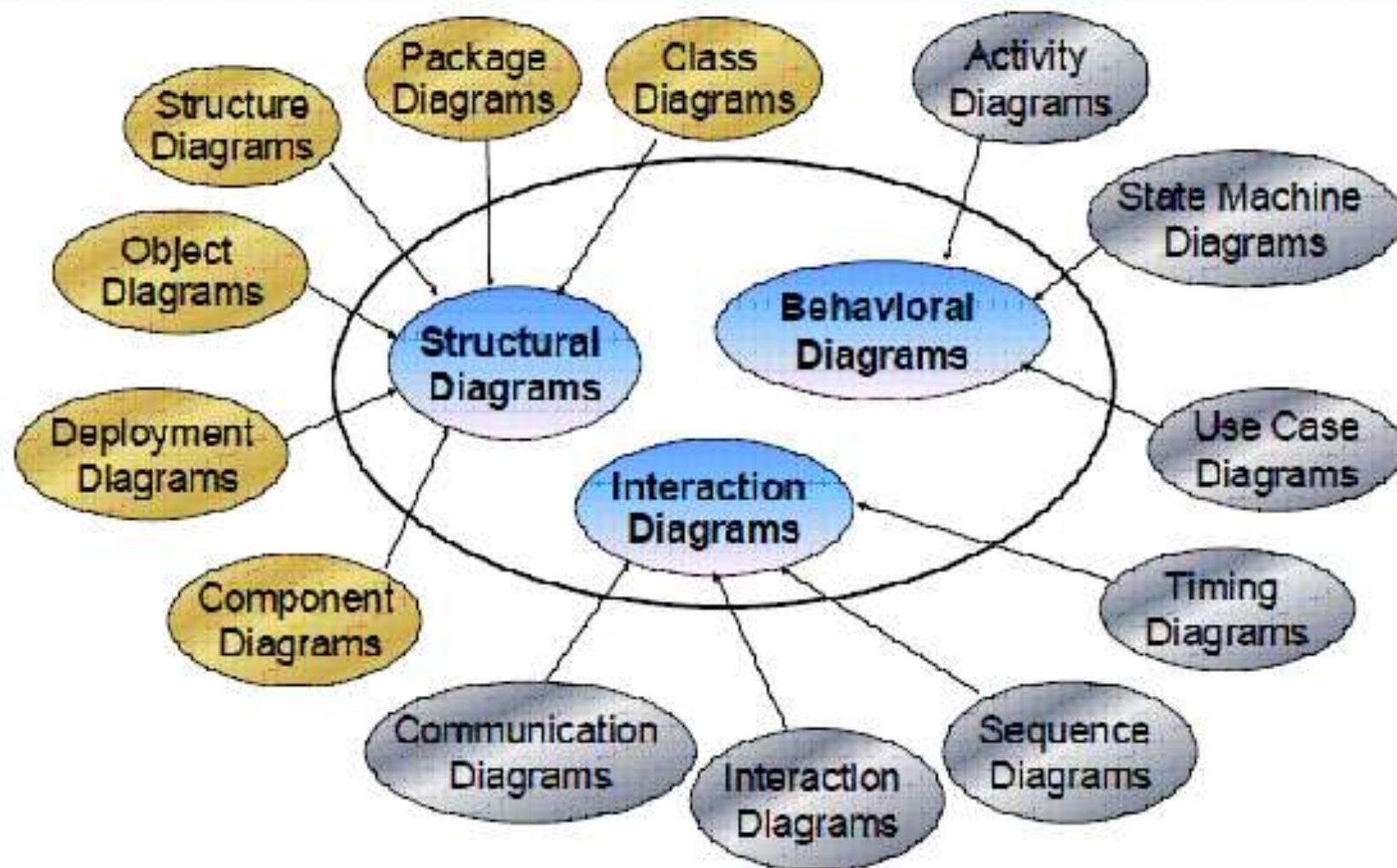


UML Architectural Views and Diagrams

UML defines 13 diagrams that describe 4+1 architectural views

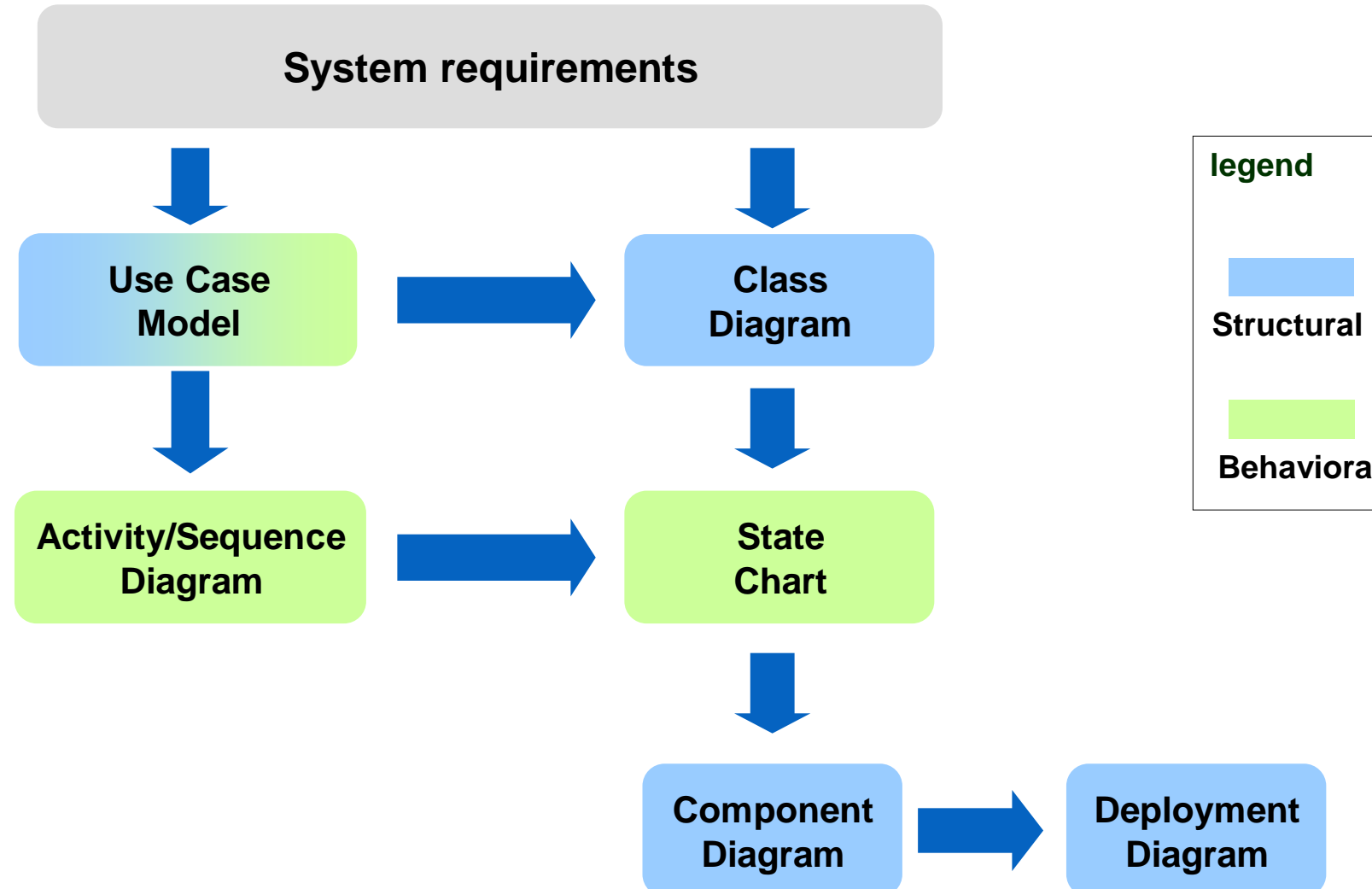


UML 2.0 Diagrams

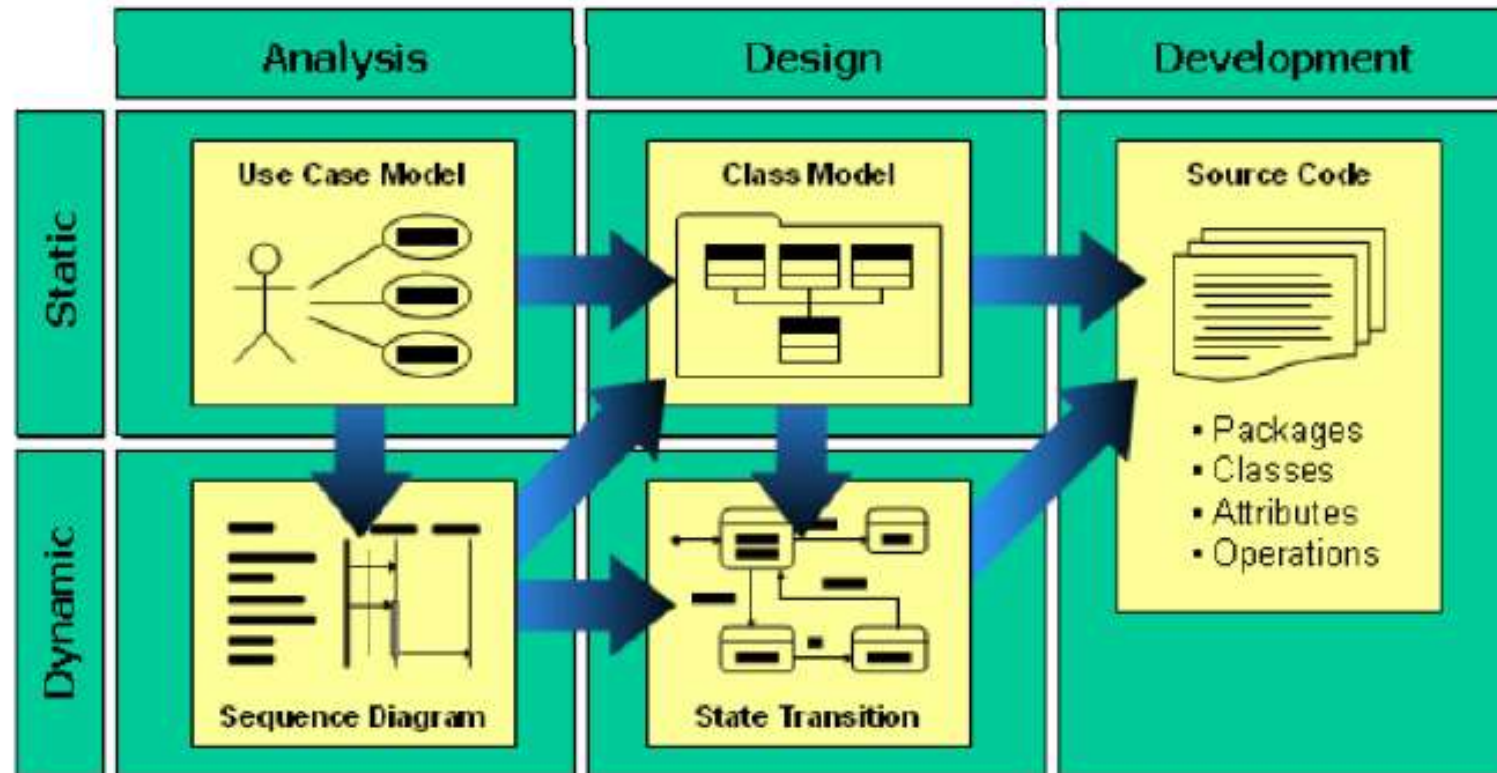


Analysis and Design Process

Zachman Framework



System Development





Use Case Diagram



Mendefinisikan Kebutuhan

- Dalam membuat sebuah sistem, langkah awal yang perlu dilakukan adalah menentukan kebutuhan
- Terdapat dua jenis kebutuhan :
 1. **Kebutuhan fungsional** adalah kebutuhan pengguna dan *stakeholder* sehari-hari yang akan dimiliki oleh sistem, dimana kebutuhan ini akan digunakan oleh pengguna dan *stakeholder*.
 2. **Kebutuhan nonfungsional** adalah kebutuhan yang memperhatikan hal-hal berikut yaitu performansi, kemudahan dalam menggunakan sistem, kehandalan sistem, keamanan sistem, keuangan, legalitas, dan operasional.
- Kebutuhan fungsional akan digambarkan melalui sebuah diagram yang dinamakan diagram use case.



Use Case Diagram

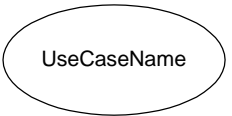
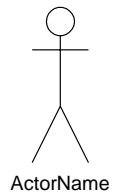
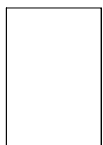
- Diagram Use Case menggambarkan apa saja aktifitas yang dilakukan oleh suatu sistem dari sudut pandang pengamatan luar. Yang menjadi persoalan itu *apa yang dilakukan* bukan *bagaimana melakukannya*.
- Diagram Use Case dekat kaitannya dengan kejadian-kejadian. Kejadian (scenario) merupakan contoh apa yang terjadi ketika seseorang berinteraksi dengan sistem.
- Sebuah *use case* merepresentasikan sebuah interaksi antara aktor dengan sistem. *Use case* merupakan sebuah pekerjaan tertentu, misalnya login ke sistem, meng-*create* sebuah daftar belanja, dan sebagainya.
- Seorang/sebuah aktor adalah sebuah entitas manusia atau mesin yang berinteraksi dengan sistem untuk melakukan pekerjaan-pekerjaan tertentu.





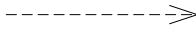
Manfaat Use Case Diagram

- Singkatnya, diagram use case digunakan untuk mengetahui fungsi apa saja yang ada di dalam sebuah sistem dan siapa saja yang berhak menggunakan fungsi-fungsi tersebut
- Diagram Use Case berguna dalam tiga hal :
 - **Menjelaskan fasilitas yang ada (*requirements*)**. Use Case baru selalu menghasilkan fasilitas baru ketika sistem dianalisa, dan *design* menjadi lebih jelas.
 - **Komunikasi dengan klien**. Penggunaan notasi dan simbol dalam diagram Use Case membuat pengembang lebih mudah berkomunikasi dengan klien-nya.
 - **Membuat test dari kasus-kasus secara umum**. Kumpulan dari kejadian-kejadian untuk Use Case bisa dilakukan test kasus layak untuk kejadian-kejadian tersebut.

Use Case Modeling: Core Elements

Construct	Description	Syntax
use case	A sequence of actions, including variants, that a system (or other entity) can perform, interacting with actors of the system.	
actor	A coherent set of roles that users of use cases play when interacting with these use cases.	
system boundary	Represents the boundary between the physical system and the actors who interact with the physical system.	

Use Case Modeling: Core Relationships

Construct	Description	Syntax
association	The participation of an actor in a use case. i.e., instance of an actor and instances of a use case communicate with each other.	
generalization	A taxonomic relationship between a more general use case and a more specific use case.	
extend	A relationship from an <i>extension</i> use case to a <i>base</i> use case, specifying how the behavior for the extension use case can be inserted into the behavior defined for the base use case.	<<extend>> 

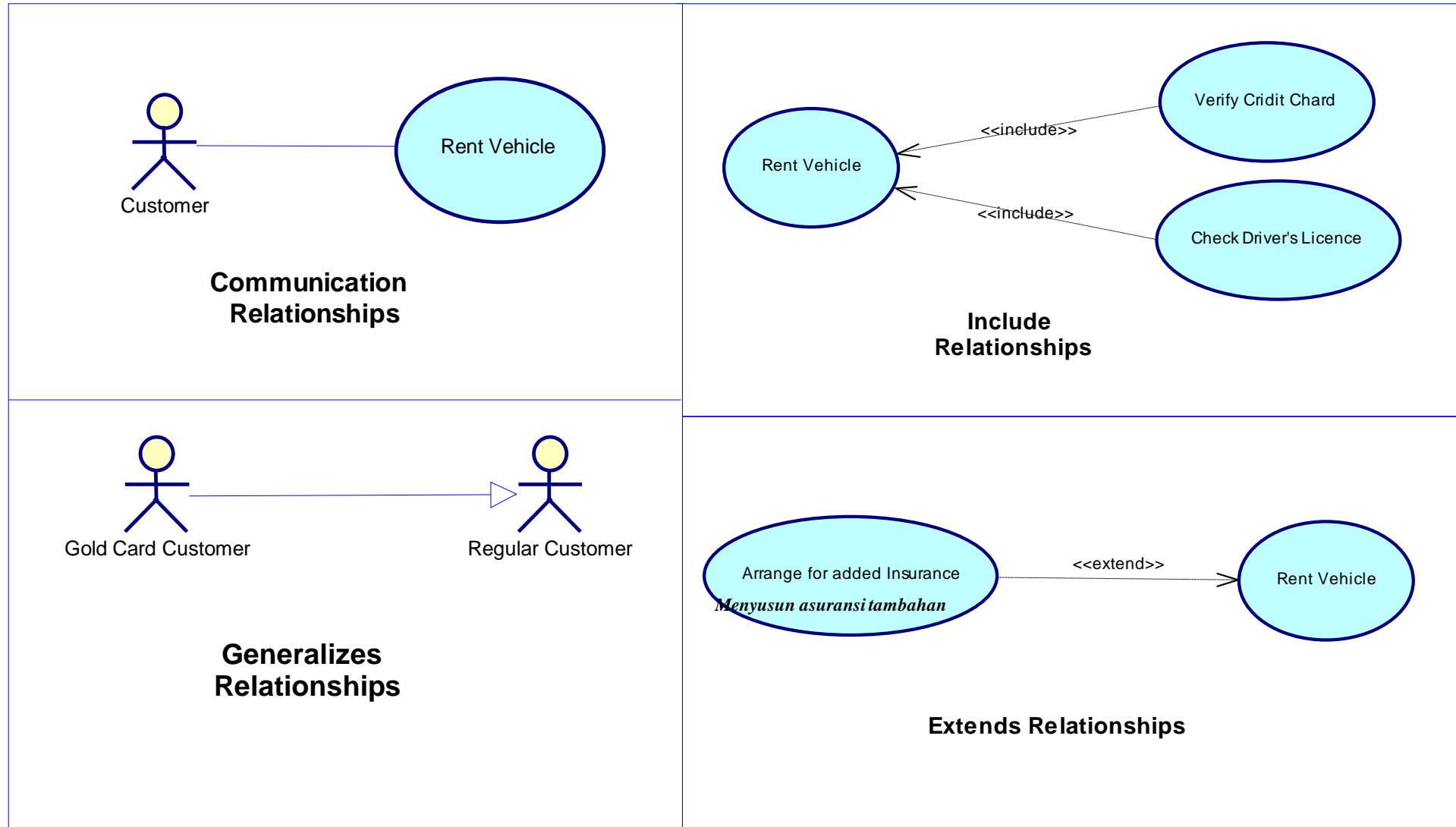
Use Case Modeling: Core Relationships (cont'd)

Construct	Description	Syntax
include	An relationship from a <i>base</i> use case to an <i>inclusion</i> use case, specifying how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.	<<include>> ----->

“Pasien menghubungi klinik untuk membuat janji (appointment) dalam pemeriksaan tahunan. Receptionist mendapatkan waktu yang luang pada buku jadwal dan memasukkan janji tersebut ke dalam waktu luang itu.”



Gambar 1. Contoh kegiatan pasien yang membuat janji



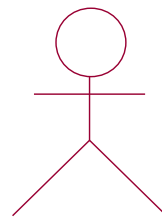


Include dan extend

- Sebuah *use case* dapat meng-*include* fungsionalitas *use case* lain sebagai bagian dari proses dalam dirinya. Secara umum diasumsikan bahwa *use case* yang di-*include* akan dipanggil setiap kali *use case* yang meng-*include* dieksekusi secara normal.
- Sebuah *use case* dapat di-*include* oleh lebih dari satu *use case* lain, sehingga duplikasi fungsionalitas dapat dihindari dengan cara menarik keluar fungsionalitas yang *common*.
- Sebuah *use case* juga dapat meng-*extend* *use case* lain dengan *behaviour*-nya sendiri.
- Sementara hubungan generalisasi antar *use case* menunjukkan bahwa *use case* yang satu merupakan spesialisasi dari yang lain.

Komponen Use Case

1. **Aktor** adalah seseorang atau apa saja yang berhubungan dengan sistem yang sedang dibangun.
 - Aktor sebaiknya diberinama dengan kata benda.
 - Ketika memberi nama actor, gunakan nama peranan dan jangan nama posisi. Seorang individu dapat memainkan beberapa peranan.
 - Dalam UML direpresentasikan dengan notasi berikut ini:



Pasien

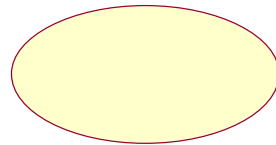


Tipe Aktor

- Ada 3 tipe aktor :
 1. Pengguna sistem. Merupakan actor secara fisik atau seorang pengguna, gambaran secara umum dan selalu ada pada setiap sistem
 2. Sistem yang lain dan berhubungan dengan sistem yang dibangun. Misalkan pada sebuah sistem Informasi Puskesmas memerlukan koneksi dengan aplikasi sistem yang lain, semisal SIM rumah sakit. Maka dalam kasus ini, SIM rumah sakit adalah actor.
 3. Waktu. Dapat menjadi actor jika seiring perjalanan waktu dapat memicu event/kejadian dalam sistem.

Komponen Use Case (2)

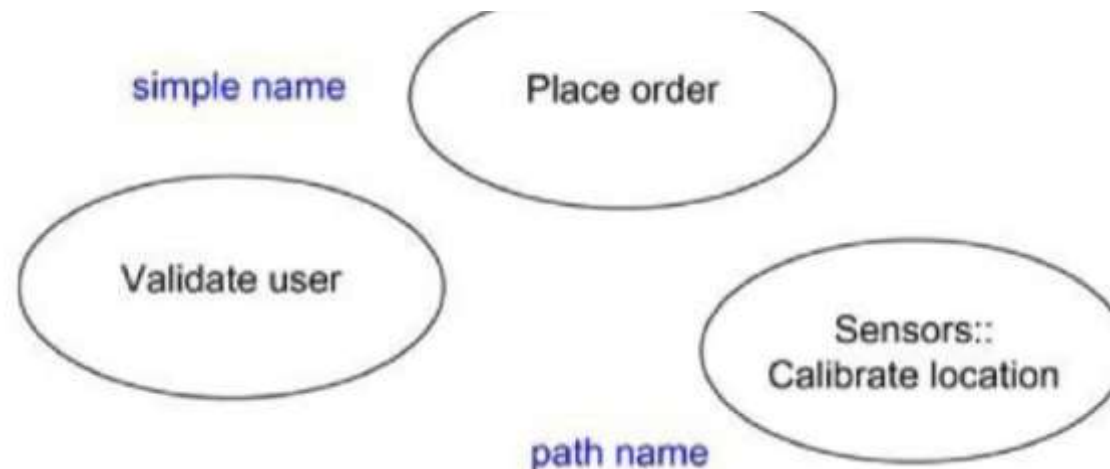
- 2. Use Case.** Adalah bagian fungsionalitas tingkat tinggi yang disediakan oleh sistem.
- Dengan kata lain, use case menggambarkan bagaimana seseorang menggunakan sistem.
 - Use Case dalam UML dinotasikan dengan simbol



Pendaftaran Pasien

Pemberian nama use case

- Pemberian nama use case sebaiknya :
- Simple name. Biasanya berupa kata kerja + kata benda
- Path name. Merupakan nama di bagian depan yang menyatakan paket (package) dimana use case tersebut berada



Komponen Use Case (3)

3. Relasi yang menghubungkan antara actor dan use case

- Arah panah menunjukkan siapa yang mengawali komunikasi.
- Dengan mengecualikan use case dalam relasi include dan relasi extend, setiap use case harus diinisialisasi oleh actor
- Relasi dinotasikan seperti gambar berikut :





Jenis Relasi

- Ada 3 jenis relasi
 - Generalization
 - Include
 - Extends

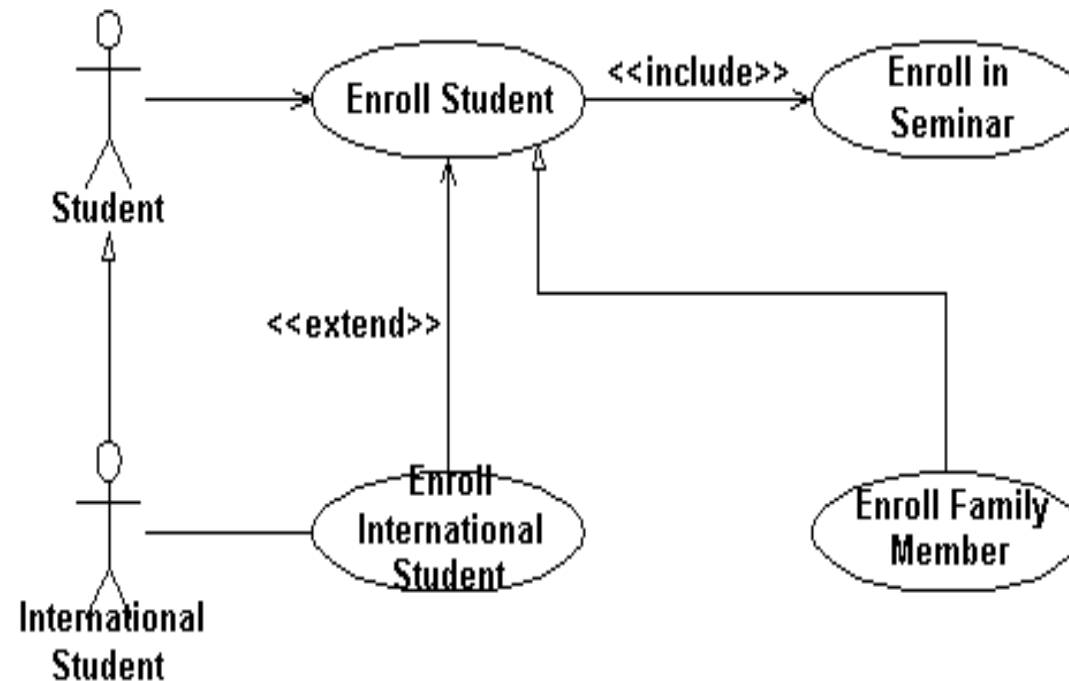


Generalization

- Hubungan antara induk dan anak
- Anak mewarisi sifat dan method dari induk
- Induk disebut root / base
- Class yang tidak memiliki anak disebut leaf
- Gambarkan generalization/inheritance antara use case secara vertical dengan inheriting use case dibawah base/parent use case
- Generalization/inheritance dipakai ketika ada sebuah keadaan yang lain sendiri/perlakuan khusus (*single condition*)
- Terbagi menjadi 2 :
 - Actor Generalization
 - Use Case Generalization

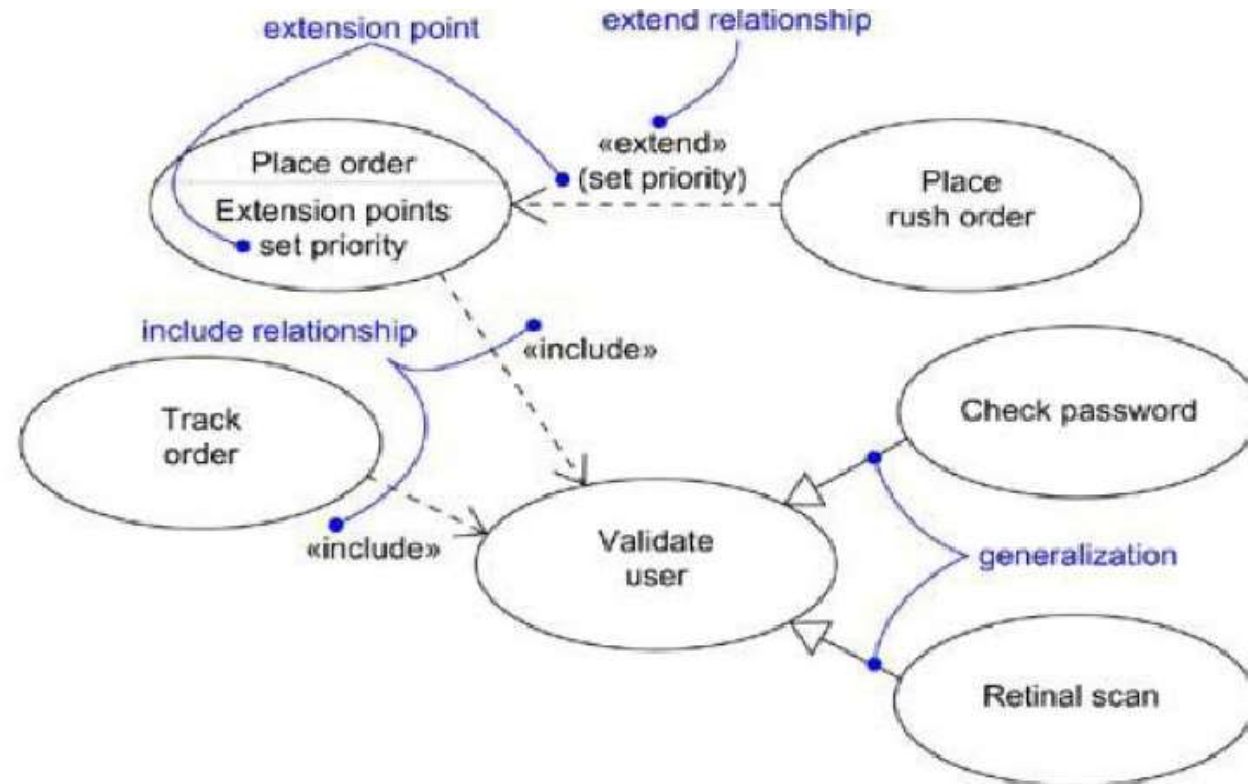
Actor Generalization

- Aktor bisa umum atau spesifik
- Gambarkan generalization/inheritance antara actors secara vertical dengan inheriting actor di bawah base/parent use case



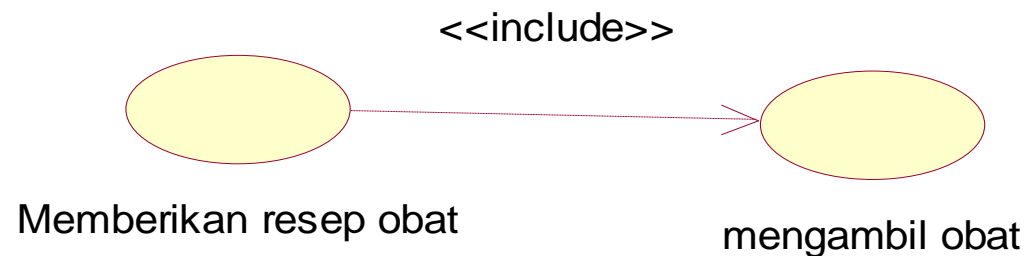
Use Case Generalization

- Use case anak mewarisi arti dari use case induk sambil menambahkan/modifikasi behaviour dari induk



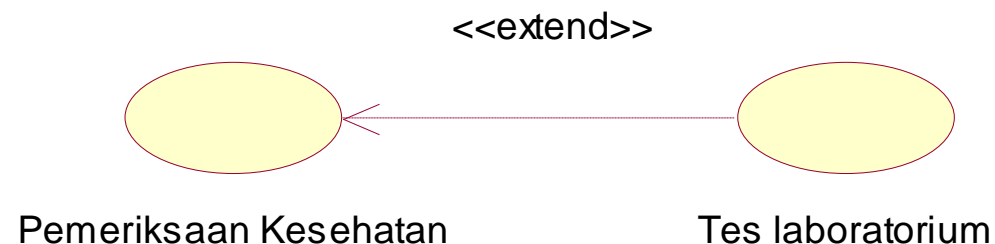
Include

- Memungkinkan (required/harus) satu use case menggunakan fungsionalitas yang disediakan oleh use case lainnya.
- Tanda panah terbuka harus terarah ke sub use case
- Gambarkan association include secara horizontal



Extend

- Memungkinkan suatu use case secara **optional** menggunakan fungsionalitas yang disediakan oleh use case lainnya.
- Kurangi penggunaan association Extend ini, terlalu banyak pemakaian association membuat diagram sulit dipahami.
- Tanda panah terbuka harus terarah ke parent/base use case
- Gambarkan association extend secara vertical
- Contoh :
 - Use case pemeriksaan kesehatan suatu saat memerlukan tes laboratorium, tapi pada saat lain tidak. Tergantung pada kondisi pasien yang diperiksa.

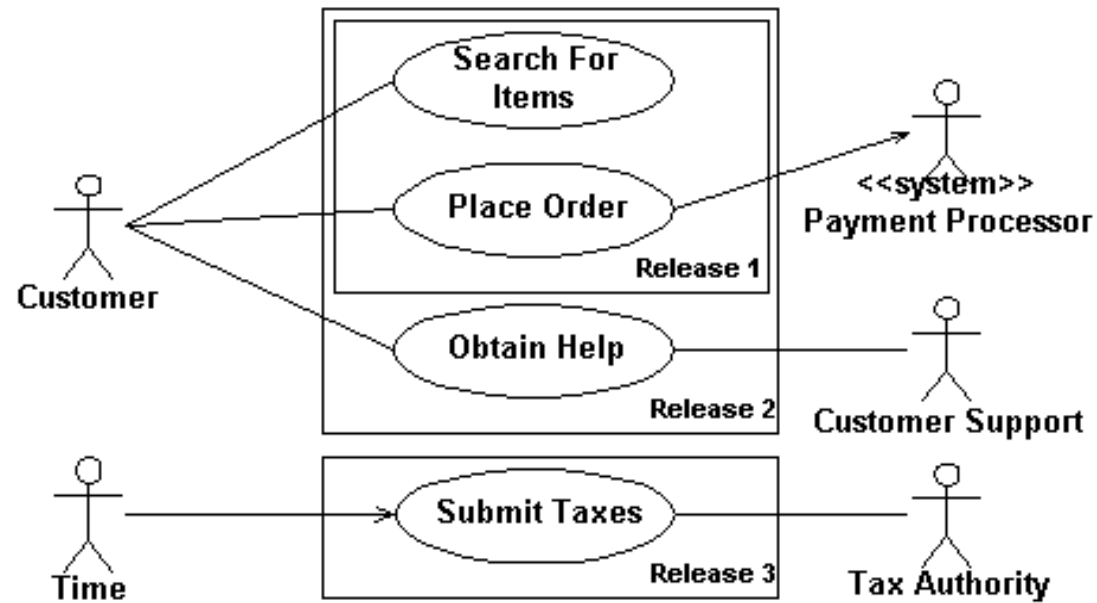


System Boundary

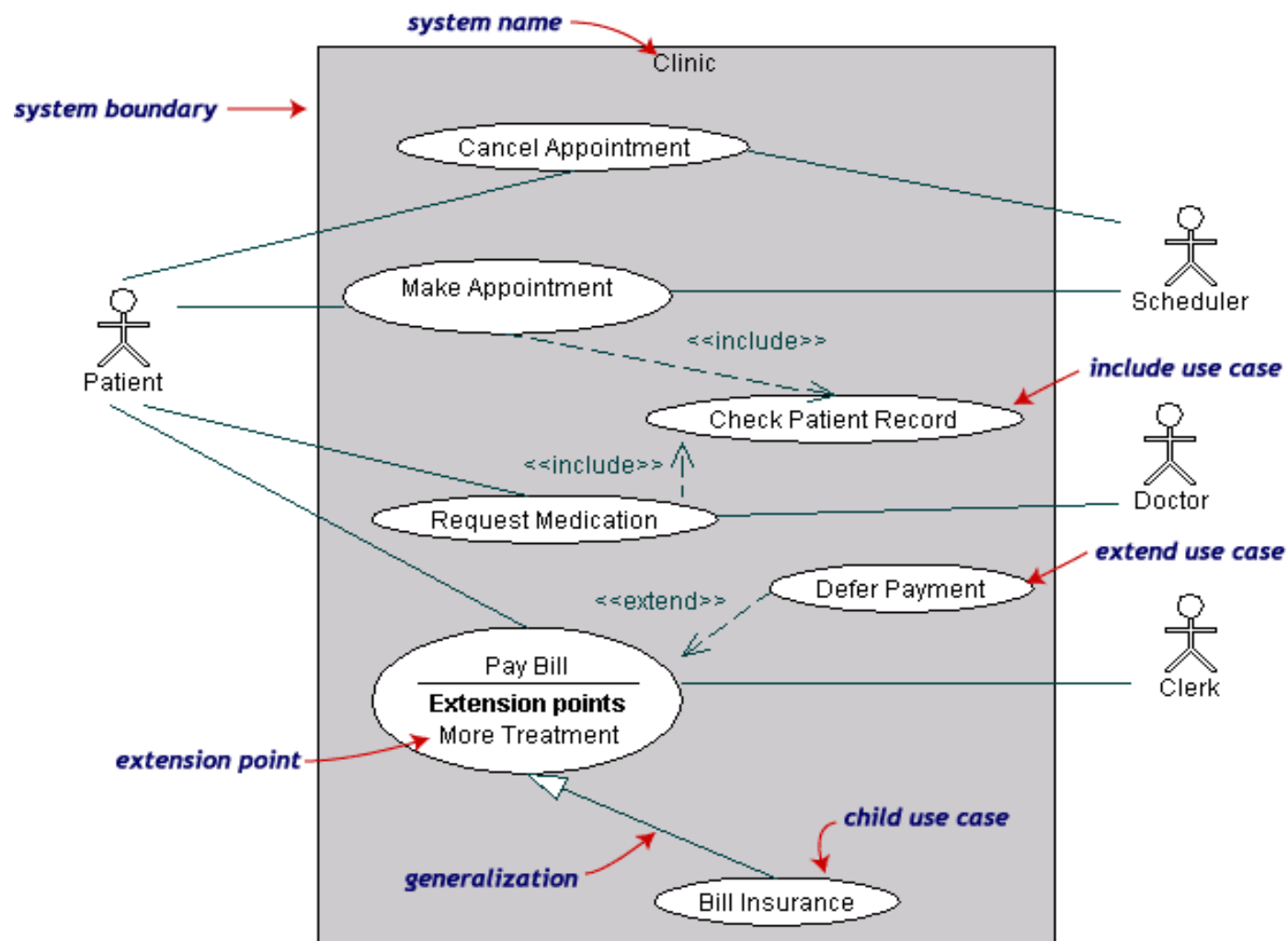
- Untuk memperlihatkan batasan sistem dalam diagram use case, Anda dapat menggambar sebuah kotak yang melingkupi semua use case, namun actor tetap berada di luar kotak.
- System boundary boxes dalam penggunaannya optional



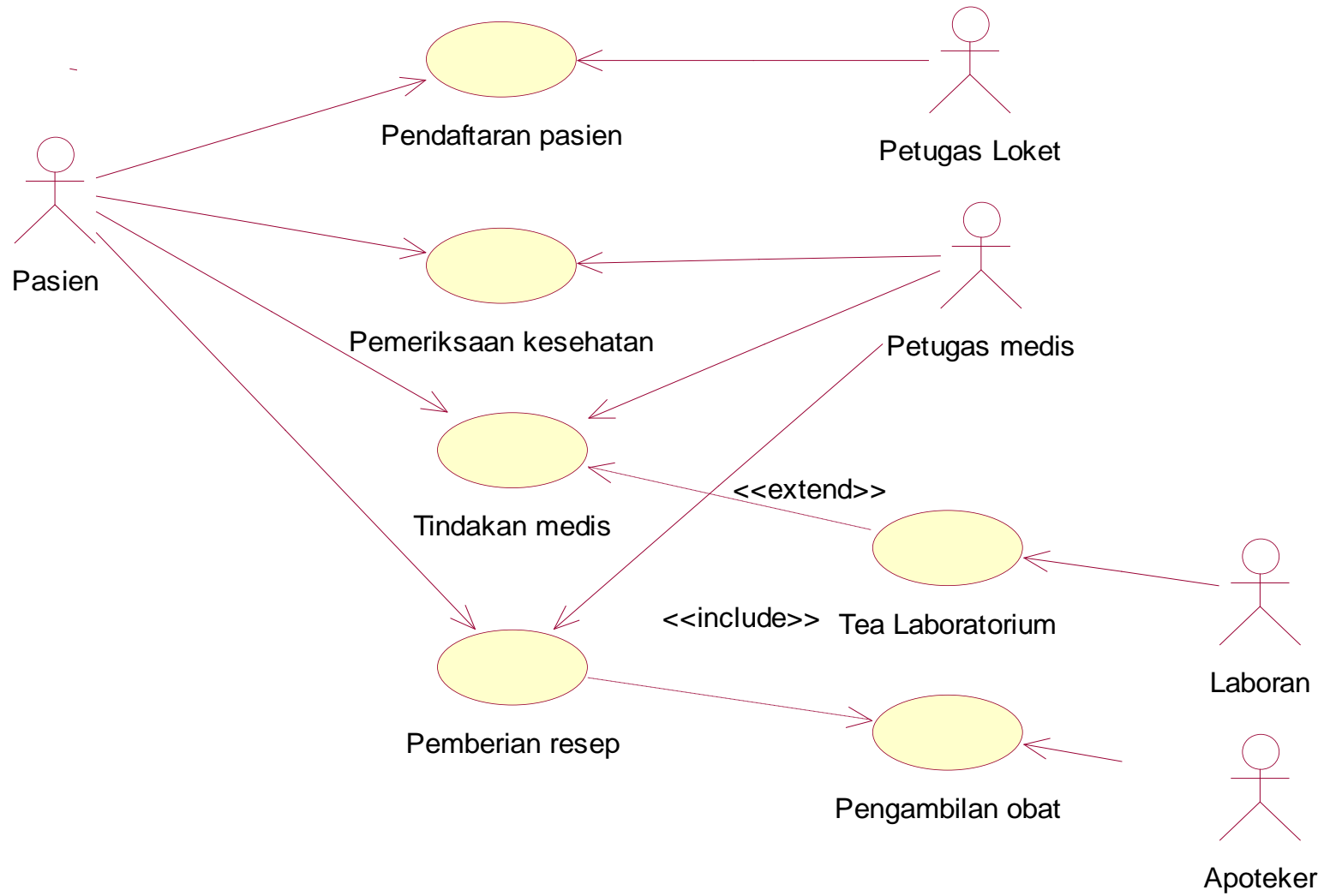
- Contoh system boundaries yang lain



Contoh Use Case Diagram



Sistem Informasi Puskesmas





DO AND DONT'S



Bagaimana Menemukan Aktor ?

- Pekerjaan awal adalah menemukan aktor, menemukan fungsionalitas dan membatasi sistem yang akan dibuat.
- Pembatasan sistem ini penting untuk menemukan aktor. Karena dari sinilah kita akan menentukan apakah sesuatu itu adalah aktor dan apakah aktor tersebut akan berbentuk orang atau sistem lain
- Cara mudah untuk menemukan aktor adalah dengan bertanya hal-hal berikut:
 - SIAPA yang akan menggunakan sistem?
 - APAKAH sistem tersebut akan memberikan NILAI bagi aktor?



- Tidak semua aktor adalah manusia, bisa saja sistem lain yang berinteraksi dengan sistem yang anda buat.
- Untuk menemukan sistem lain sebagai aktor, hal-hal di bawah ini bisa menjadi pertimbangan :
 - Jika anda bergantung pada sistem lain untuk melakukan sesuatu, maka sistem lain itu adalah aktor.
 - Jika sistem lain itu meminta (*request*) informasi dari sistem anda, maka sistem lain itu adalah aktor
- Untuk penamaan aktor diberi nama sesuai dengan PERAN-nya

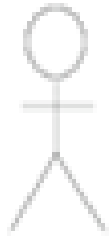
Contoh

- Berikut ini contoh menemukan aktor pada sistem pencatatan penjualan di Supermarket.

Pertanyaan	Analisis
Siapa sajakah yang berinteraksi dengan sistem pencatatan penjualan di supermarket?	<ul style="list-style-type: none"> Bagian yang akan mencatat penjualan barang Bagian yang ingin tahu berapa besar keuntungan yang didapatkan Bagian yang ingin tahu berapa banyak produk yang berkurang
Peran apa saja yang terlibat?	Kasir, manajer, bagian gudang.
Nilai apa sajakah yang akan diberikan sistem kepada aktor?	<p>Nilai bagi kasir:</p> <ul style="list-style-type: none"> la akan mendapatkan struk belanja. Lama aktivitas kerja akan terekam kedalam sistem. <p>Nilai bagi manajer</p> <ul style="list-style-type: none"> la perlu mengetahui laporan

	<p>keuntungan dalam rentang waktu tertentu</p> <p>Nilai bagi bagian gudang</p> <ul style="list-style-type: none">• Ia perlu mengetahui produk apa saja yang berkurang
<p>Apakah sistem pencatatan penjualan bergantung pada sesuatu?</p>	<p>Printer</p> <ul style="list-style-type: none">• Untuk mencetak struk <p>Mesin debit ATM</p> <ul style="list-style-type: none">• Untuk menarik sejumlah uang pada <i>account</i> seseorang

Jadi...Aktornya adalah



Kasir



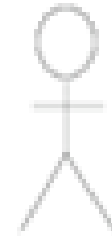
Manajer



Bagian
Gudang



Printer



Mesin debit ATM

- INGAT !! Dalam kasus ini, PELANGGAN tidak berinteraksi langsung dengan sistem, KASIR yang berinteraksi langsung dengan sistem



- Sistem dibangun untuk menyediakan kebutuhan bagi aktor, jika suatu saat nanti *stakeholder* akan menentukan bahwa sistem pencatatan penjualan akan berinteraksi dengan pelanggan, maka aktor di atas pun tentu saja akan berubah.
- Inilah yang dimaksud dengan batasan sistem.
- *Stakeholder* dan pengguna akan menentukan batasan sistem yang akan dibuat.



Menemukan Use Case

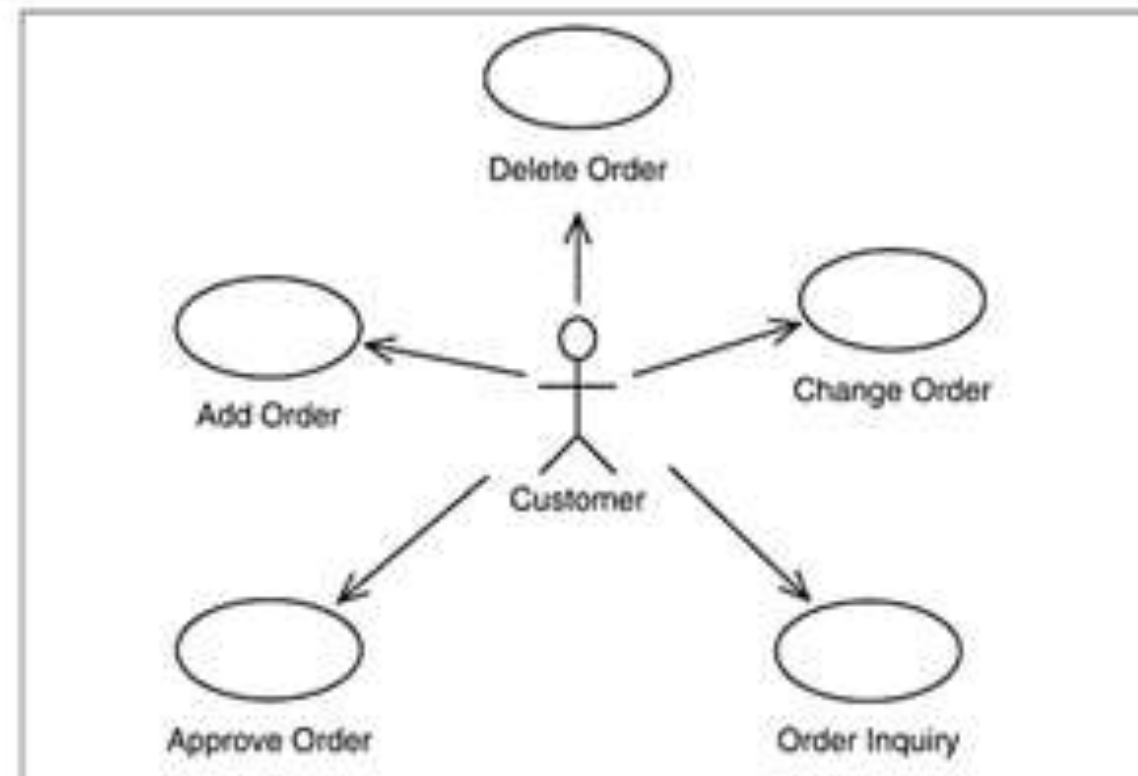
- Jika anda sudah berhasil menemukan aktor, maka untuk menemukan use case akan lebih mudah dilakukan.
- Sebuah use case harus mendeskripsikan **sebuah pekerjaan dimana pekerjaan tersebut akan memberikan NILAI yang bermanfaat bagi aktor**
- Untuk menemukan use cases, mulailah dari sudut pandang aktor, misalnya dengan bertanya :
 - Informasi apa sajakah yang akan didapatkan aktor dari sistem?
 - Apakah ada kejadian dari sistem yang perlu diberitahukan ke aktor?



- Sedangkan dari sudut pandang sistem, misalnya dengan pertanyaan sebagai berikut :
 - Apakah ada informasi yang perlu disimpan atau diambil dari sistem?
 - Apakah ada informasi yang harus dimasukkan oleh aktor?

Common Errors

- Seringkali sebuah use case dianggap sebagai sebuah “*function*” atau item menu. Hal ini adalah **SALAH**.
- Perhatikan contoh berikut:





Mengapa Salah ?

- Use case di atas menggambarkan mengenai apa yang harus dilakukan oleh sistem yang terdiri dari beberapa proses yaitu menyetujui pemesanan, memesan informasi, mengubah pemesanan, menghapus pemesanan, dan menambah pemesanan.
- Diagram di atas memperlihatkan proses penguraian fungsi-fungsi (*functional decomposition*) yaitu mengurai proses kedalam bagian yang lebih kecil.
- Hal ini adalah salah karena use case di atas tidak memberikan nilai kepada aktor.



- Cobalah bertanya seperti ini: Apakah saya akan menggunakan proses mengubah pemesanan jika saya tidak pernah melakukan pemesanan? Tentu saja tidak. Semua proses di atas akan menjadi berguna jika terdapat proses melakukan pemesanan, dan semua proses di atas sebenarnya berkaitan dengan melakukan pemesanan

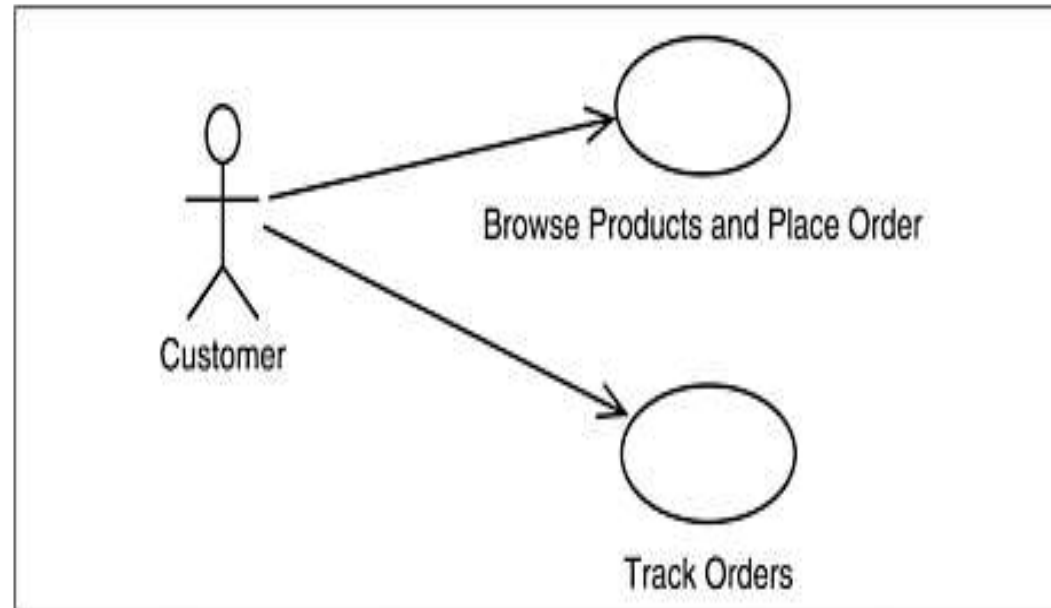


Apanya yang salah ?

- Diagram di atas tidak memberikan nilai kepada aktor, atau dengan kata lain jika kita menggambarkan diagram seperti di atas, nilai akan menjadi hilang.
- Sebuah use case seharusnya dibuat untuk menghasilkan suatu nilai kepada aktor, pada level tertentu jika aktor melakukan pemesanan maka proses tersebut akan memberikan nilai kepada aktor.
- Tapi jika proses pemesanan saja tidak pernah dilakukan, apakah hal ini akan memberikan nilai? Tentu saja tidak.

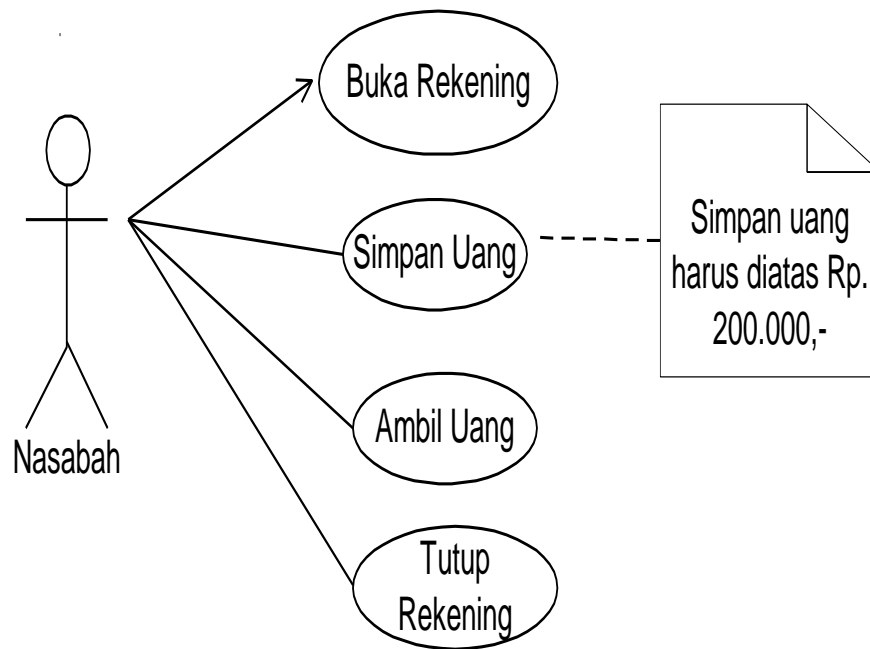
Yang BENAR adalah

- Gambarlah diagram use case yang berfokus pada nilai yang akan diberikan kepada aktor.

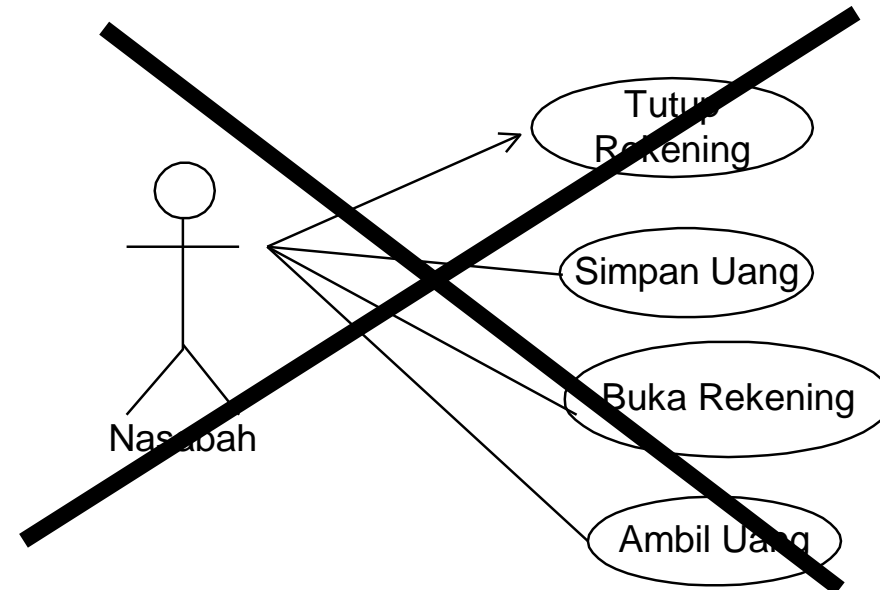


Perhatikan urutan Use Case

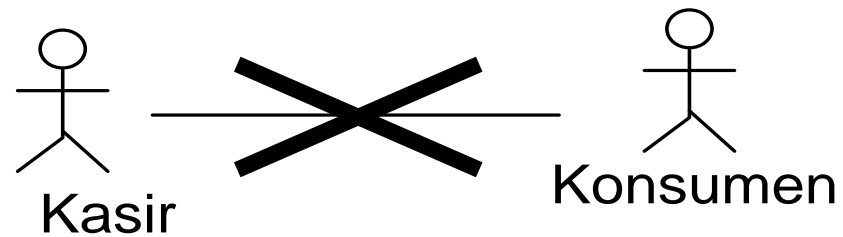
• BENAR



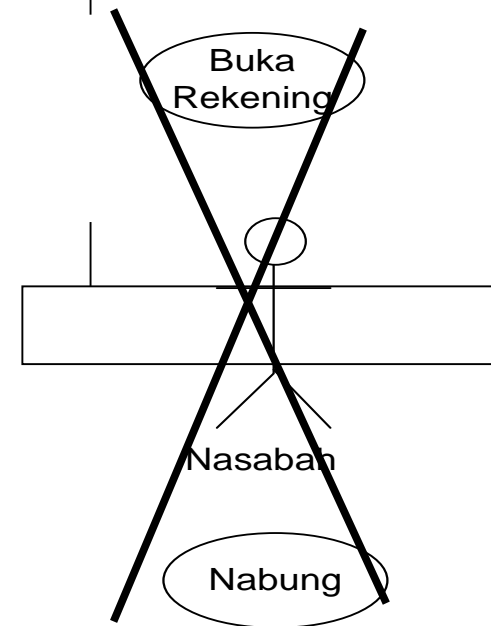
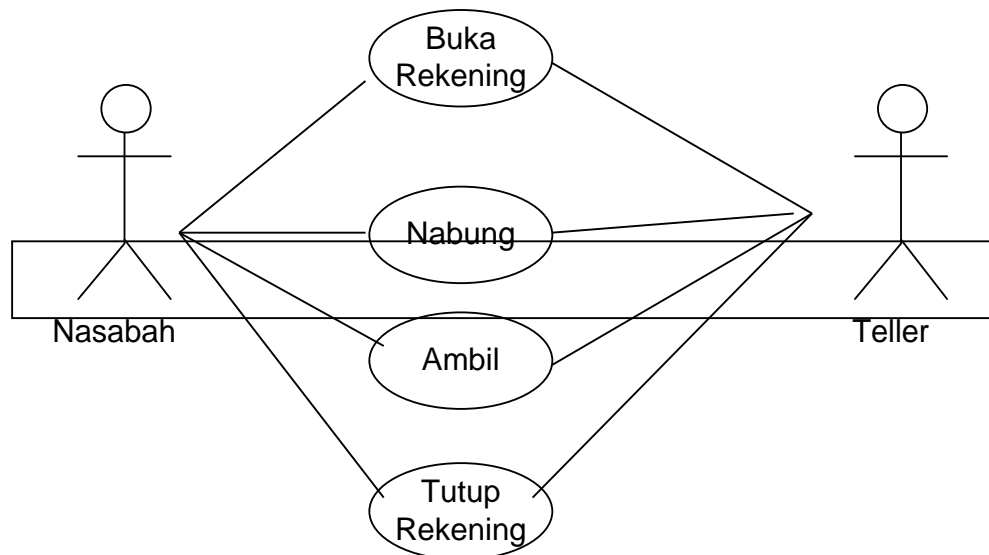
• SALAH



- Tidak boleh ada komunikasi langsung antar actor (Actors don't interact with one another)



- Letakkan actor utama pada pojok kiri atas dari diagram
- Mengapa ? in western culture people read from left to right, top to bottom
- Actor jangan digambarkan ditengah-tengah use cases





Deskripsi Use Case

- Setiap use case harus dijelaskan alur prosesnya melalui sebuah deskripsi use case (*use case description*) atau scenario use case
- Deskripsi use case berisi:
 1. Nama use case yaitu penamaan use case yang menggunakan kata kerja
 2. Deskripsi yaitu penjelasan mengenai tujuan use case dan nilai yang akan didapatkan oleh aktor
 3. Kondisi sebelum (*pre-condition*) yaitu kondisi-kondisi yang perlu ada sebelum use case dilakukan.



4. Kondisi sesudah (*post-condition*) yaitu kondisi-kondisi yang sudah dipenuhi ketika uses case sudah dilaksanakan
5. Alur dasar (*basic flow*) yaitu alur yang menceritakan jika semua aksi yang dilakukan adalah benar atau proses yang harusnya terjadi
6. Alur alternatif (*alternatif flow*) yaitu alur yang menceritakan aksi alternatif, yang berbeda dari alur dasar.

Contoh Deskripsi

Nama Use case: Login

Skenario:

Aksi Aktor	Reaksi Sistem
Skenario Normal	
1. Memasukkan id dan password	
	2. Mengecek valid tidaknya data masukan
	3. Masuk ke aplikasi pengelolaan data perpustakaan
Skenario Alternatif	
1. Memasukkan id dan password	
	2. Mengecek valid tidaknya data masukan
	3. Menampilkan pesan login tidak valid
4. Memasukkan id dan password yang valid	
	5. Mengecek valid tidaknya data masukan
	6. Masuk ke aplikasi



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.

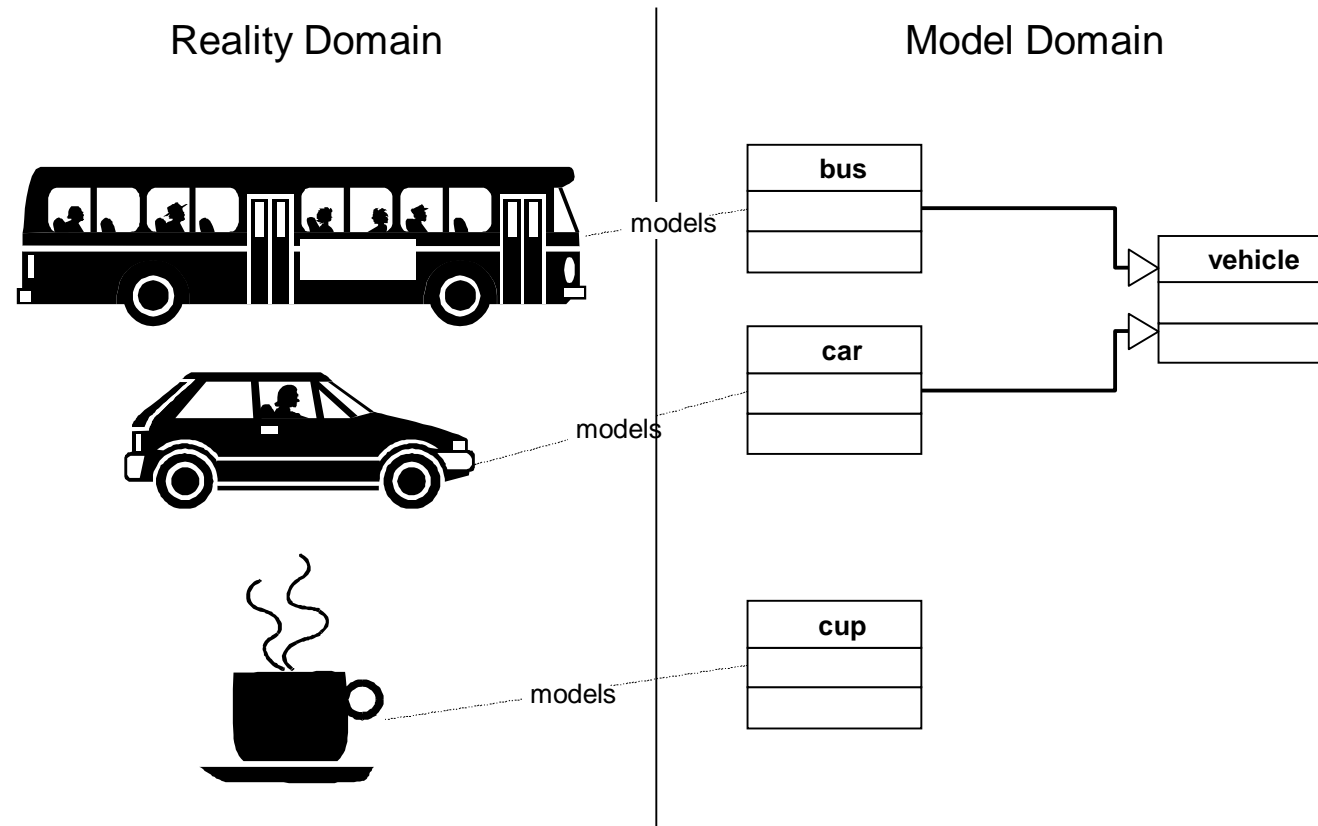


Class Diagram

- 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
- Public, dapat dipanggil oleh siapa saja

Object-Oriented Approach

- Objects are abstractions of real-world or system entities





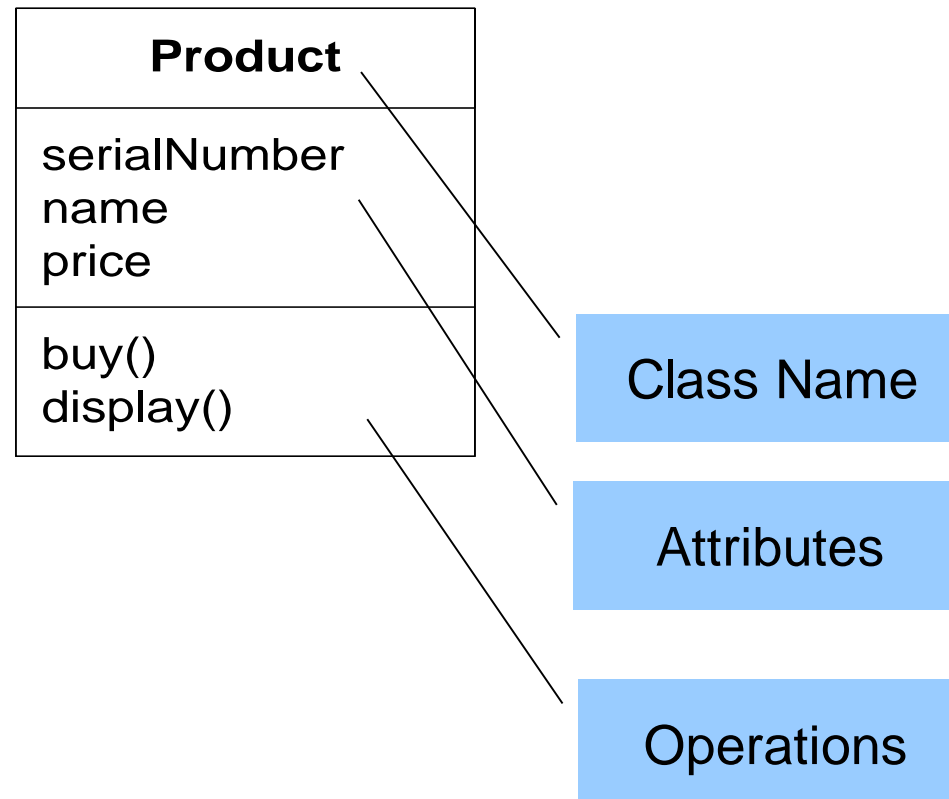
Object-Oriented Approach

- **Object** adalah gambaran dari entity, baik dunia nyata atau konsep dengan batasan-batasan 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 *Identity* (identitas).

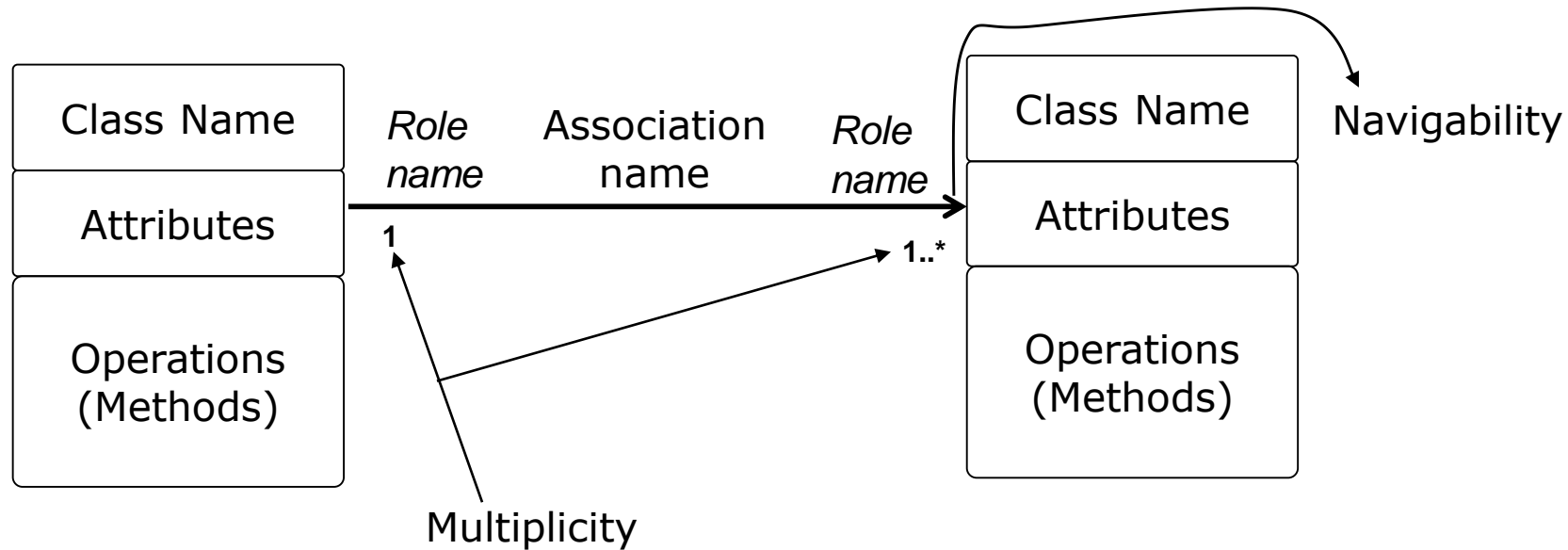


Classes

- A class is a template for actual, in-memory, instances



Class Diagram Format and association:



Multiplicity Notation

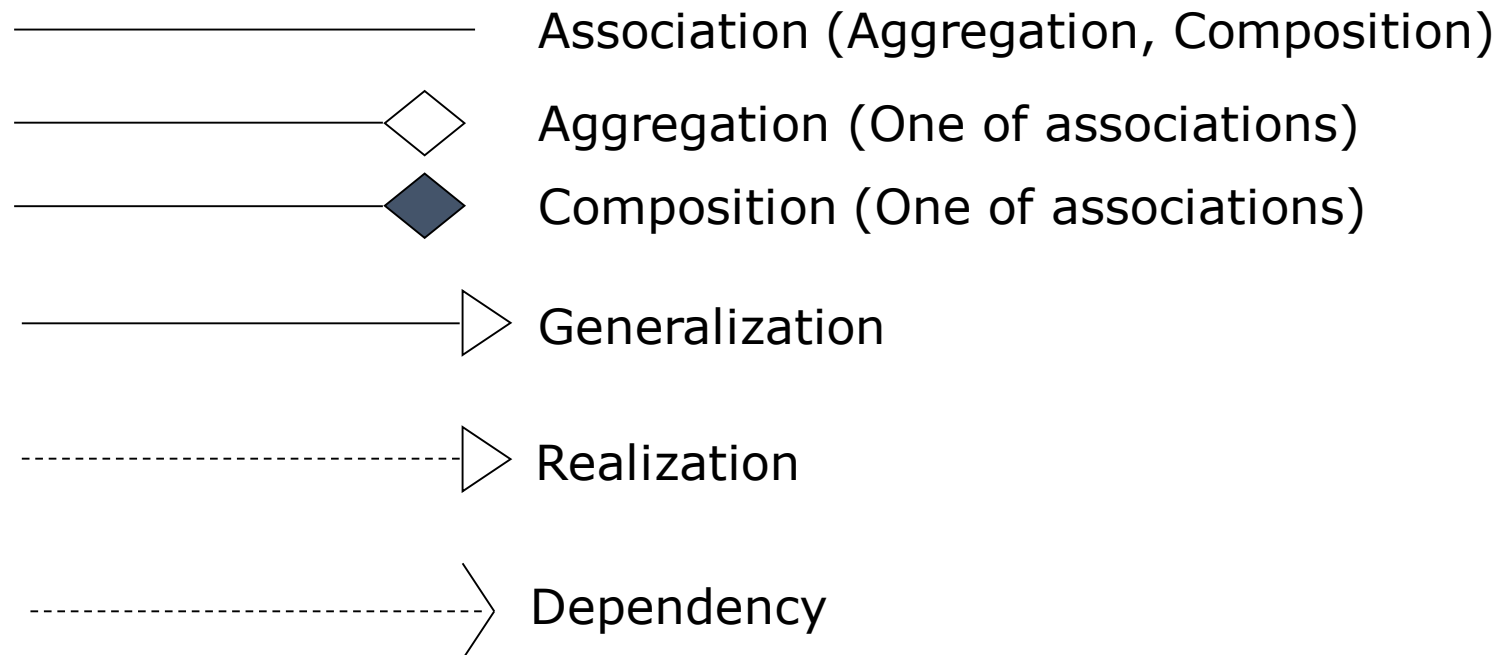
- 1 : One and only one
- 0..* : None or more
- 1..* : One or more
- 0..1 : None or one



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

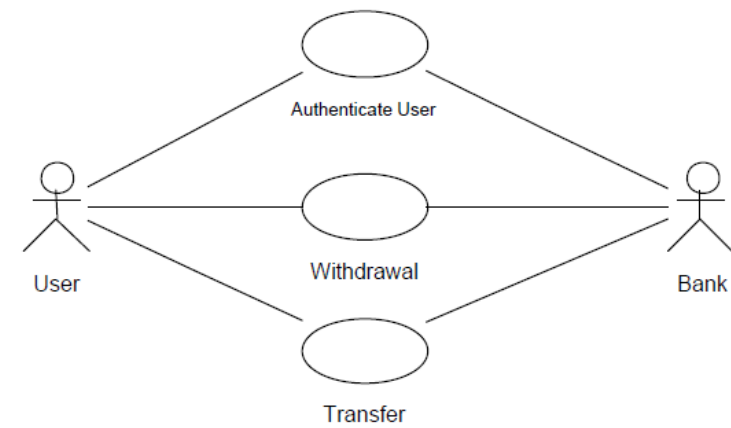
Multiplicity dari suatu titik *association* adalah angka kemungkinan bagian dari hubungan kelas dengan single *instance* (bagian) pada titik yang lain. *Multiplicity* berupa single number (angka tunggal) atau range number (angka batasan). Pada contoh, hanya bisa satu 'Customer' untuk setiap 'Order',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

Tabel di bawah mengenai *multiplicity* yang sering digunakan :

Tabel *Multiplicity*

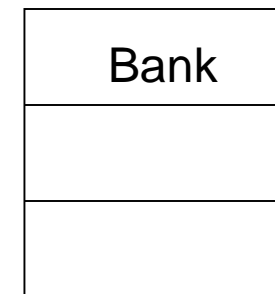
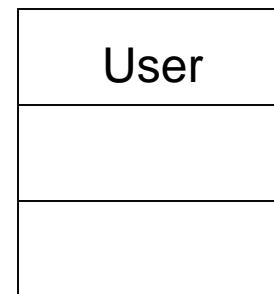
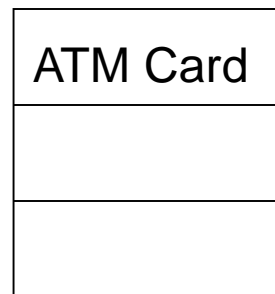
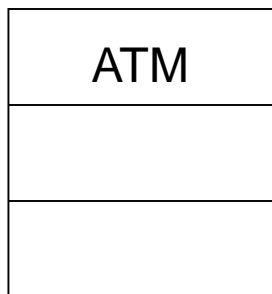
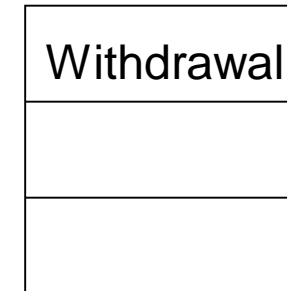
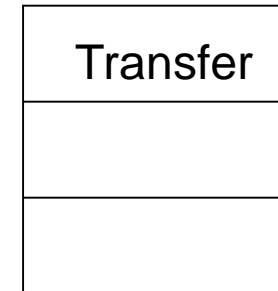
Multiplicities	Artinya
0..1	Nol atau satu bagian. Notasi $n . . m$ menerangkan n sampai m bagian.
0..* or *	Tak hingga pada jangkauan bagian (termasuk kosong).
1	Tepat satu bagian
1..*	Sedikitnya hanya satu bagian

Pembuatan Class



Candidate Class

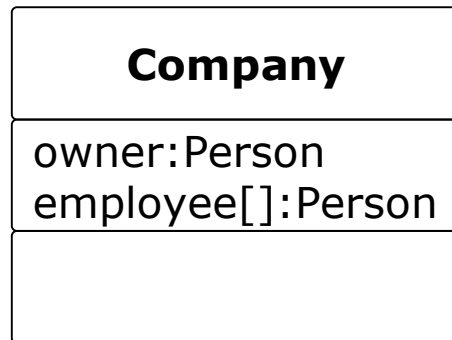
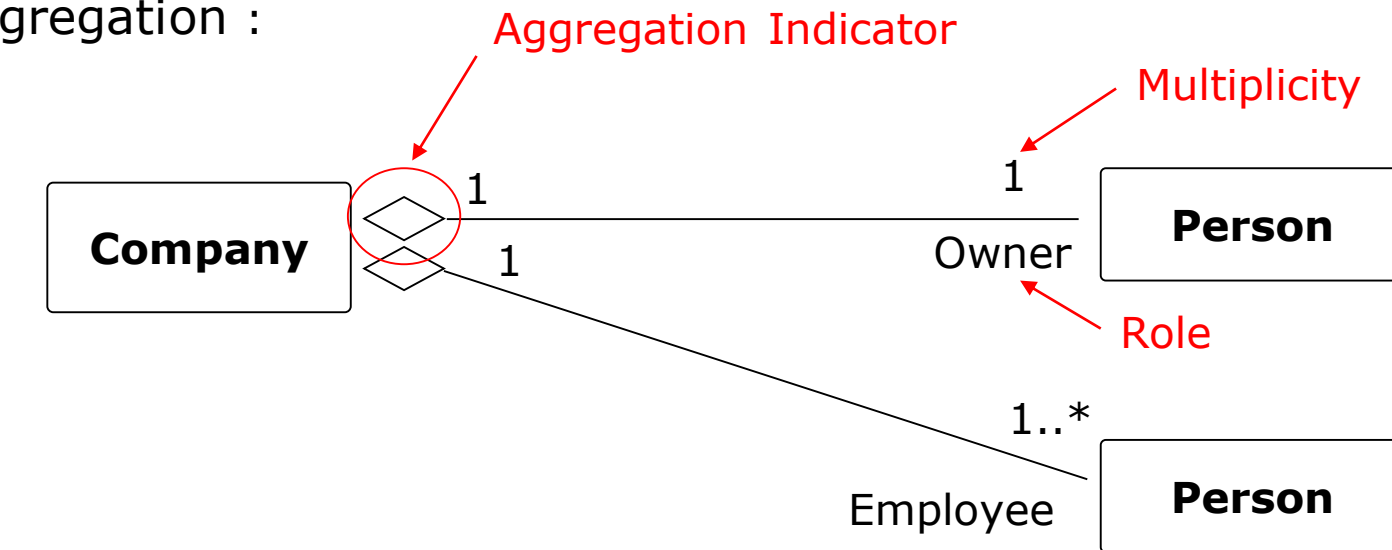
No	Kategori Object	Nama Object
1.	Object Fisik	ATM (Mesin), ATM card
2.	Transaksi	Withdrawal, Transfer
3.	Butir yang terlibat pada transaksi
4.	Peran	User(Pemegang ATMCard) Bank
5.	Piranti	ATM Komputer
6.	Proses	Withdrawal Update
7.	Katalog	Daftar Account





Contoh Class Diagram

Aggregation :

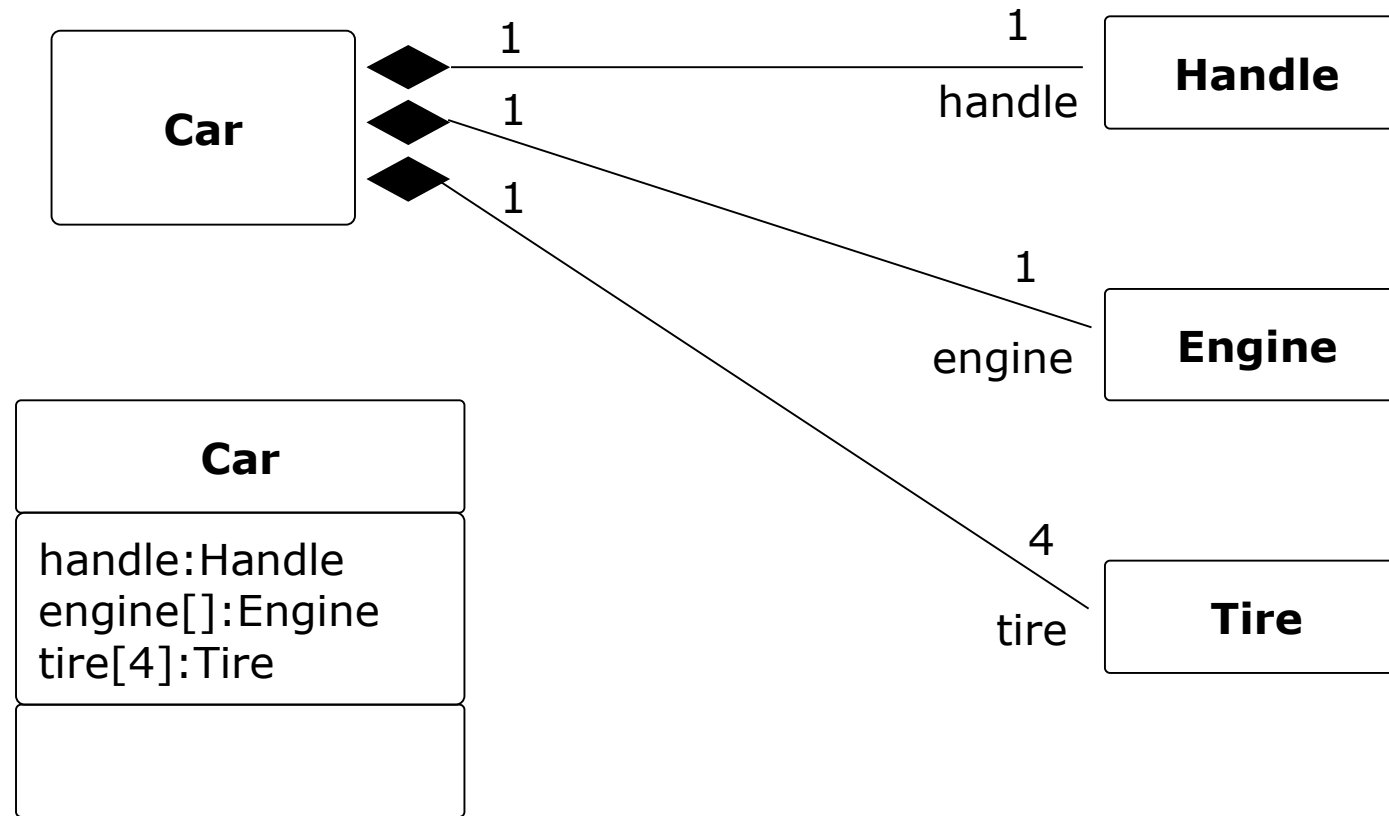


Detail of the class



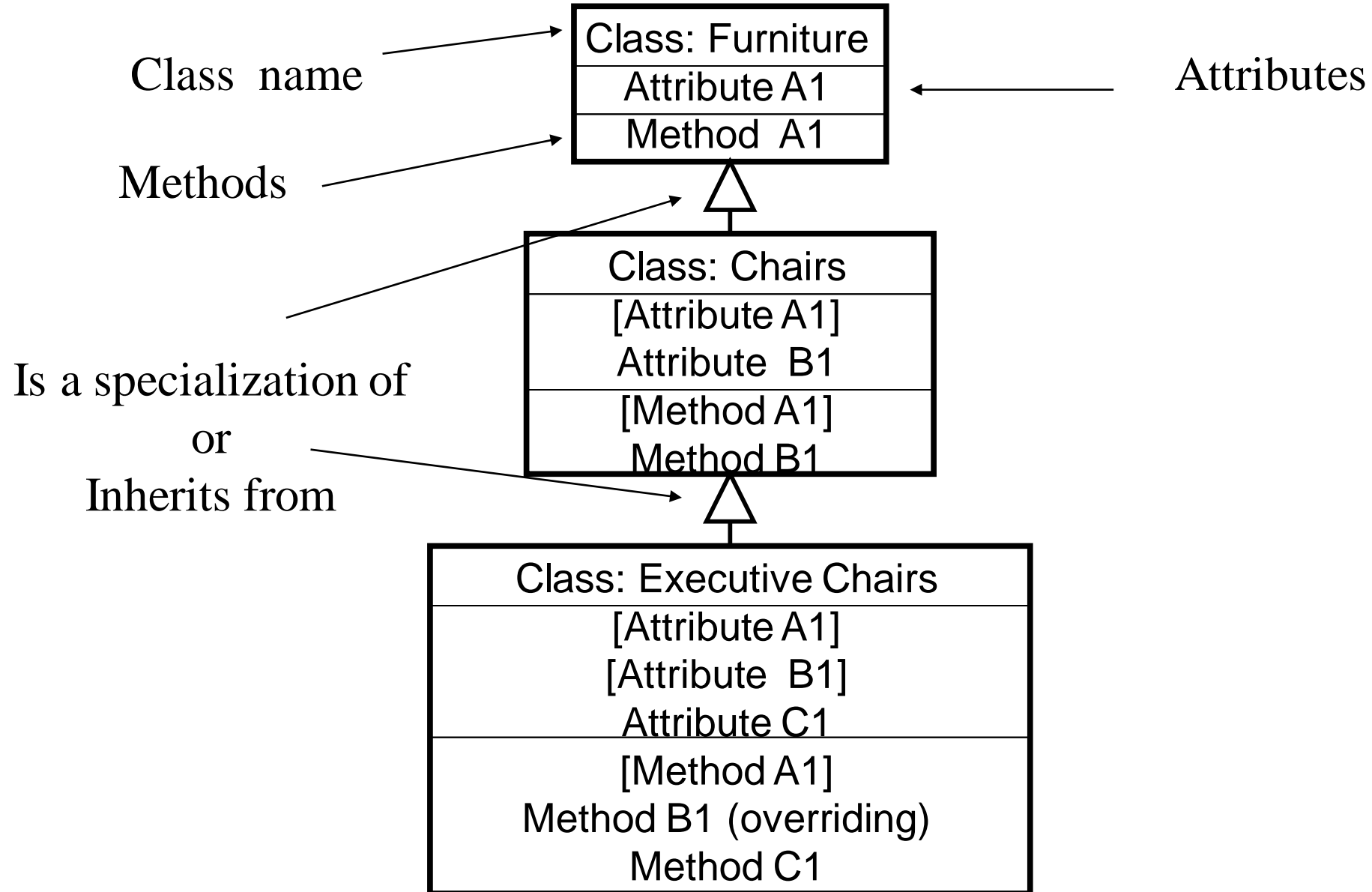
Contoh Class Diagram

Composition :

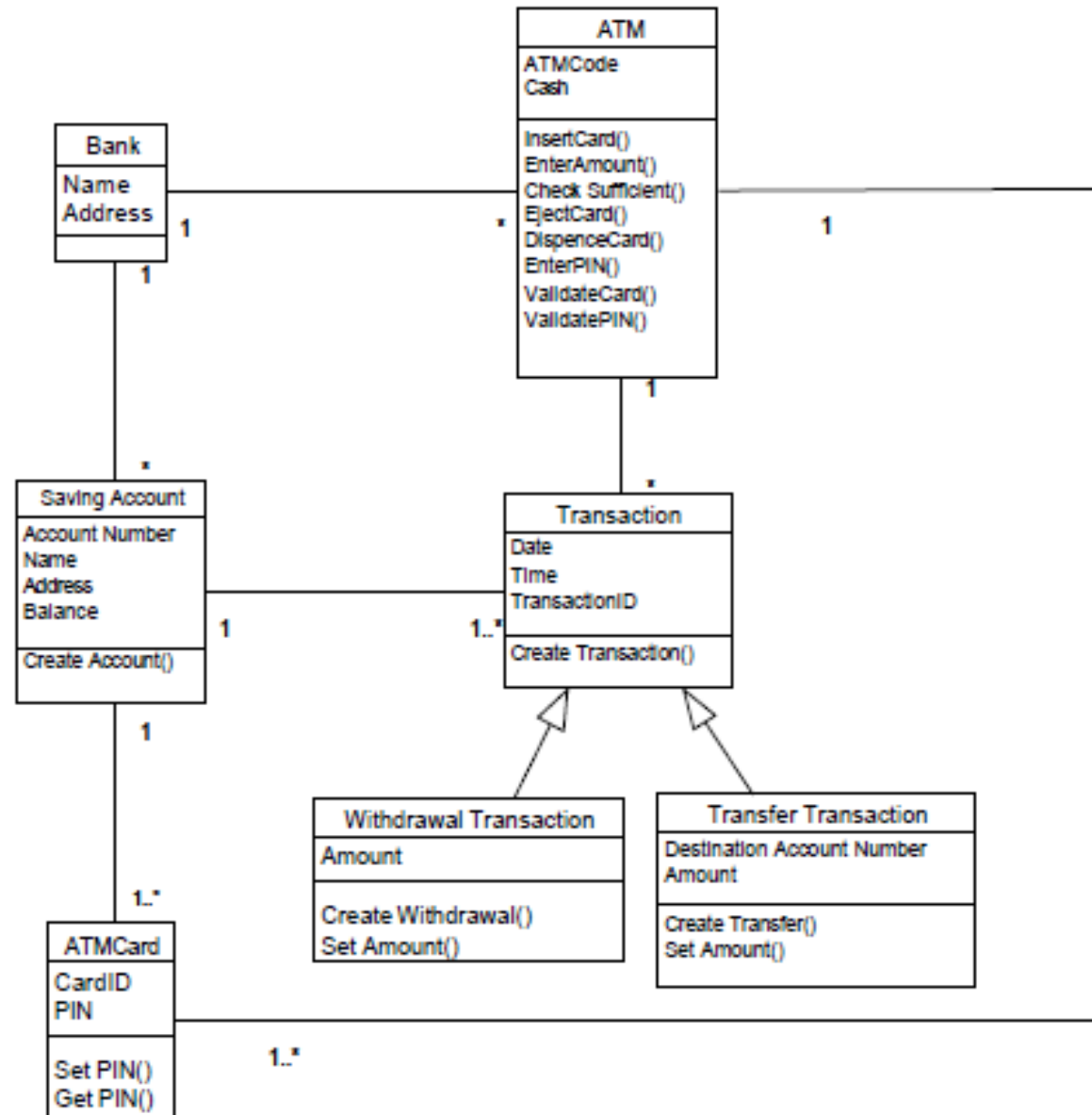


Detail of the class

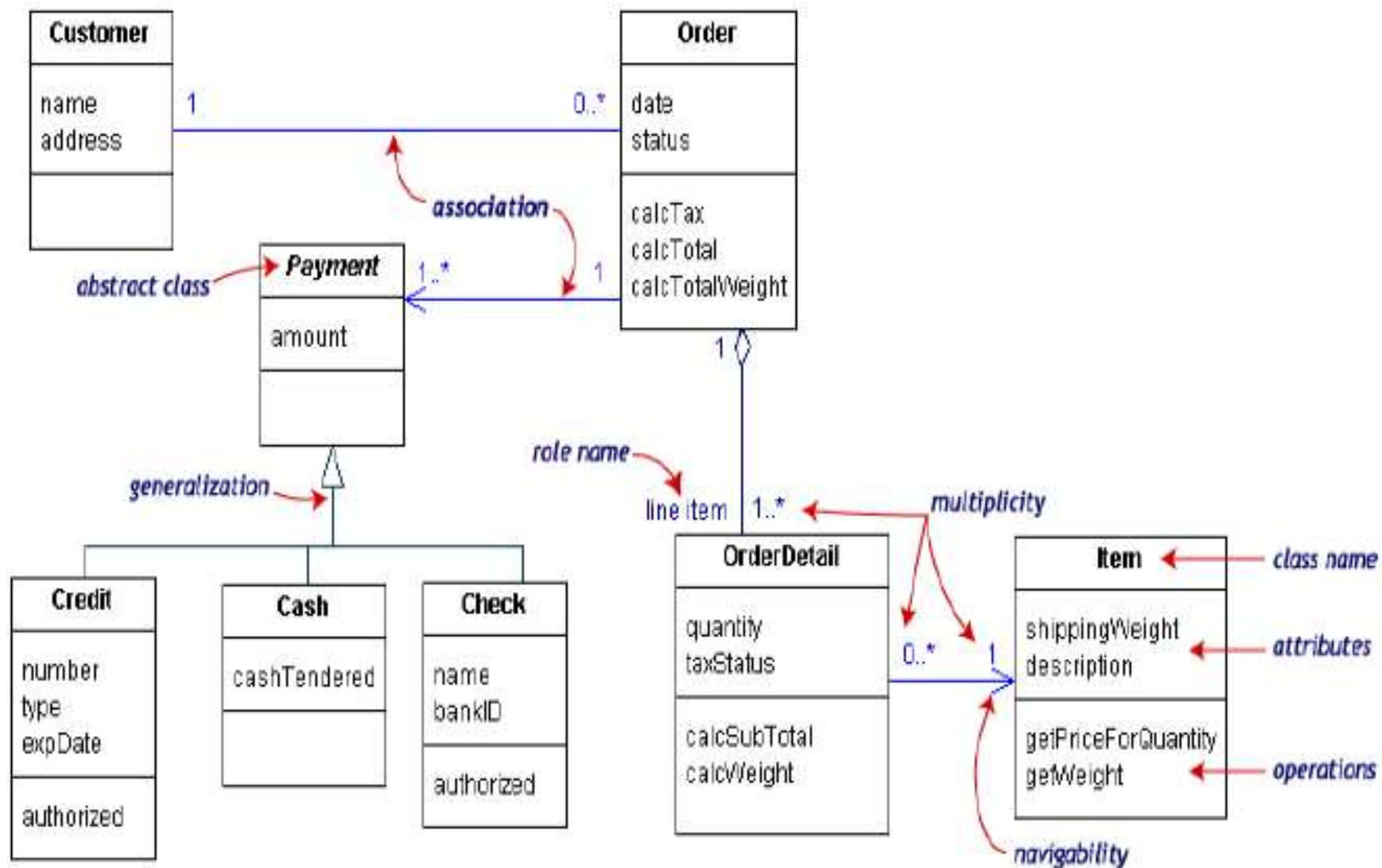
Class Inheritance & Specialization



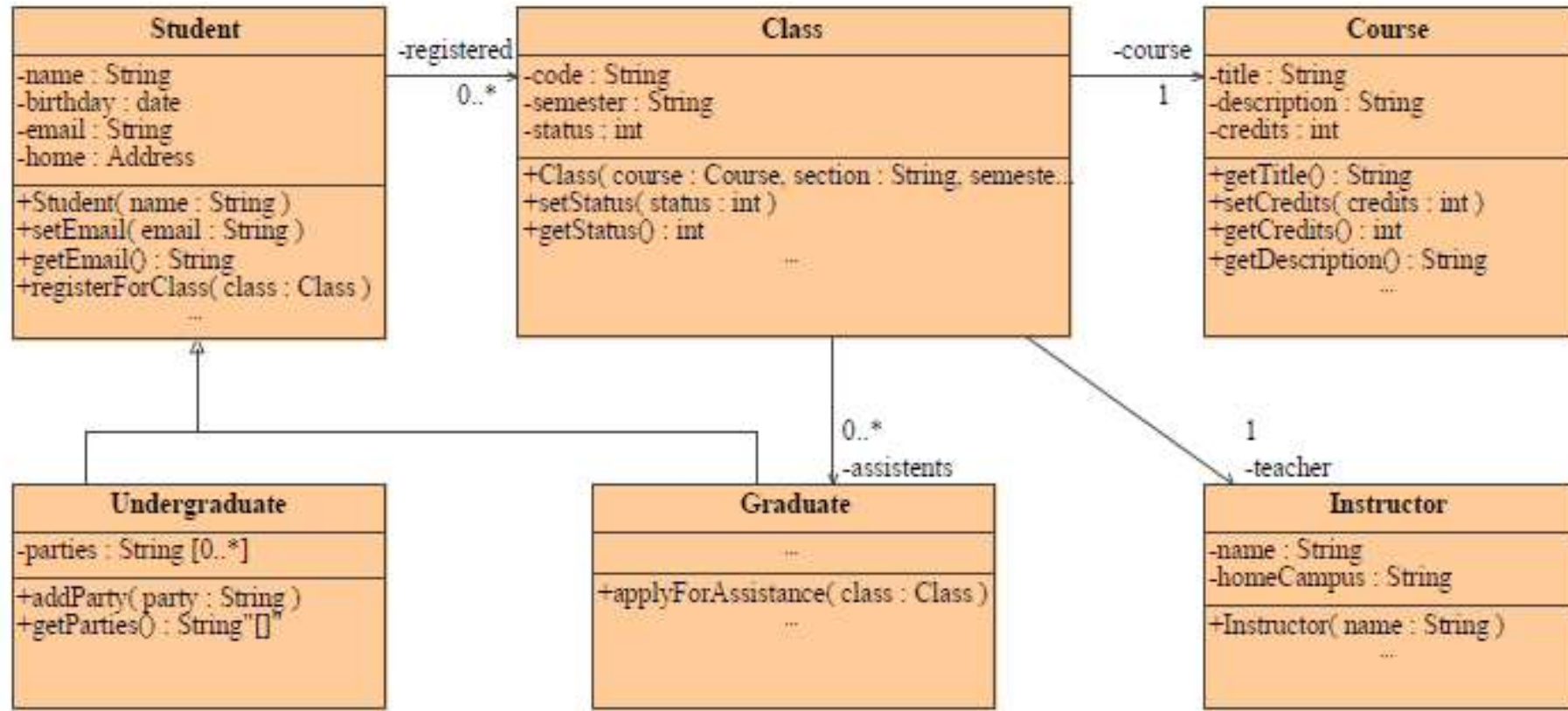
Contoh Class Diagram (Studi Kasus ATM)



Contoh Class Diagram (Pembelian Barang)



Contoh Class Diagram (Sistem Akademik)





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).

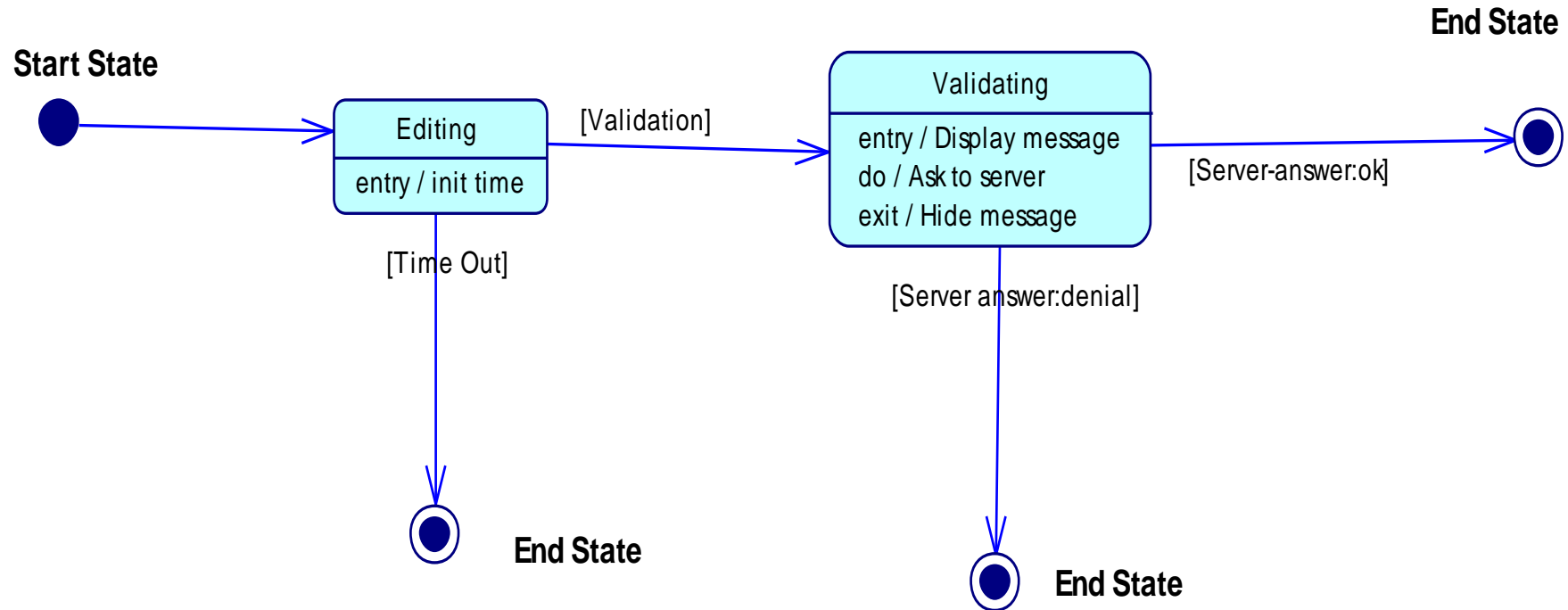


StateChart Diagram

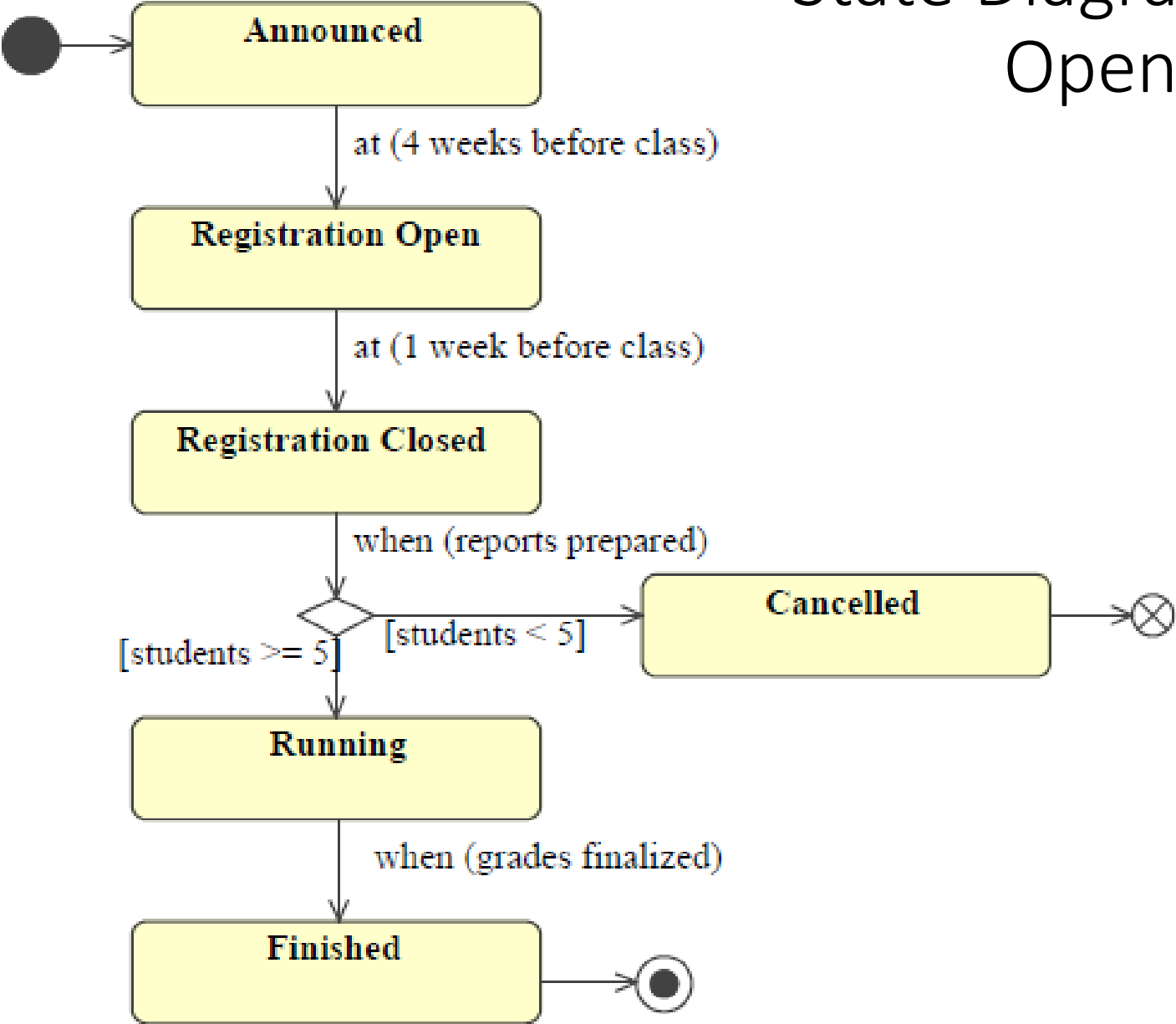
- Dalam UML, state digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya saat itu.
- Transisi antar state umumnya memiliki kondisi guard yang merupakan syarat terjadinya transisi yang bersangkutan, dituliskan dalam kurung siku.
- Action yang dilakukan sebagai akibat 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



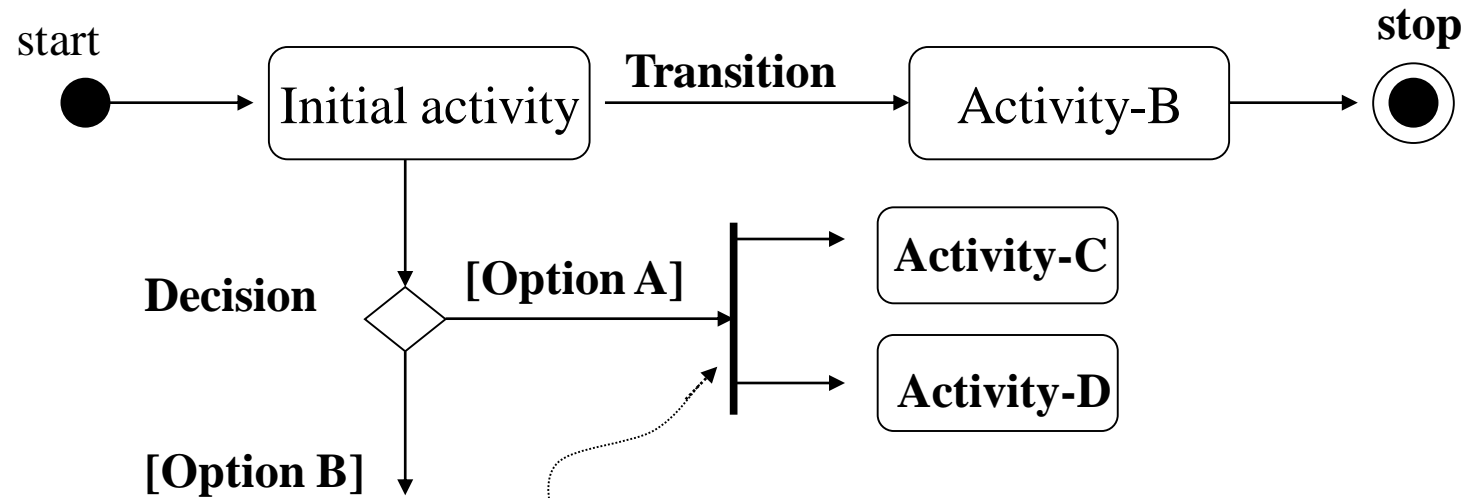


Activity Diagram

- 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.

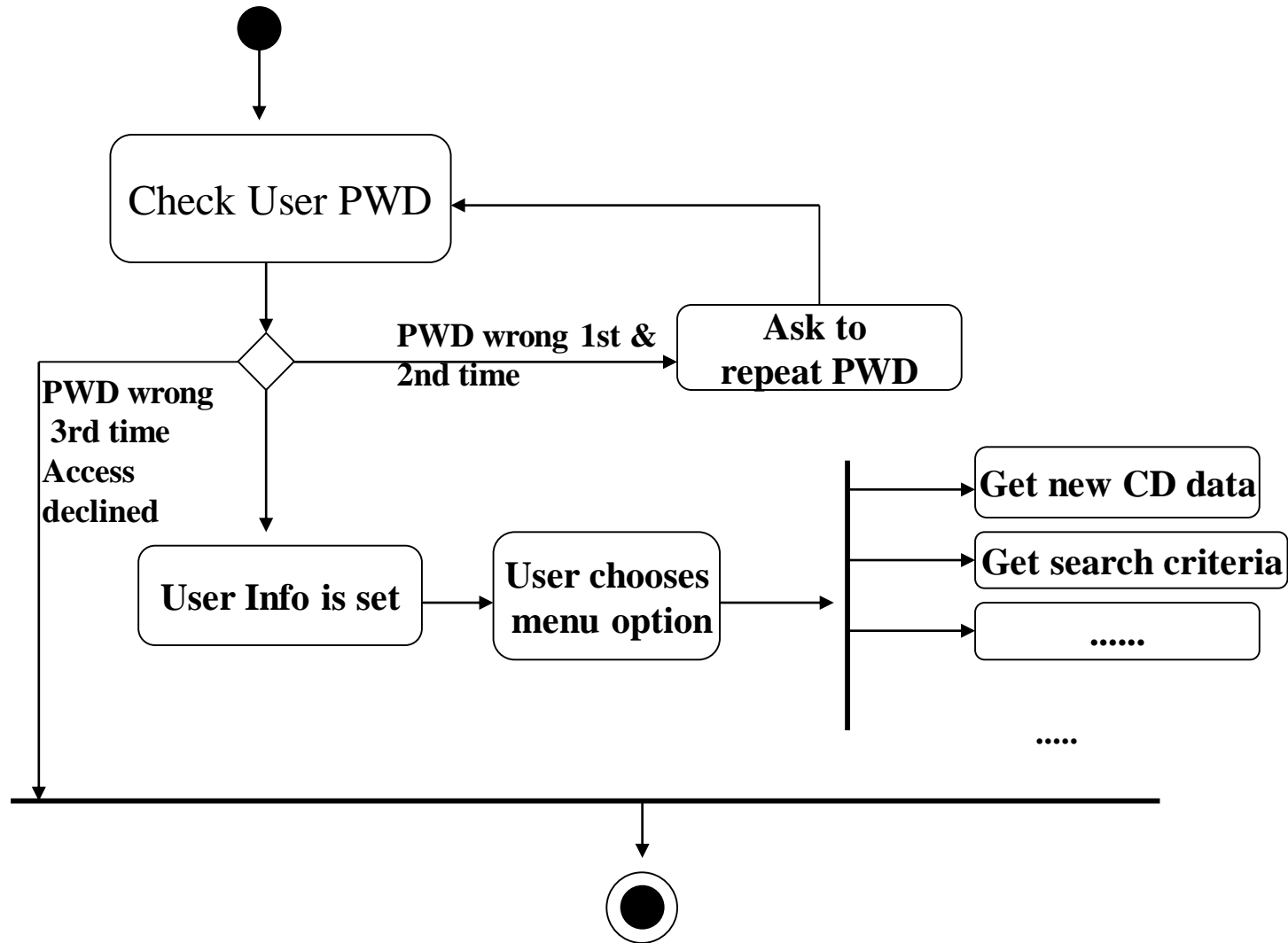


Activity Diagrams Format

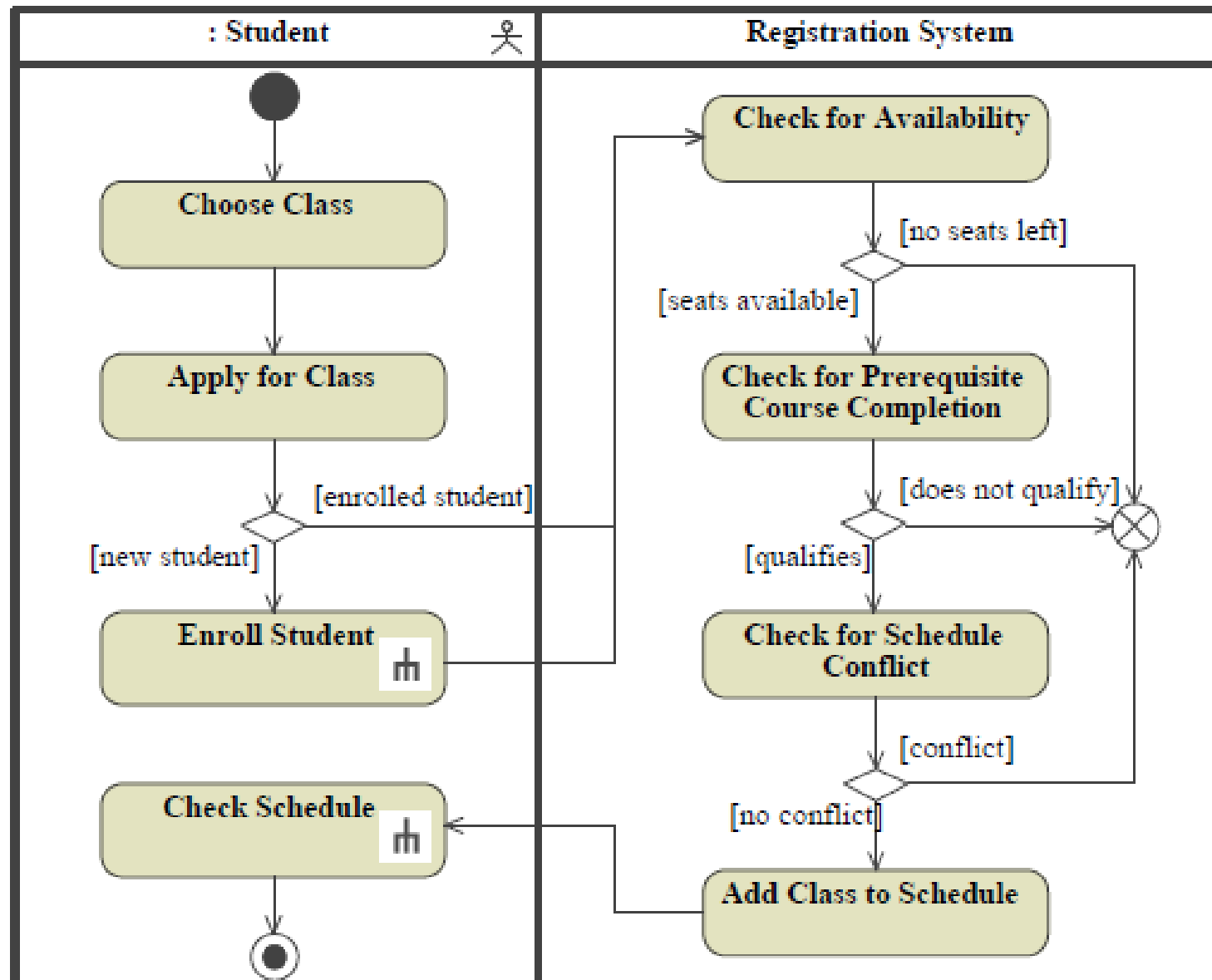


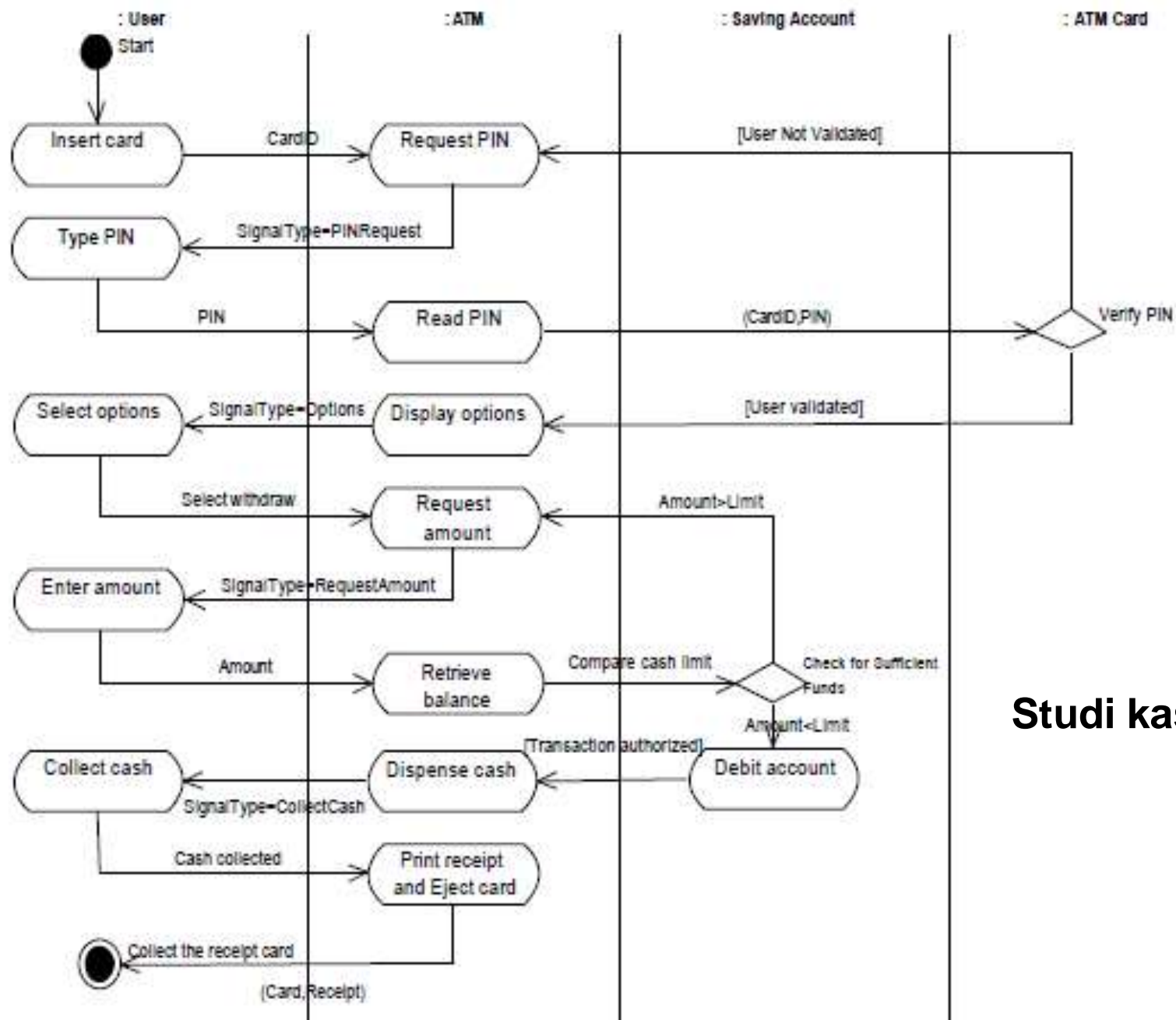
The bar shows that one activity leads to several that occur in parallel or in an unpredictable order.

Activity Diagrams Example

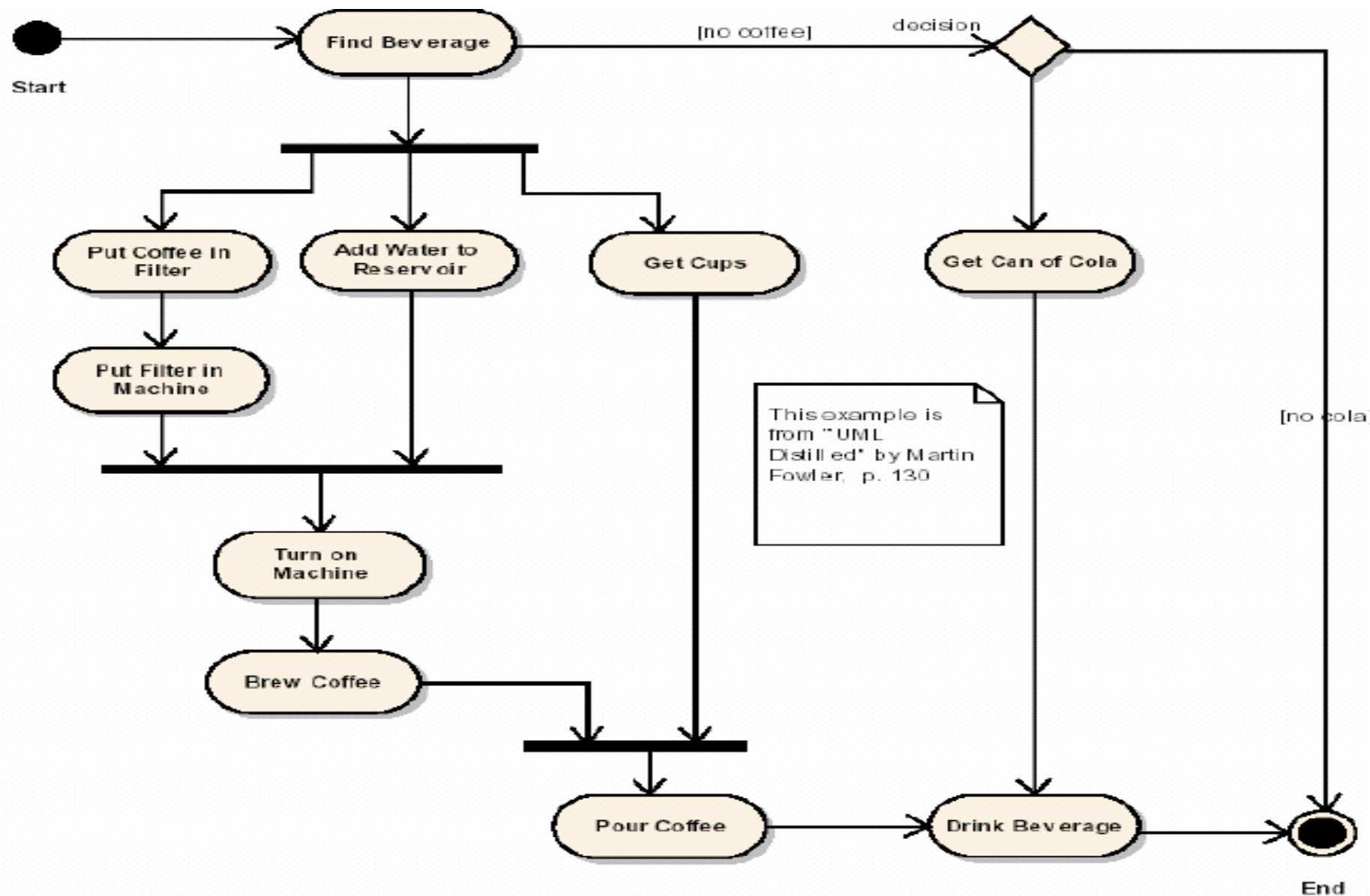


Activity Diagrams Example





Studi kasus ATM





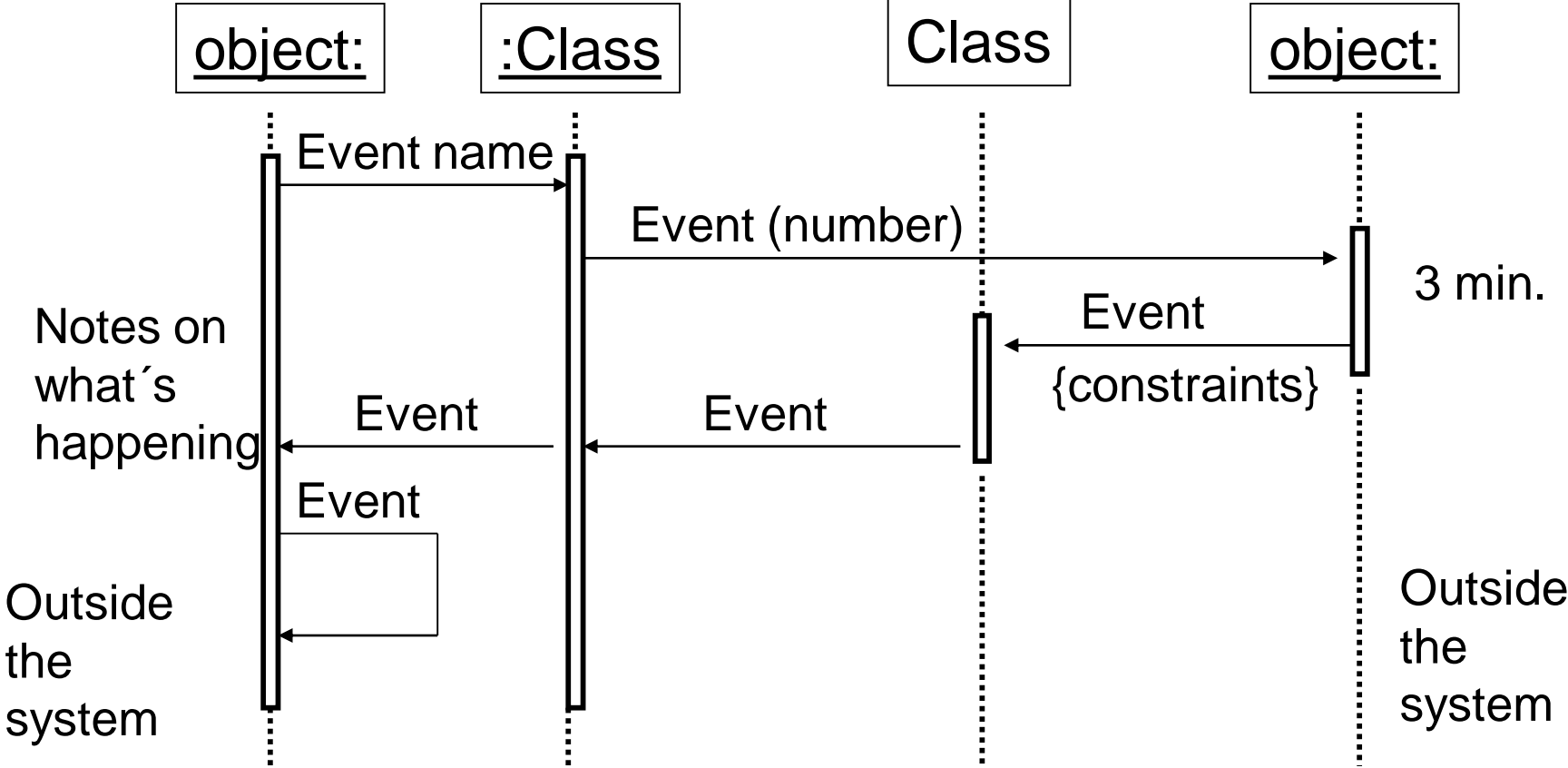
Sequence Diagram

- 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).
- Sequence diagram biasa digunakan untuk menggambarkan skenario atau rangkaian langkah-langkah yang dilakukan sebagai respons dari sebuah event untuk menghasilkan output tertentu. Diawali dari apa yang men-trigger aktivitas tersebut, proses dan perubahan apa saja yang terjadi secara internal dan output apa yang dihasilkan

Sequence diagram notation (Reihenfolge-Diagramm)

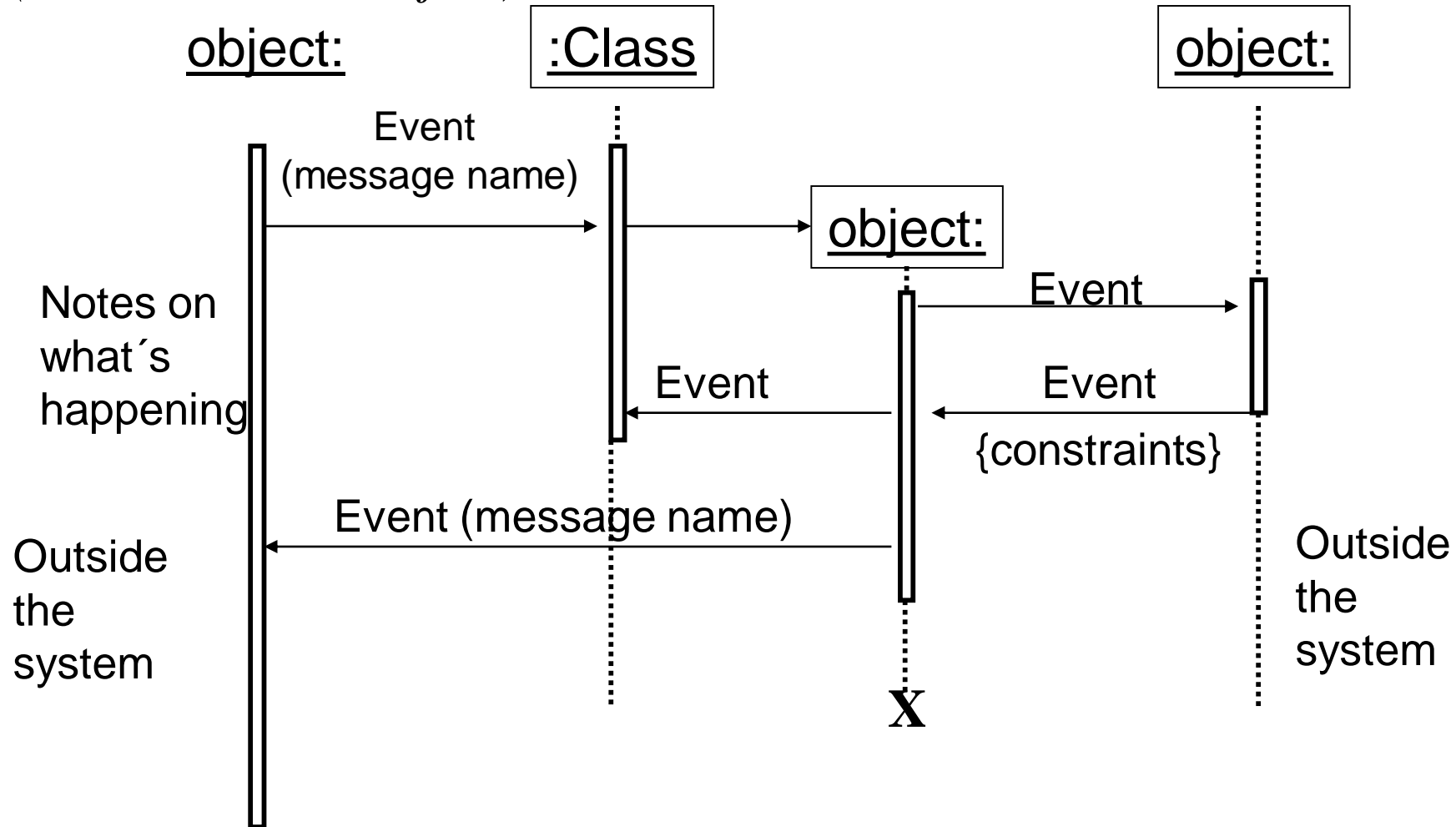


(This is not an object)

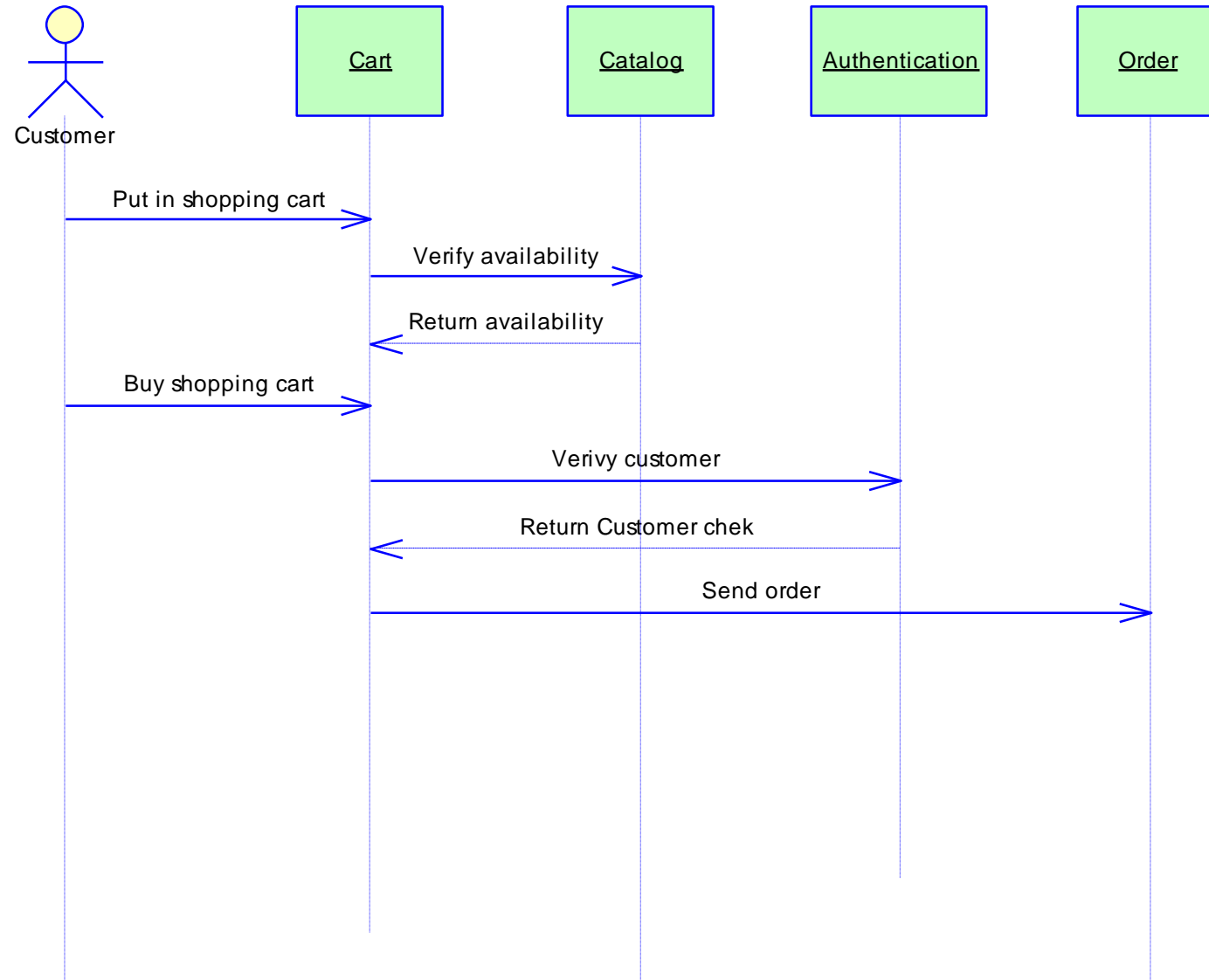


Sequence diagram notation (2)

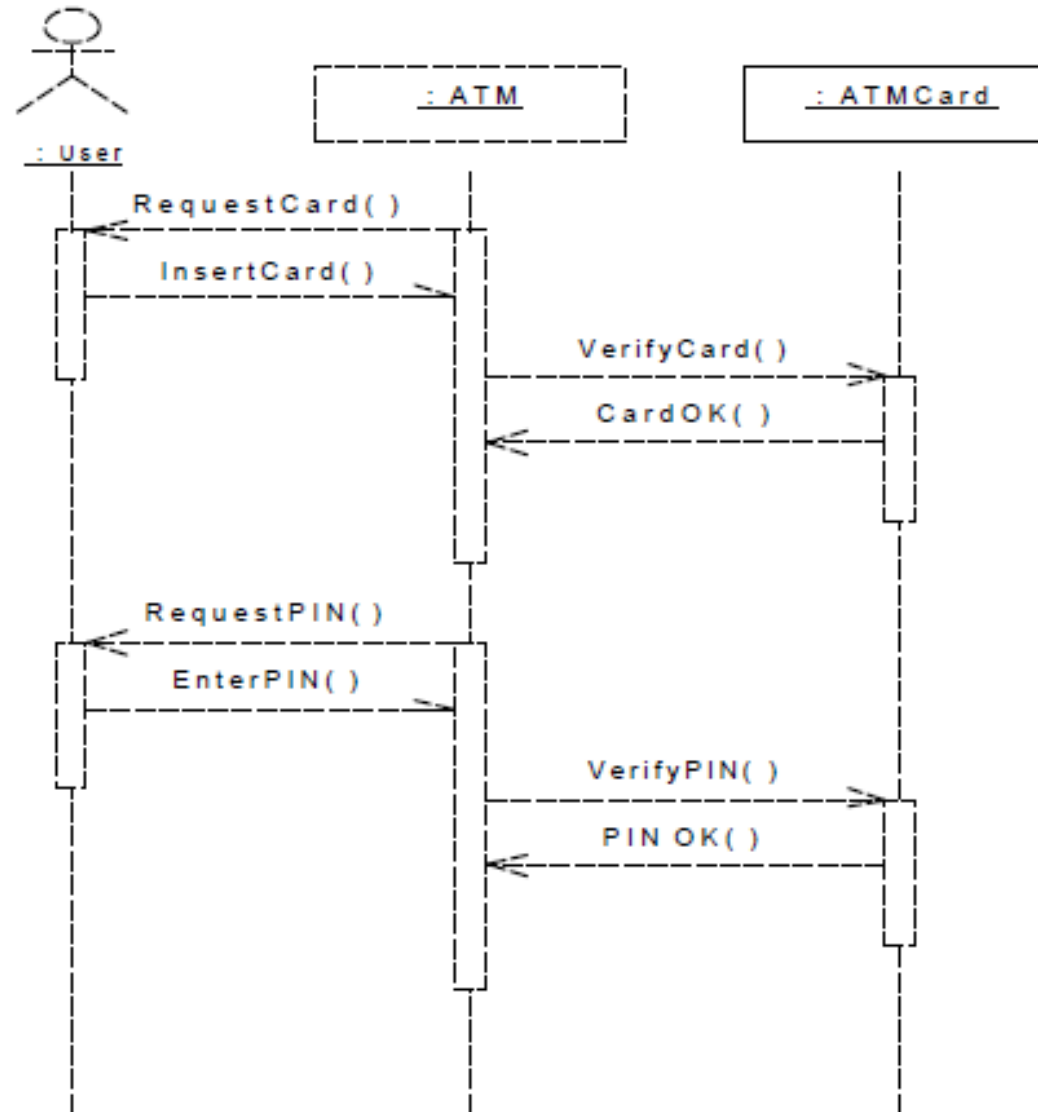
(This isn't a SW object)



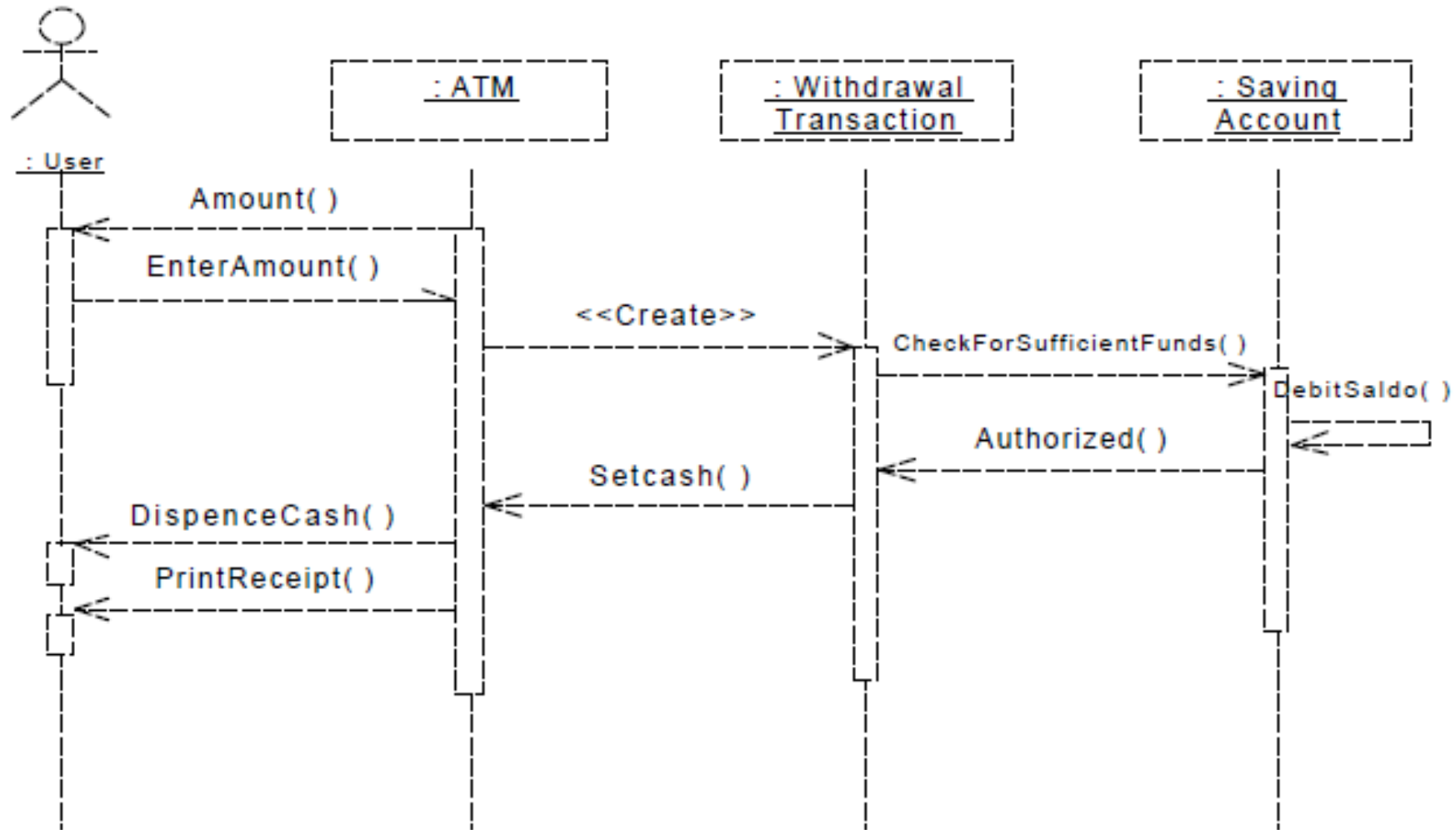
Proses pemesanan buku



Sequence Diagram for Authenticate User's ATM



Sequence Diagram for Withdrawal Transaction in ATM





Tool Yang Mendukung UML

- Rational Rose (www.rational.com)
- Together (www.togethersoft.com)
- Object Domain (www.objectdomain.com)
- Jvision (www.object-insight.com)
- Objecteering (www.objecteering.com)
- MagicDraw (www.nomagic.com/magicdrawuml)
- Visual Object Modeller (www.visualobject.com)
- Edraw UML Diagram