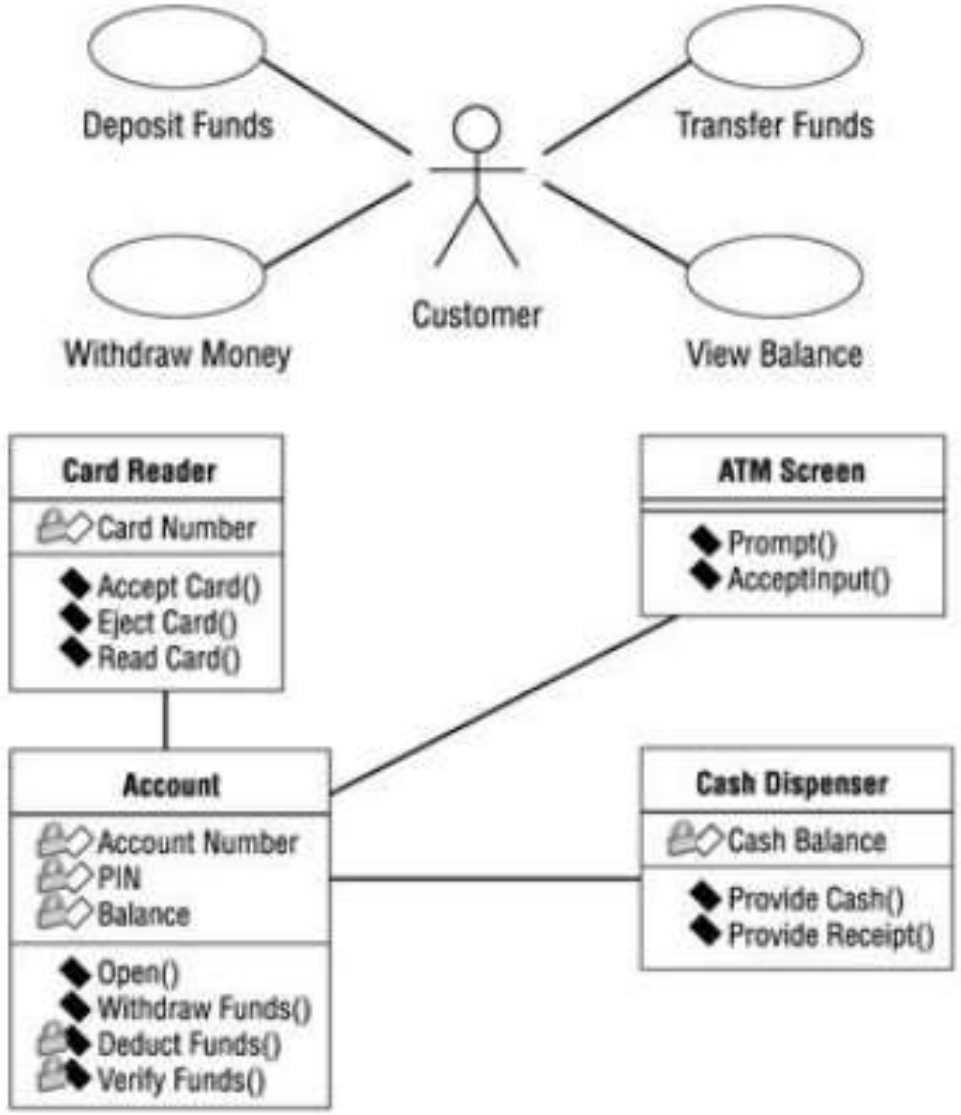
UNIFIED MODELLING LANGUAGE

Oleh : Rahmat Robi Waliyansyah, M.Kom.

Pengertian UML

- Unified Modelling Language (UML) adalah sebuah bahasa yang telah menjadi standar dalam industri untuk visualisasi, merancang dan mendokumentasikan sistem piranti lunak.
- UML menawarkan sebuah standar untuk merancang model sebuah sistem.

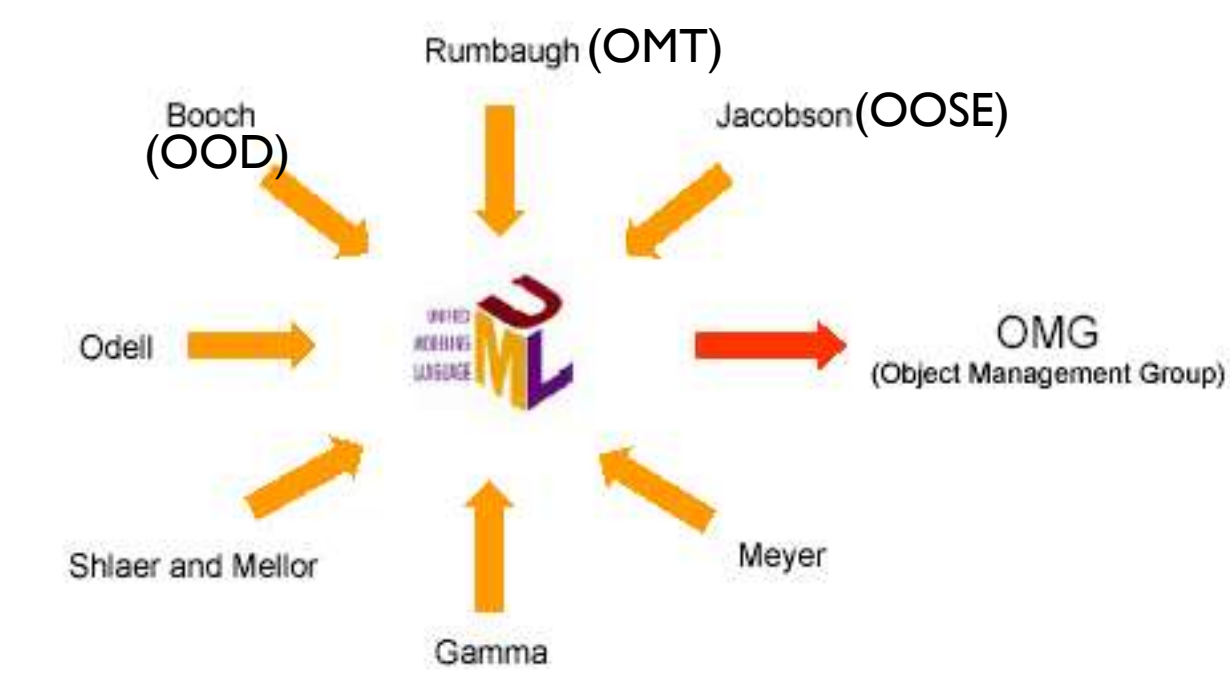
Contoh notasi UML



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).

SEJARAH UML



Th 1994. Munculnya tokoh pelopor (Booch,Rumbough dan Jacobson)

Th 1995. Direlease draft pertama UML (versi 0.8)

Th 1996. Pengkoordinasian oleh Object Management Group (OMG)

Th 1997. Munculnya UML (versi 1.1)

Th 1999. Penyusunan 3 buku UML oleh (Booch,Rumbough dan Jacobson)

Th 1999. UML menjadi standart bahasa permodelan berorientasi objek

Th 2003. Direlease UML versi 1.5

Th 2004 direlease UML Versi 2.0 (masih dalam pengembangan)

UNIFIED MODELLING LANGUAGE

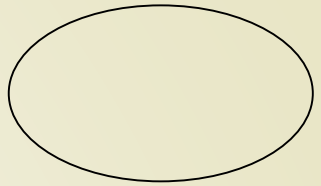
UML mendefinisikan diagram-diagram berikut ini :

- Use case diagram
- Class diagram
- Behaviour diagram :
  -- Statechart diagram
  -- Activity diagram
- Interaction diagram :
  -- Sequence diagram
  -- Collaboration diagram
- Component diagram
- Deployment diagram

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



- Use case dibuat berdasar keperluan actor, merupakan "apa" yang dikerjakan system, bukan "bagaimana" system mengerjakannya
- Use case diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan actor.
- Use case dinotasikan dengan gambar (horizontal ellipse).
- Use case biasanya menggunakan kata kerja.
- Nama use case boleh terdiri dari beberapa kata. dan tidak boleh ada 2 use case yang memiliki nama yang sama

ACTOR

- Actor menggambarkan orang, system atau external entitas / stakeholder yang menyediakan atau menerima informasi dari system
- Actor menggambarkan sebuah tugas/peran dan bukannya posisi sebuah jabatan
- Actor memberi input atau menerima informasi dari system
- Actor biasanya menggunakan Kata benda
- Tidak boleh ada komunikasi langsung antar actor
- Indikasi <<system>> untuk sebuah actor yang merupakan sebuah system
- Adanya actor bernama "Time" yang mengindikasikan scheduled events (suatu kejadian yang terjadi secara periodik/bulanan)
- Letakkan actor utama anda pada pojok kiri atas dari diagram

ASSOCIATION

- Associations bukan menggambarkan aliran data/informasi
- Associations digunakan untuk menggambarkan bagaimana actor terlibat dalam use case
- Ada 4 jenis relasi yang bisa timbul pada use case diagram
    1. Association antara actor dan use case
    2. Association antara use case
    3. Generalization/Inheritance antara use case
    4. Generalization/Inheritance antara actors

ASSOCIATION ANTARA ACTOR DAN USE CASE

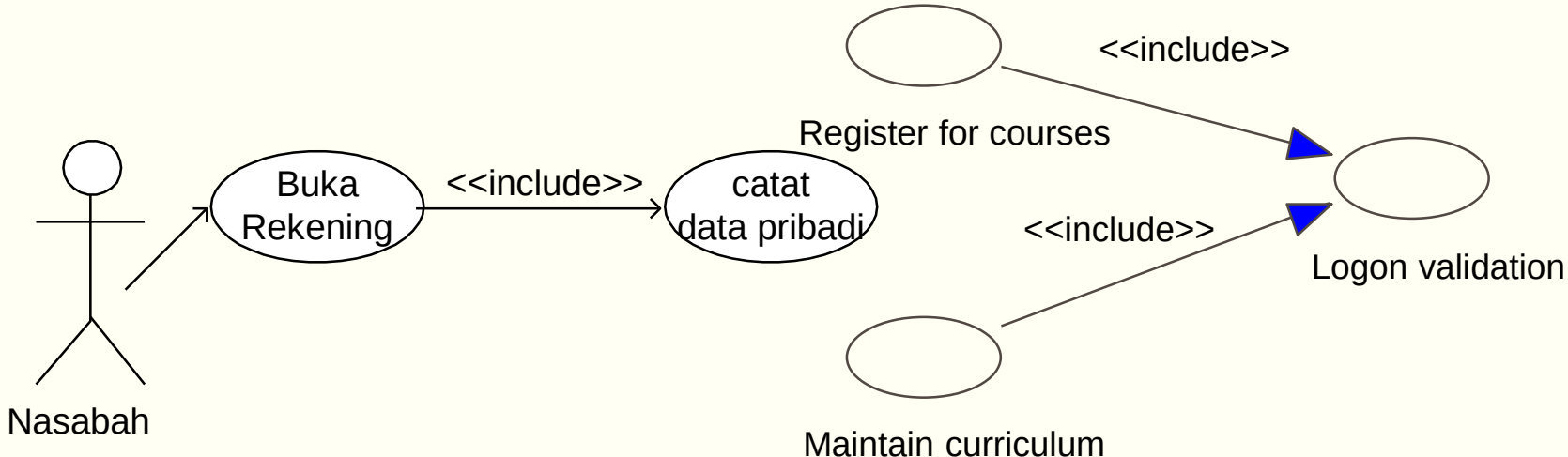
- Ujung panah pada association antara actor dan use case mengindikasikan ***siapa/apa*** yang meminta interaksi dan bukannya mengindikasikan aliran data
- Sebaiknya gunakan **Garis tanpa panah** untuk association antara actor dan use case

---

- Association antara actor dan use case yang menggunakan **panah terbuka** untuk mengindikasikan bila actor berinteraksi secara ***pasif*** dengan system anda

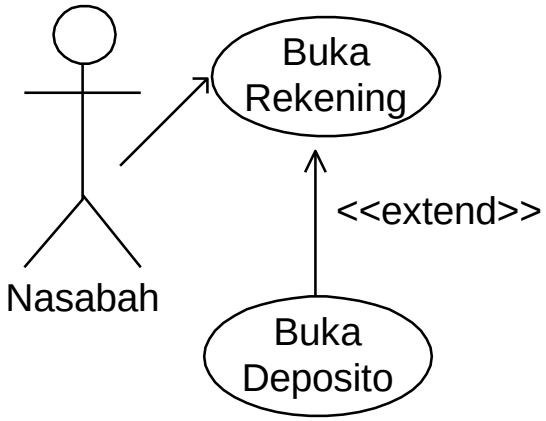
ASSOCIATION ANTARA USE CASE

- <<include>> termasuk didalam use case lain (required) / (diharuskan)
  - Pemanggilan use case oleh use case lain, contohnya adalah pemanggilan sebuah fungsi program
  - Tanda panah terbuka harus terarah ke sub use case
  - Gambarkan association include secara horizontal



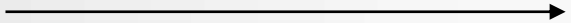
Association Antara Use Case (Lanjut)

- <<extend>> perluasan dari use case lain jika kondisi atau syarat terpenuhi
  - Kurangi penggunaan association Extend ini, terlalu banyak pemakaian association ini membuat diagram sulit dipahami.
  - Tanda panah terbuka harus terarah ke parent/base use case
  - Gambarkan association extend secara vertical

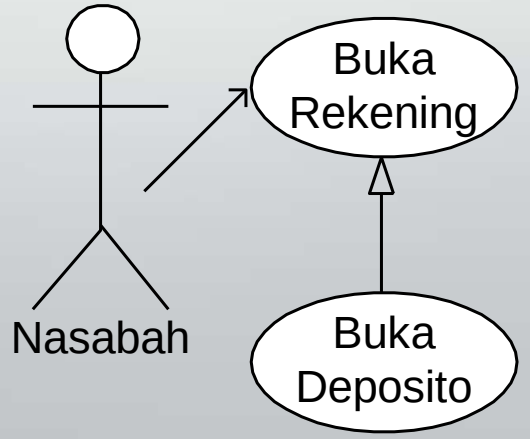


Generalization/Inheritance Antara Use Case

- Generalization/inheritance digambarkan dengan sebuah garis berpanah tertutup pada salah satu ujungnya yang menunjukkan lebih umum

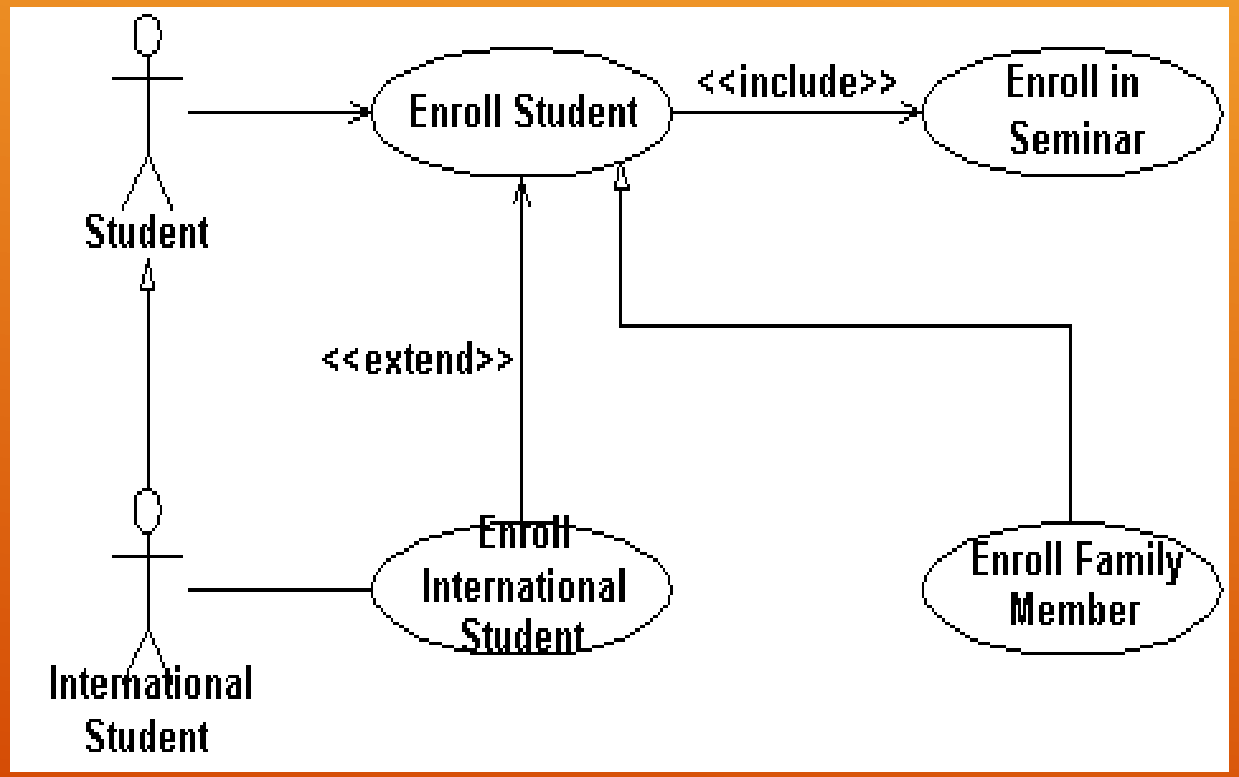


- Gambarkan generalization/inheritance antara use case secara vertical dengan inheriting use case dibawah base/parent use case
- Generalization/inheritance dipakai ketika ada sebuah keadaan yang lain sendiri/perlakuan khusus *(single condition)*



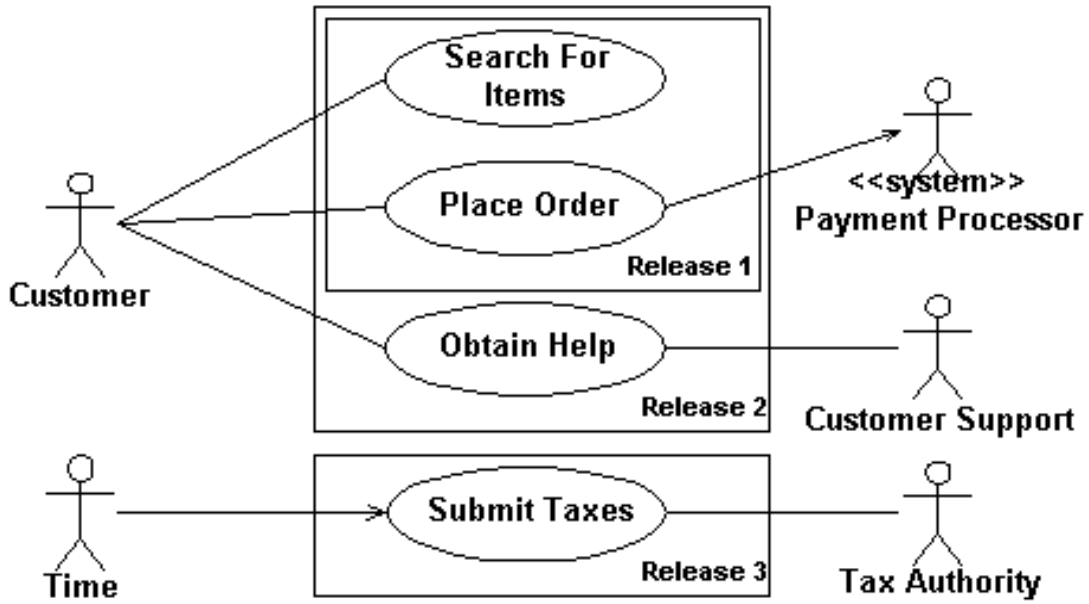
GENERALIZATION/INHERITANCE ANTARA ACTOR

- Gambarkan generalization/inheritance antara actors secara vertical dengan inheriting actor dibawah base/parent use case

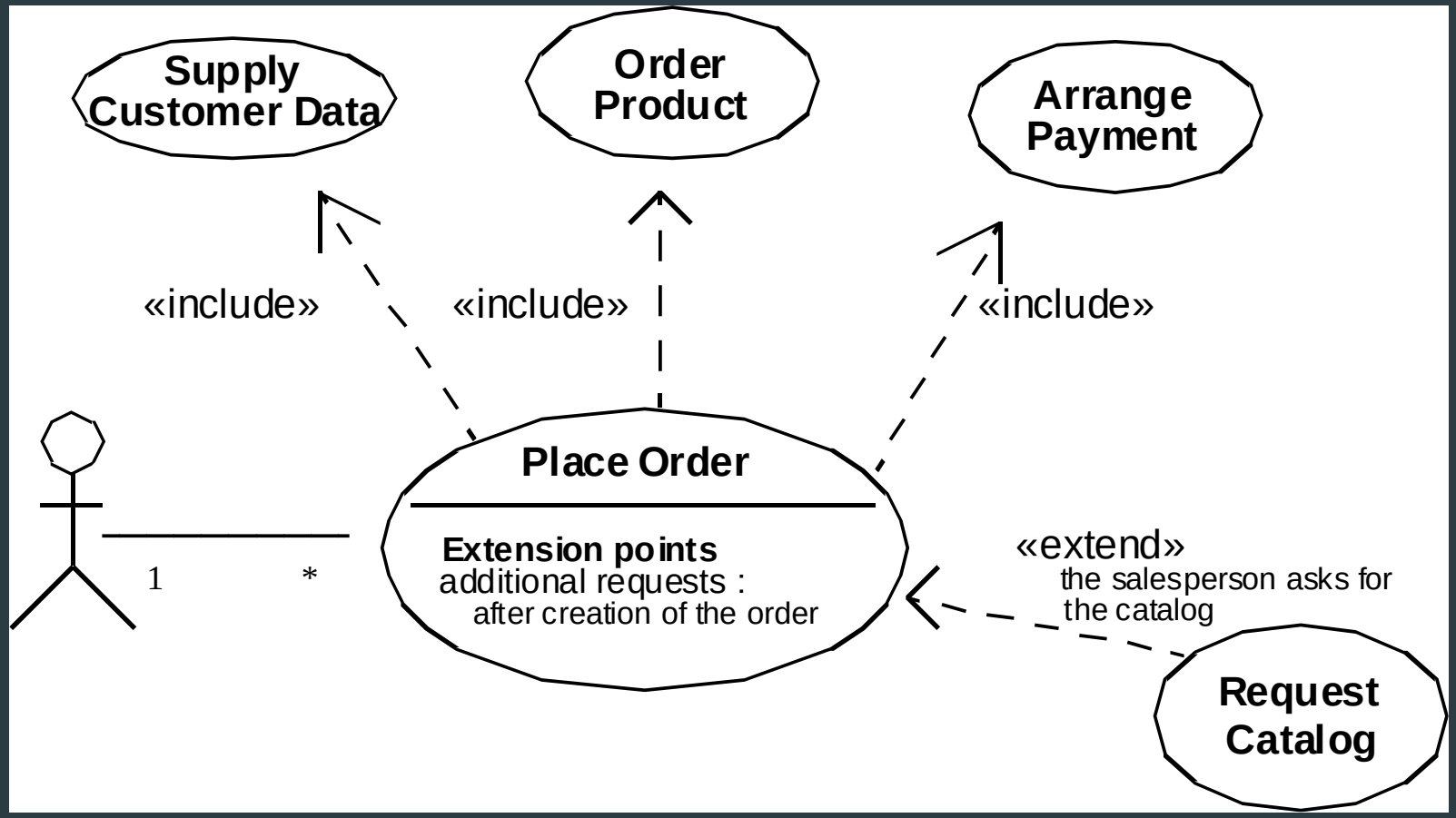


Use Case System Boundary Boxes

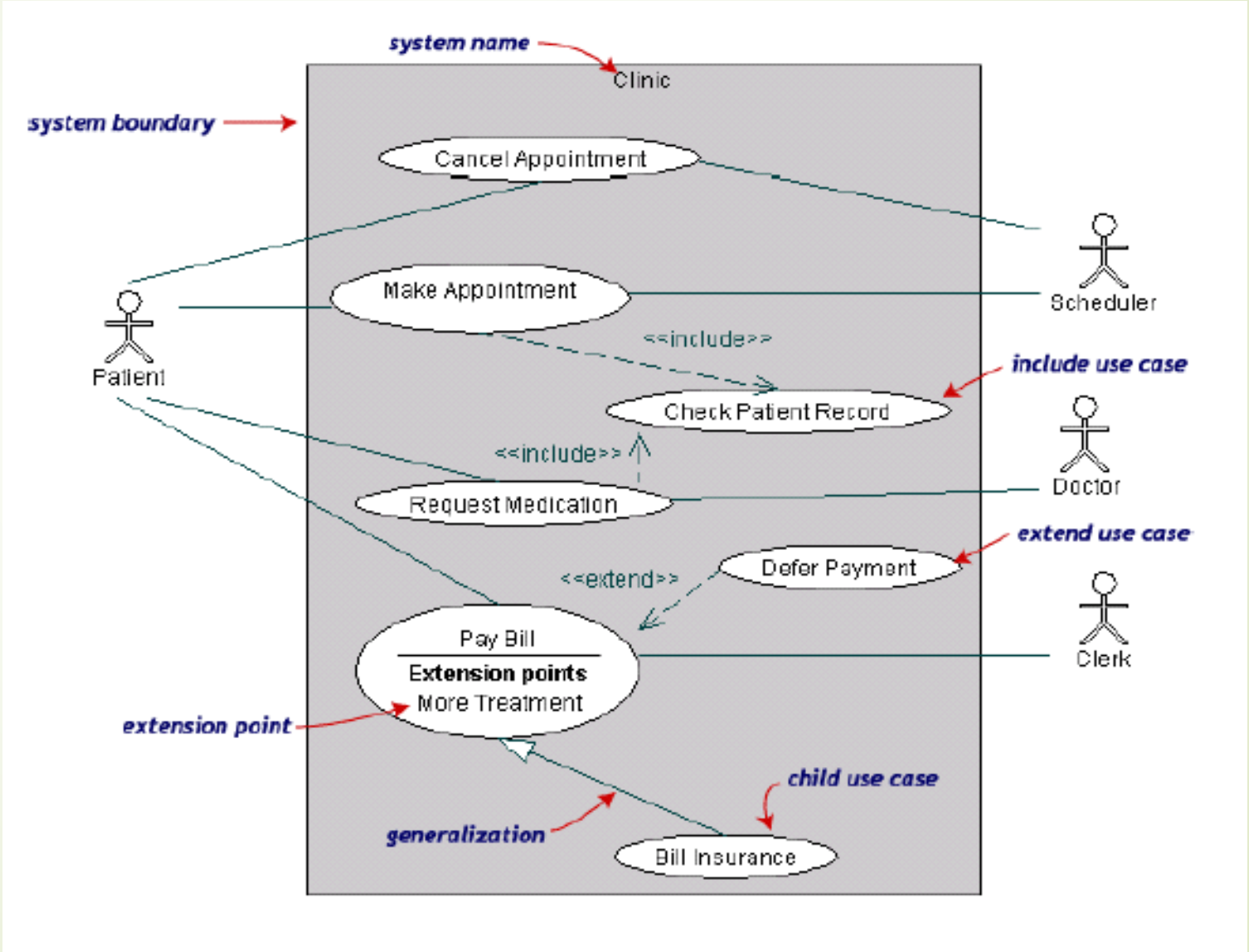
- Digambarkan dengan kotak disekitar use case, untuk menggambarkan jangkauan system anda (scope of of your system).
- Biasanya digunakan apabila memberikan beberapa alternative system yang dapat dijadikan pilihan
- System boundary boxes dalam penggunaannya optional



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 :

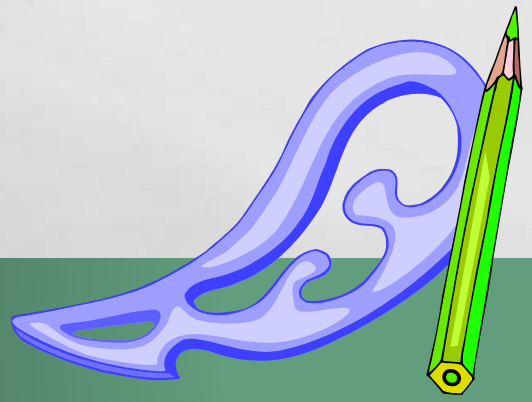
1. Nama (dan stereotype)
2. Atribut
3. Metoda

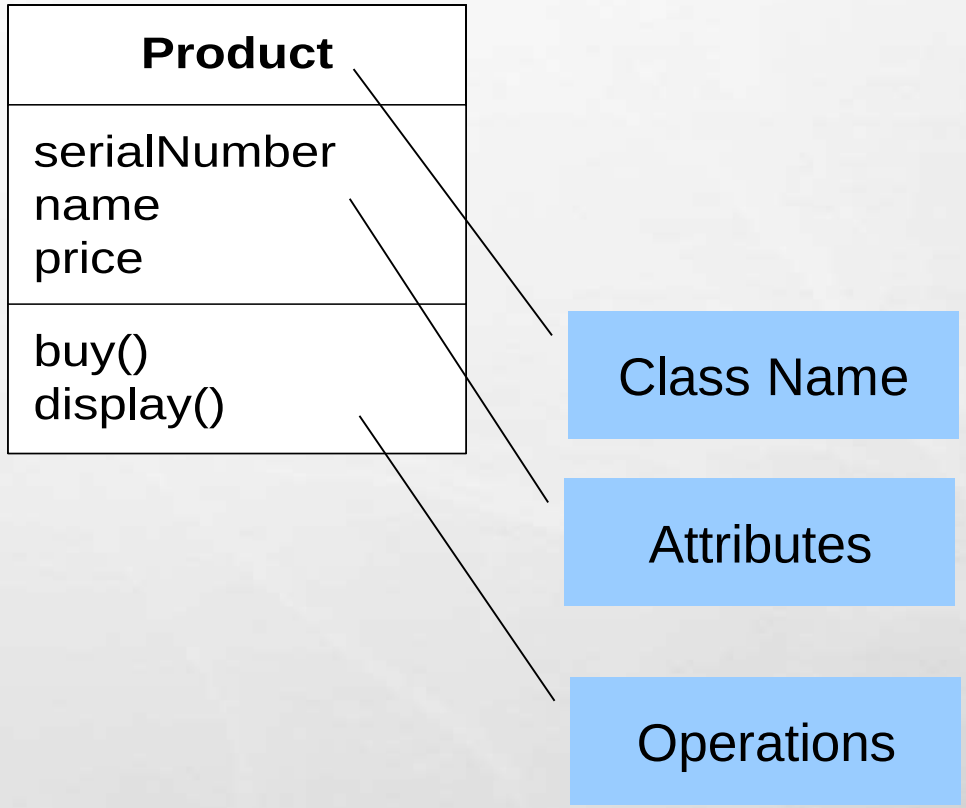
Atribut dan metoda dapat memiliki salah satu sifat berikut :

- Private, tidak dapat dipanggil dari luar class yang bersangkutan
- Protected, hanya dapat dipanggil oleh class yang bersangkutan dan anak-anak yang mewarisinya
- Public, dapat dipanggil oleh siapa saja

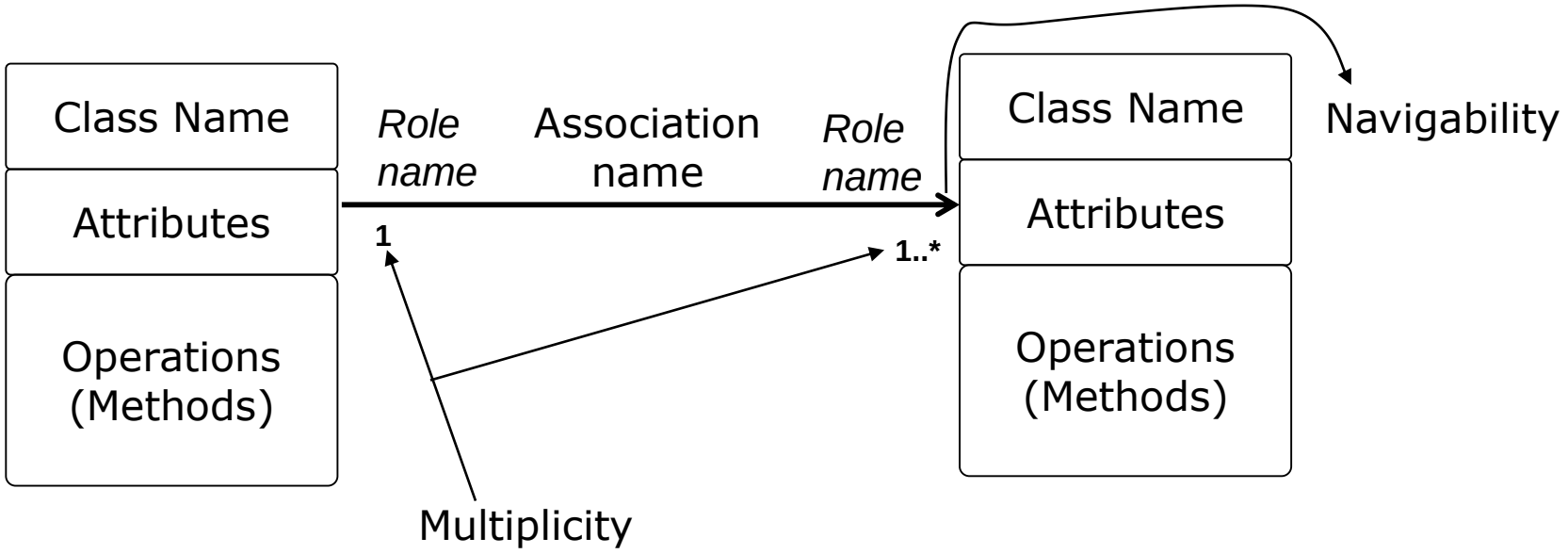
CLASSES

- Kelas A adalah template untuk yang sebenarnya, in-memory, contoh





Class Diagram Format and association:



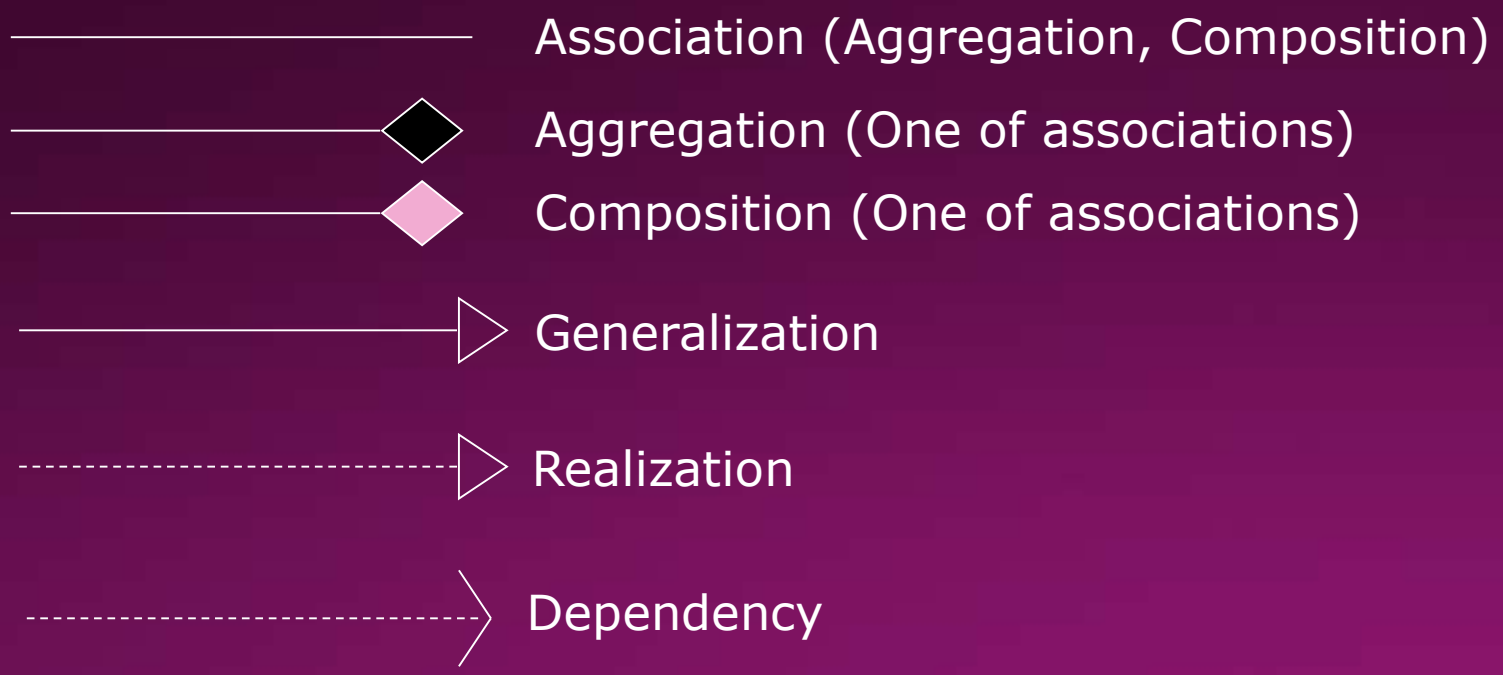
Multiplicity Notation

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 :

- Apakah-a (Generalisasi, Realisasi: Warisan)
- Memiliki-a (Association)
- Lainnya (Association, Ketergantungan)



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'.

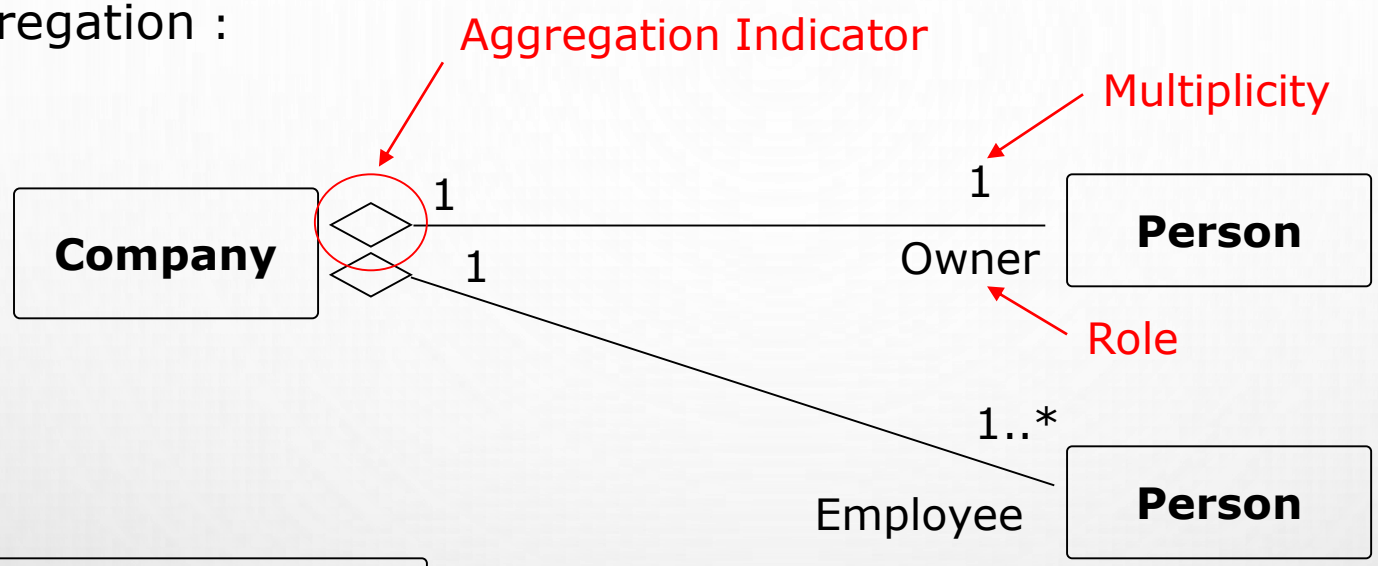
Tabel di bawah mengenai *multiplicity* yang sering digunakan :

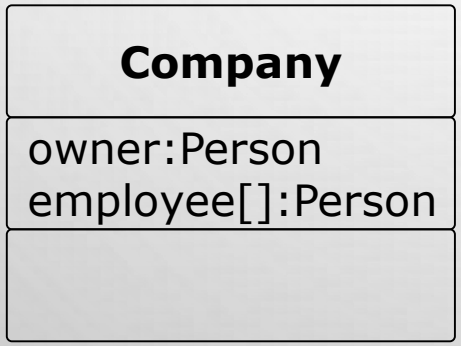
Tabel *Multiplicity*

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

CONTOH CLASS DIAGRAM

Aggregation :

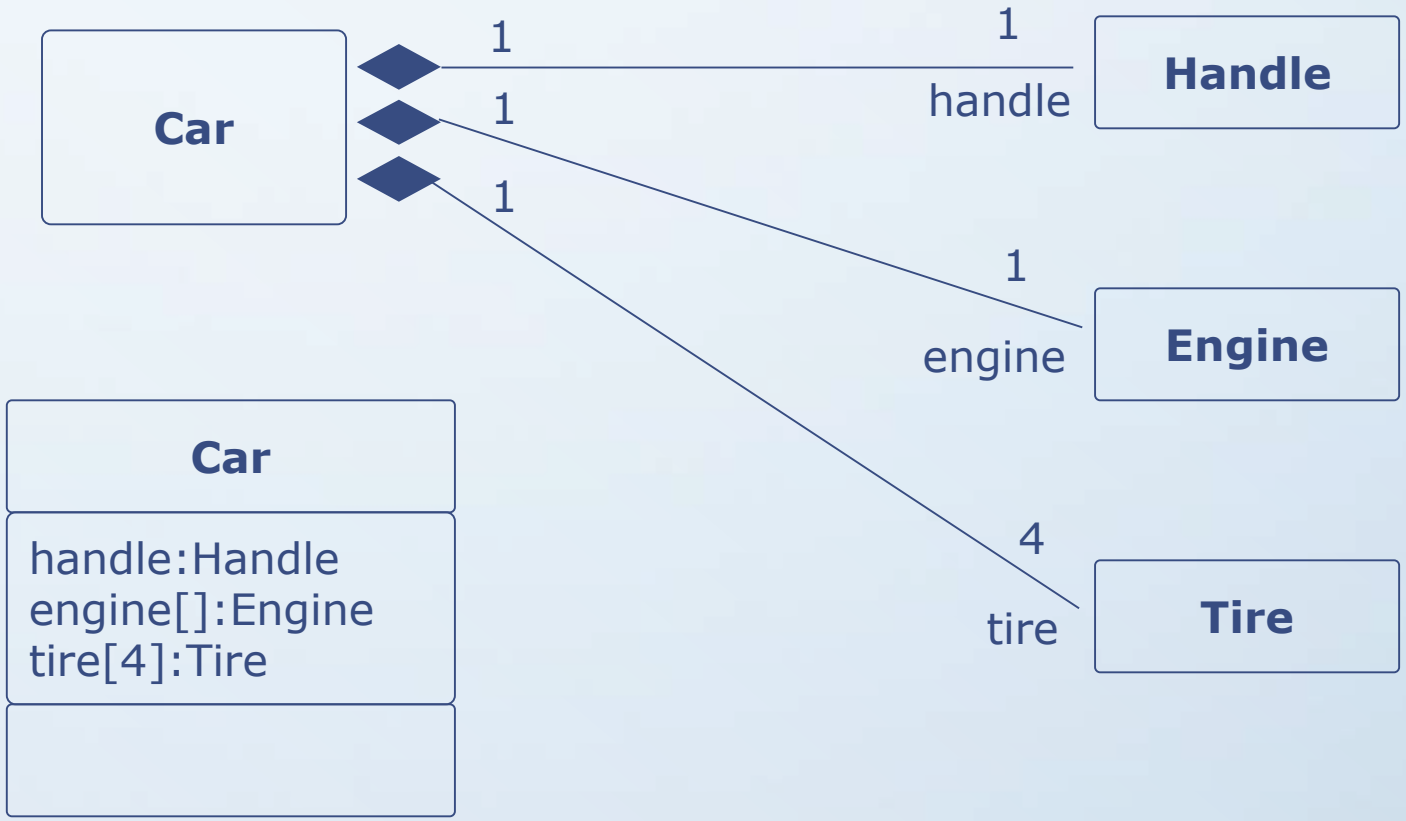




Detail of the class

Contoh Class Diagram

Composition :



Detail of the class

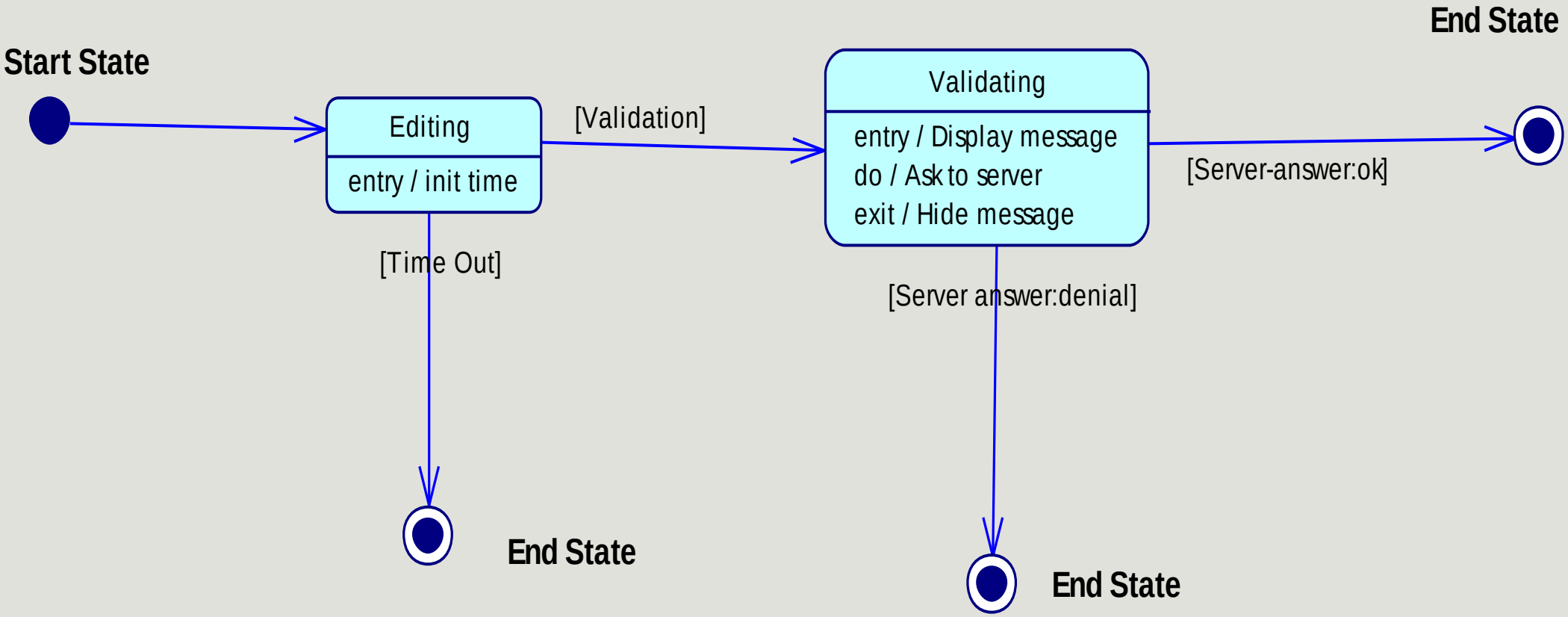
StateChart Diagram

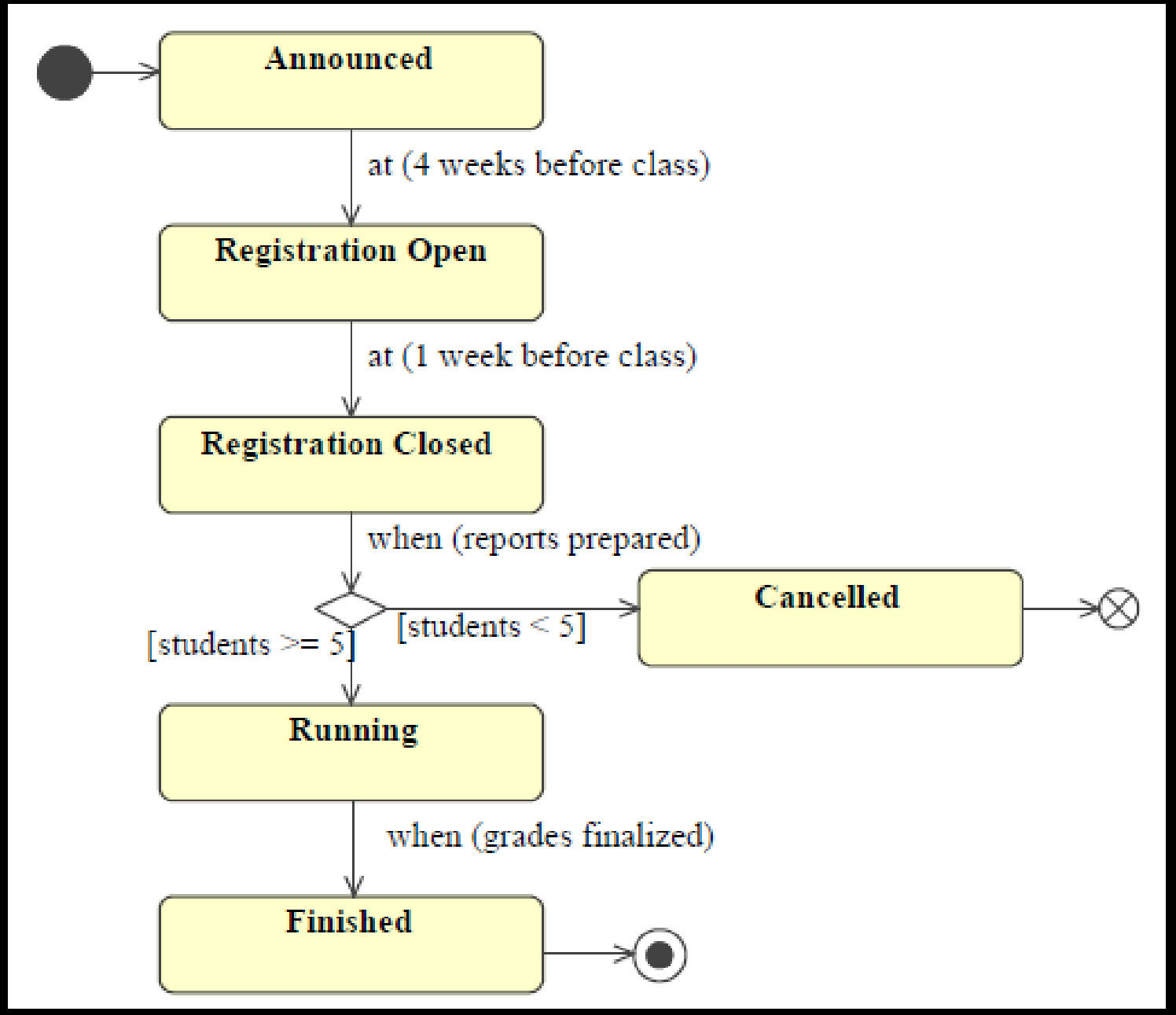
- Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya)
- Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

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



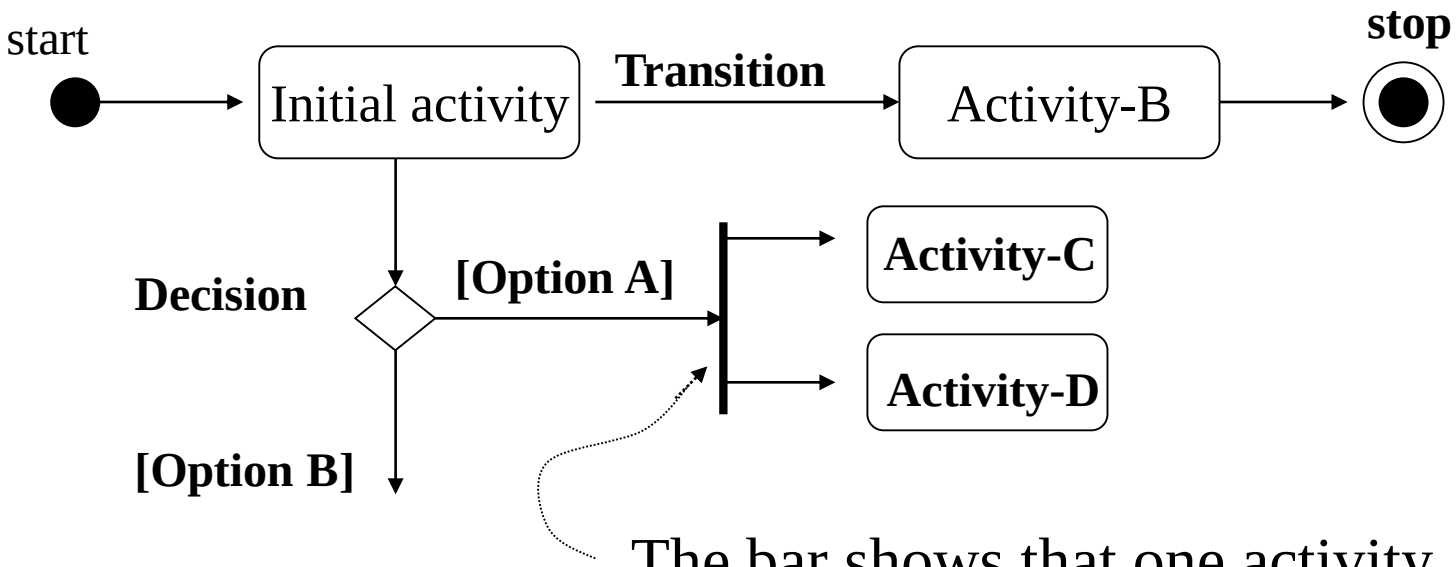


State Diagram Class Open Process

ACTIVITY DIAGRAM

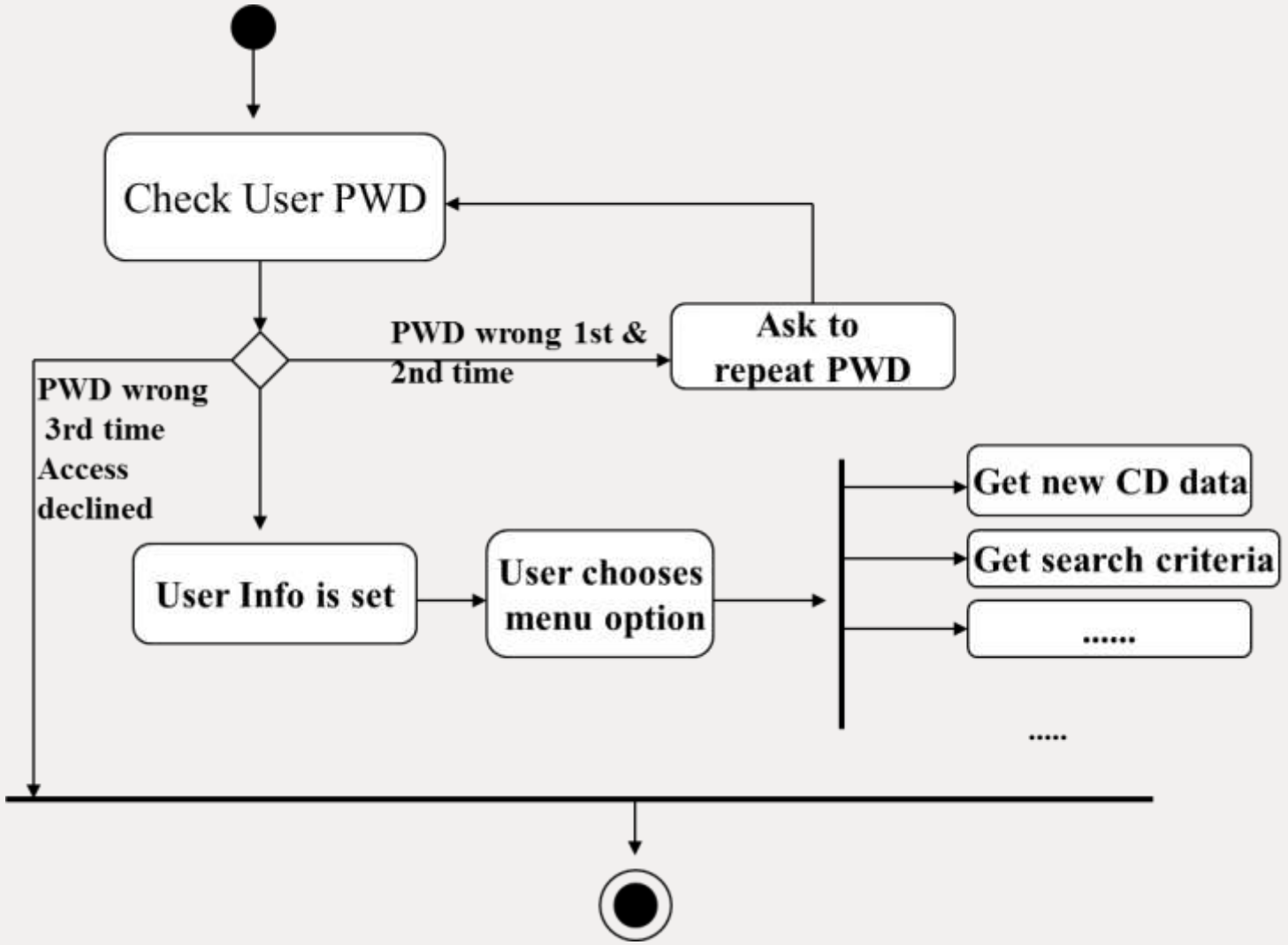
- Activity diagrams menggambarkan berbagai alir aktivitas dalam sistem yang sedang dirancang, bagaimana masing-masing alir berawal, decision yang mungkin terjadi, dan bagaimana mereka berakhir.
- Activity diagram juga dapat menggambarkan proses paralel yang mungkin terjadi pada beberapa eksekusi.
- Activity 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



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

Proses pemesanan buku