

ARTIFACT UML

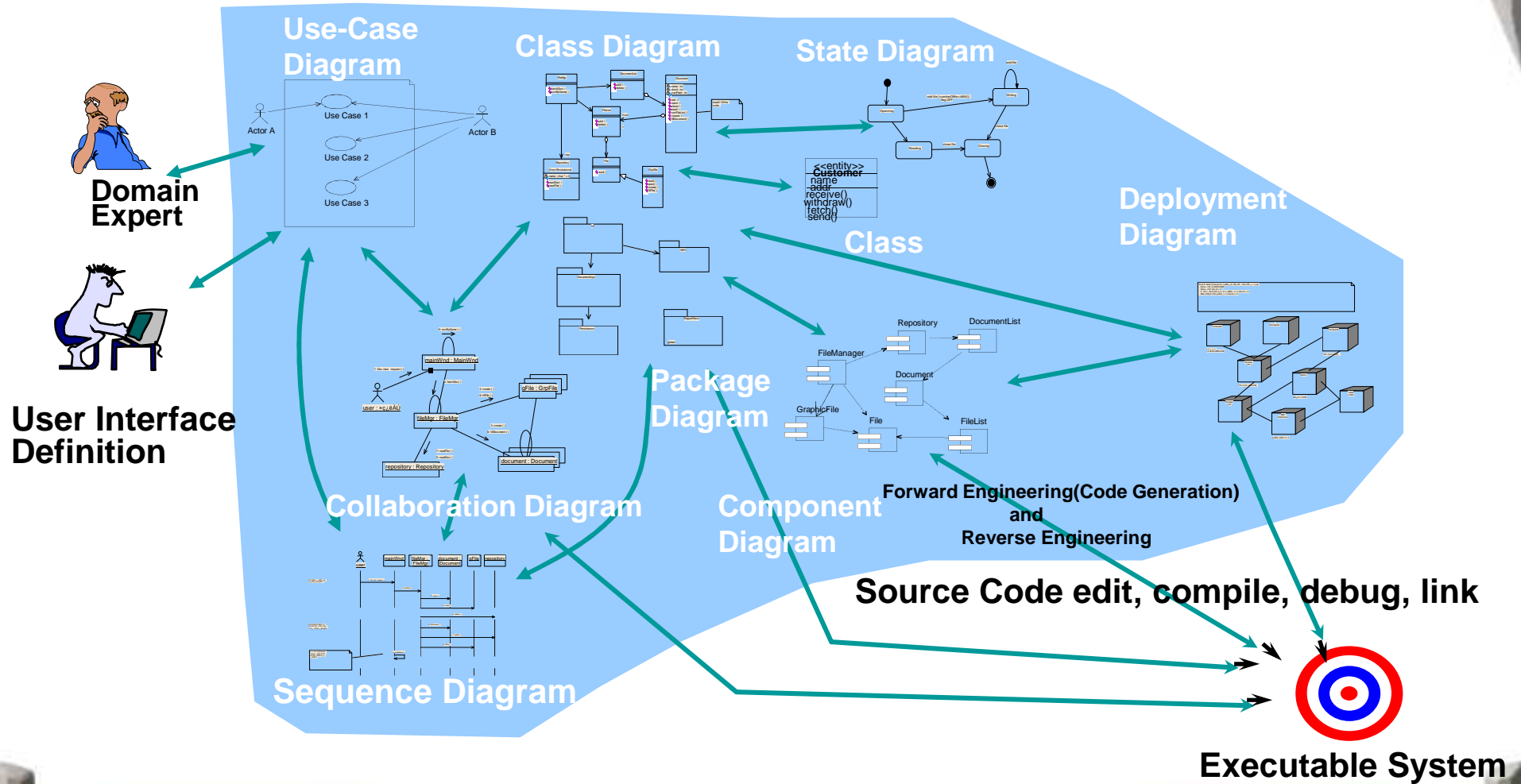
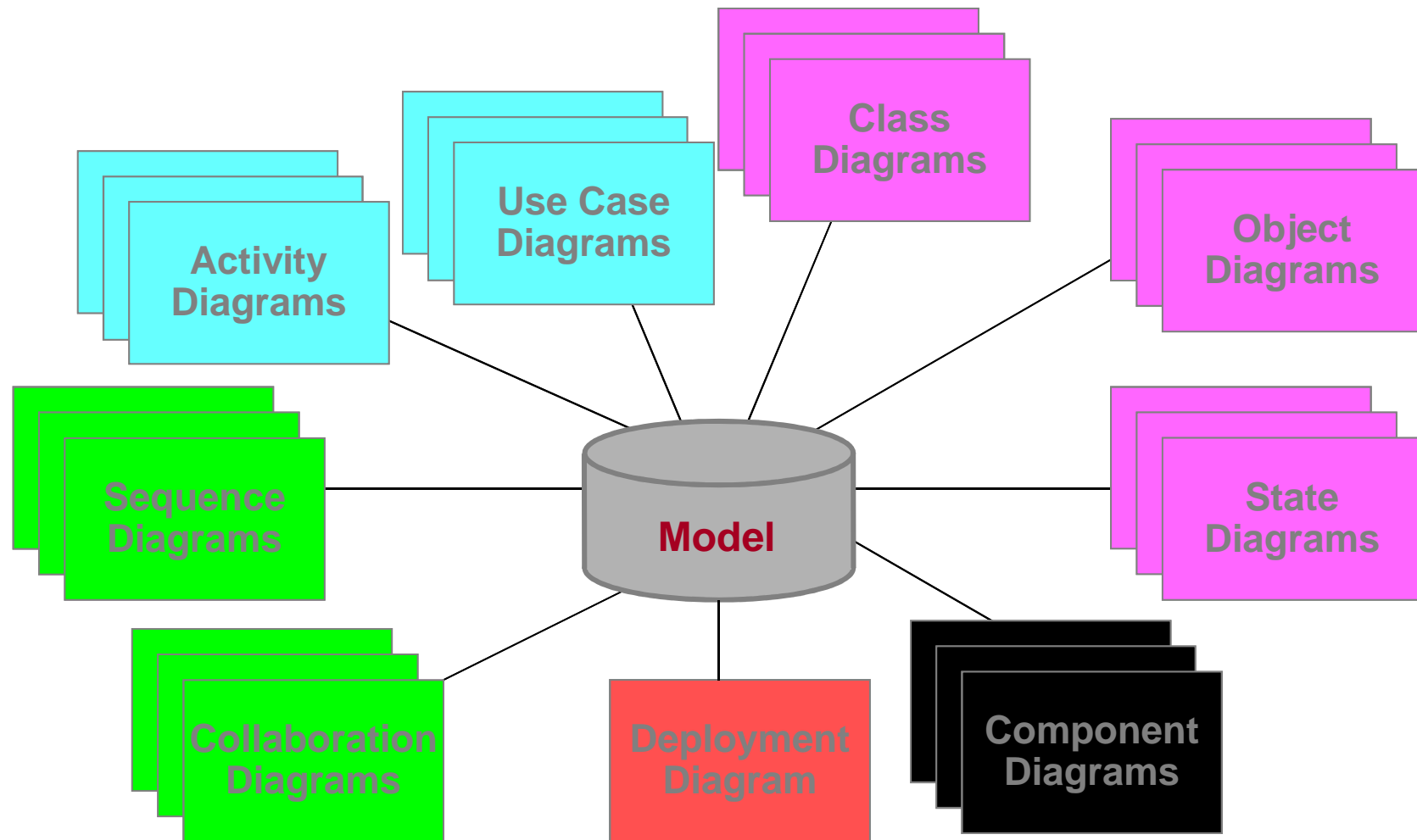


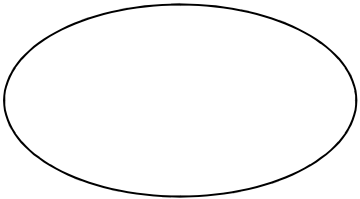
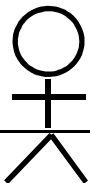

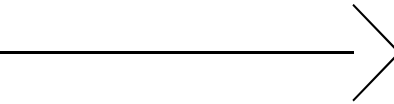
DIAGRAM-DIAGRAM DI UML



USE CASE DIAGRAM

- Menggambarkan fungsionalitas yang diharapkan dari sebuah sistem.
- Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”.
- Menggambarkan kebutuhan sistem dari sudut pandang user
- Mengfokuskan pada proses komputerisasi (automated processes)
- Menggambarkan hubungan antara use case dan actor

- Use case menggambarkan proses system (kebutuhan sistem dari sudut pandang user)
- Secara umum use case adalah:
 - Pola perilaku sistem
 - Urutan transaksi yang berhubungan yang dilakukan oleh satu actor
- Use case diagram terdiri dari
 - a. Use case
 - b. Actors
 - c. Relationship
 - d. System boundary boxes (optional)
 - e. Packages (optional)

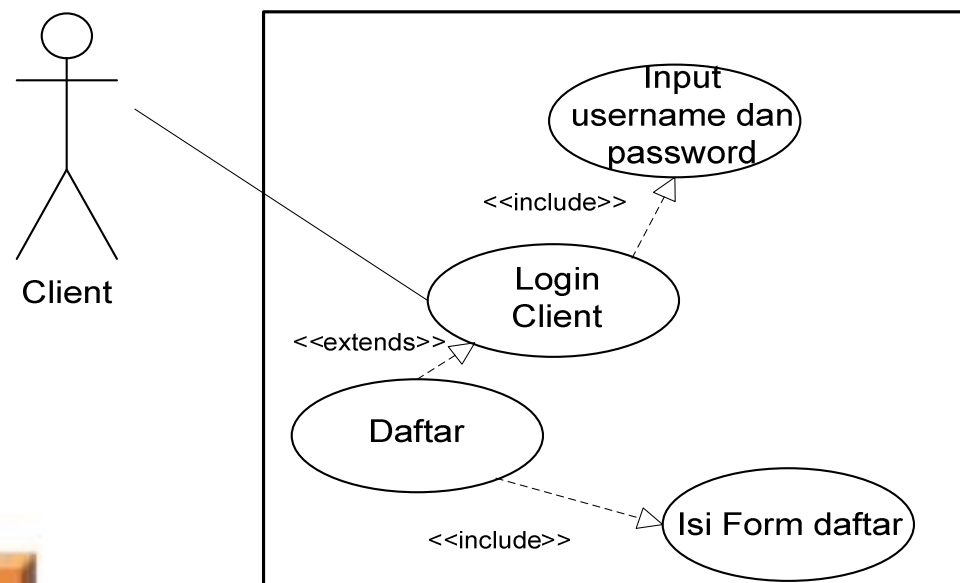
Use case	Use case dibuat berdasar keperluan actor, merupakan “apa” yang dikerjakan system, bukan “bagaimana” system mengerjakannya Use case biasanya menggunakan kata kerja	
Actor	Tidak boleh ada komunikasi langsung antar actor . <i>Actor</i> menggambarkan sebuah tugas/peran dan bukannya posisi sebuah jabatan <i>Actor</i> memberi input atau menerima informasi dari system. <i>Actor</i> biasanya menggunakan Kata benda	
Assosiation Garis tanpa panah	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	
Assosiation Panah Teerbuka	association antara actor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara pasif dengan system anda	

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 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

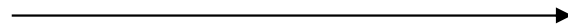


Association antara use case (Lanjut)

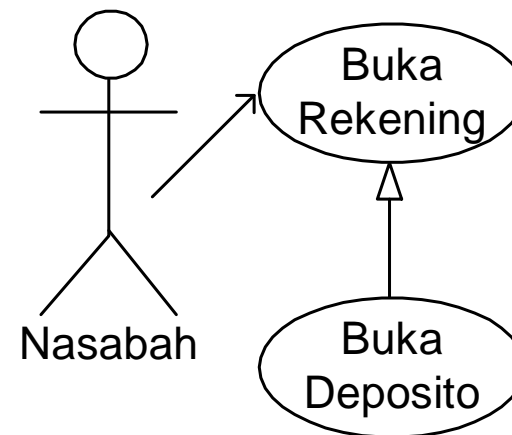
- <<extend>> perluasan dari use case lain jika kondisi atau syarat terpenuhi
 - Kurangi penggunaan association Extend ini, terlalu banyak pemakaian association ini membuat diagram sulit dipahami.
 - Tanda panah terbuka harus terarah ke parent/base use case
 - Gambarkan association extend secara vertical

Generalization/inheritance antara use case

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

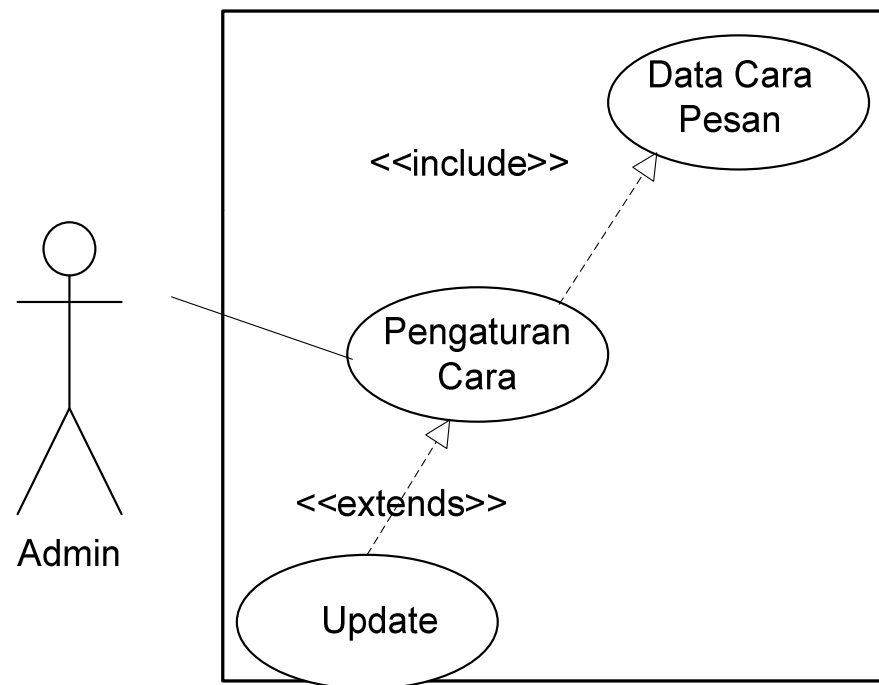


- Gambarkan generalization/inheritance antara use case secara vertical dengan inheriting use case dibawah base/parent use case
- Generalization/inheritance dipakai ketika ada sebuah keadaan yang lain sendiri/perlakuan khusus (*single condition*)



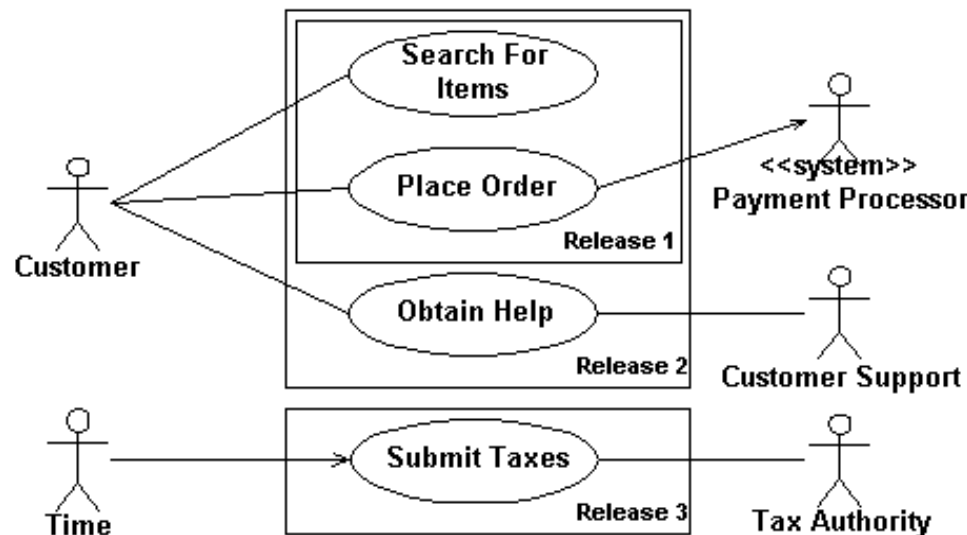
Generalization/inheritance antara actor

- 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



CLASS DIAGRAM

CLASS DIAGRAM

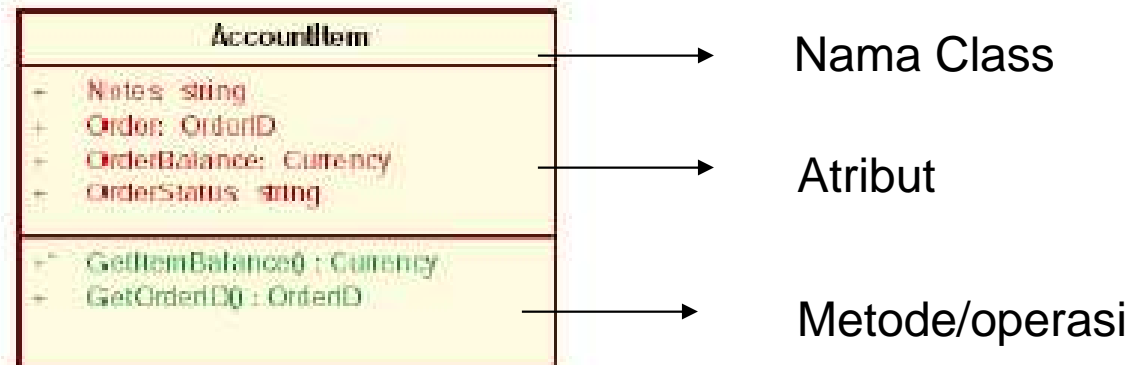
- Adalah sebuah spesifikasi yang jika diinstansiasi akan menghasilkan sebuah objek dan merupakan inti dari pengembangan dan desain berorientasi objek.
- *Class* menggambarkan keadaan (atribut/properti) suatu sistem, sekaligus menawarkan layanan untuk memanipulasi keadaan tersebut (metoda/fungsi).
- *Class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

Class memiliki tiga area pokok :

1. Nama, merupakan nama dari sebuah kelas
2. Atribut, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari class
3. Operasi, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*

CLASS DIAGRAM (LANJUTAN)

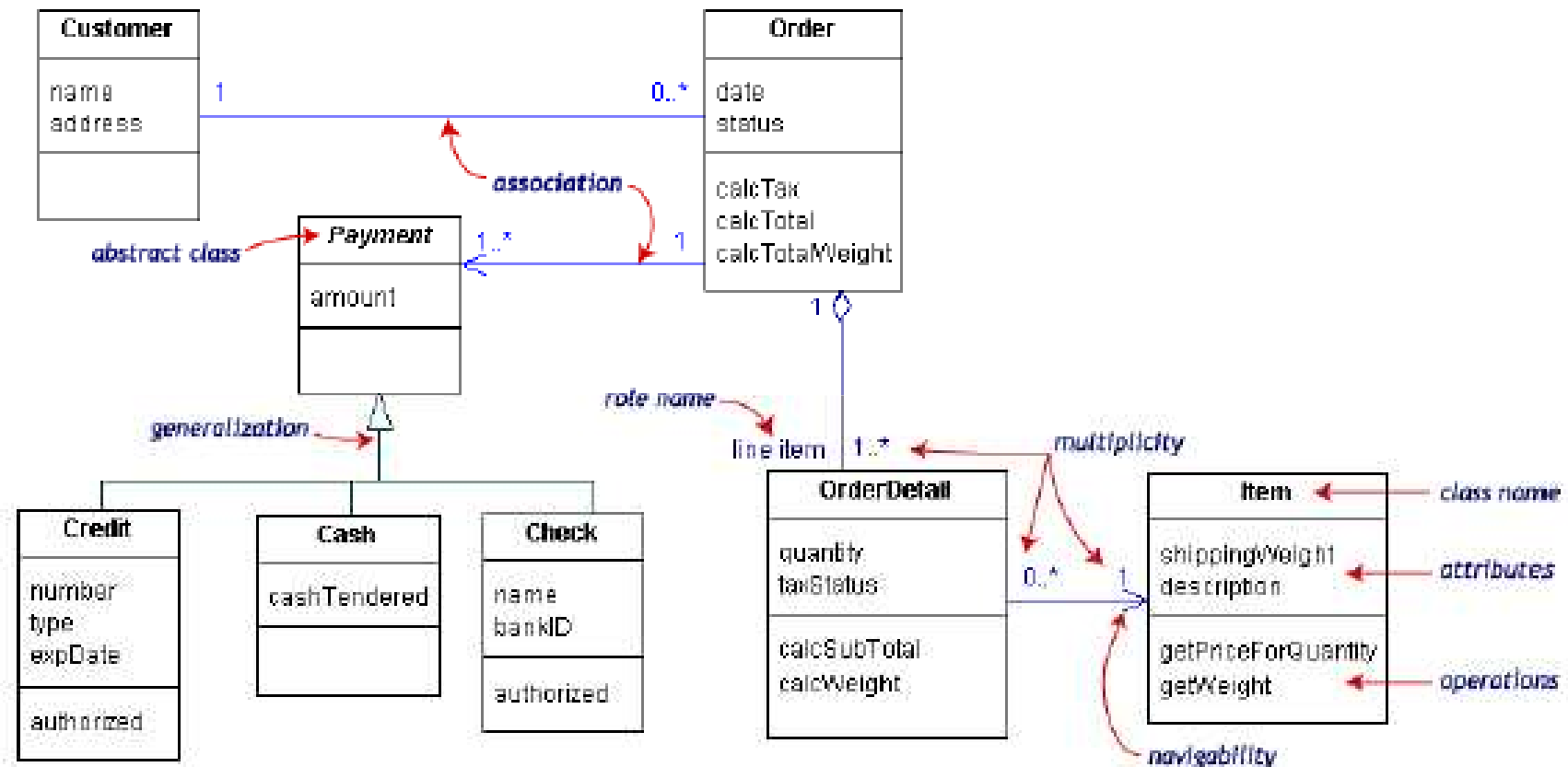
- Atribut dan metoda dapat memiliki salah satu sifat berikut :
 - *Private*
 - *Protected*
 - *Public*
 - *Package*



HUBUNGAN ANTAR CLASS

1. Asosiasi, yaitu hubungan statis antar *class*.
2. Agregasi, yaitu hubungan yang menyatakan bagian (“terdiri atas..”).
3. Pewarisan, yaitu hubungan hirarkis antar *class*. *Class* dapat diturunkan dari *class* lain dan mewarisi semua atribut dan metoda *class* asalnya dan menambahkan fungsionalitas baru.
4. Hubungan dinamis, yaitu rangkaian pesan (*message*) yang di-*passing* dari satu *class* kepada *class* lain.

CONTOH CLASS DIAGRAM



MULTIPLICITY

- Unspecified
- Exactly one
- Zero or more (many, unlimited)
- One or more
- Zero or one (optional scalar role)
- Specified range
- Multiple, disjoint ranges

1

0..*

*

1..*

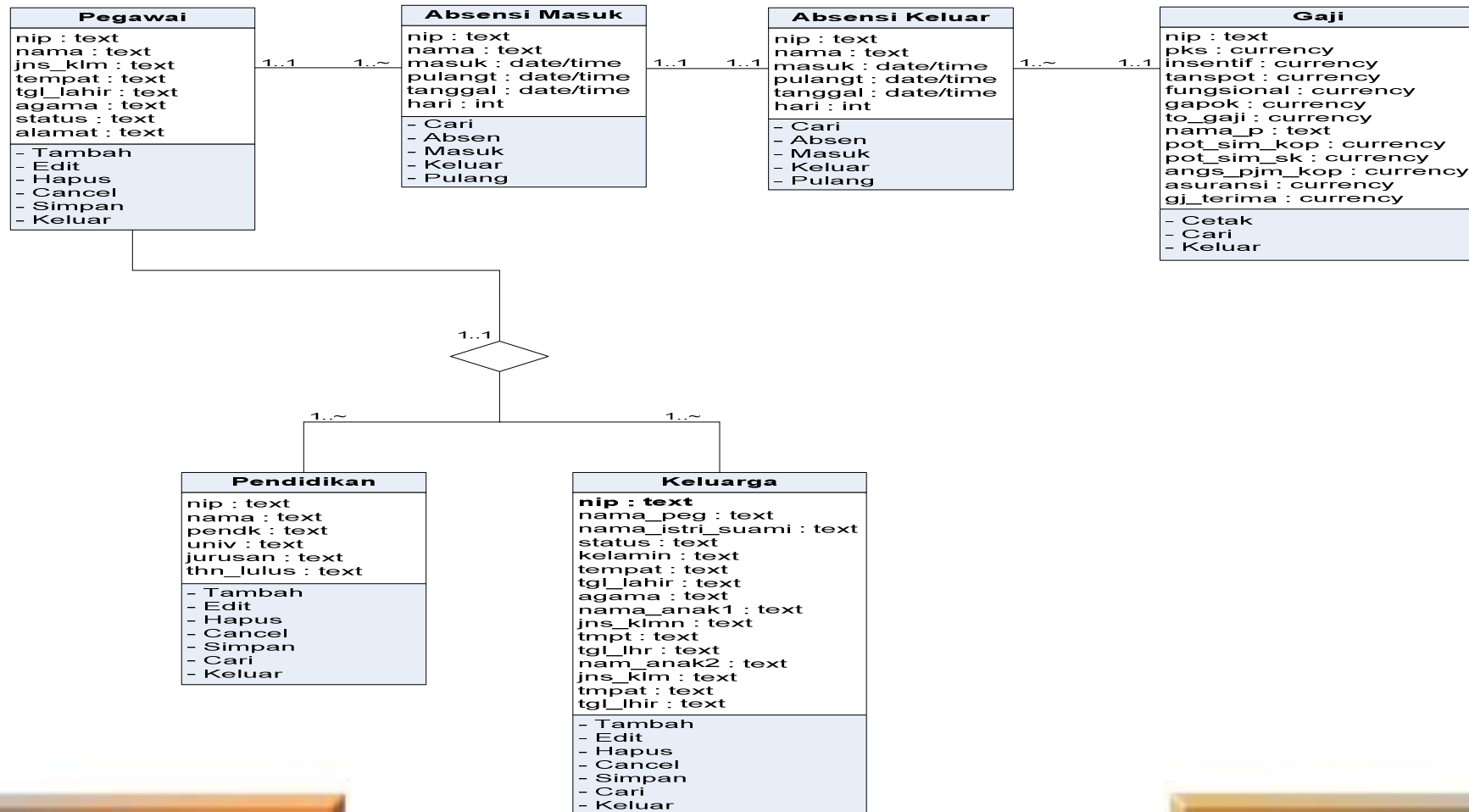
0..1

2..4

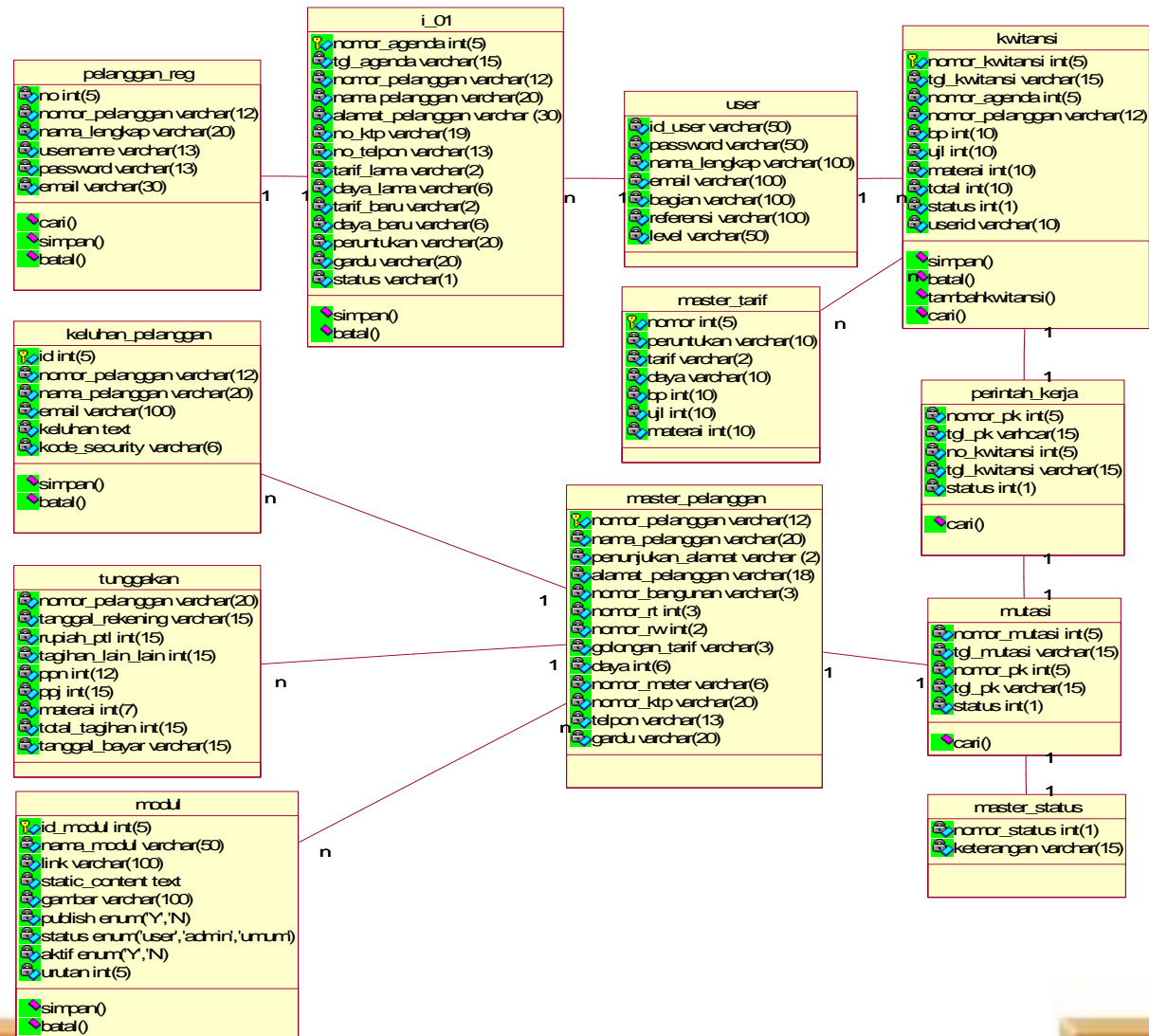
2, 4..6

Class Diagram diperoleh berdasarkan dari database

Contoh Kasus (Acknowledgments Evi Lutfi Muktar)

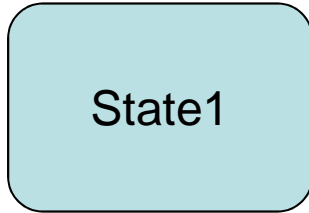

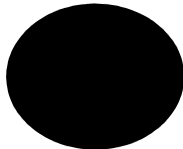
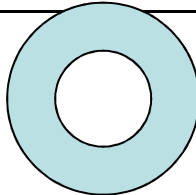


Contoh Kasus (Acknowledgments Toeko triyanto)



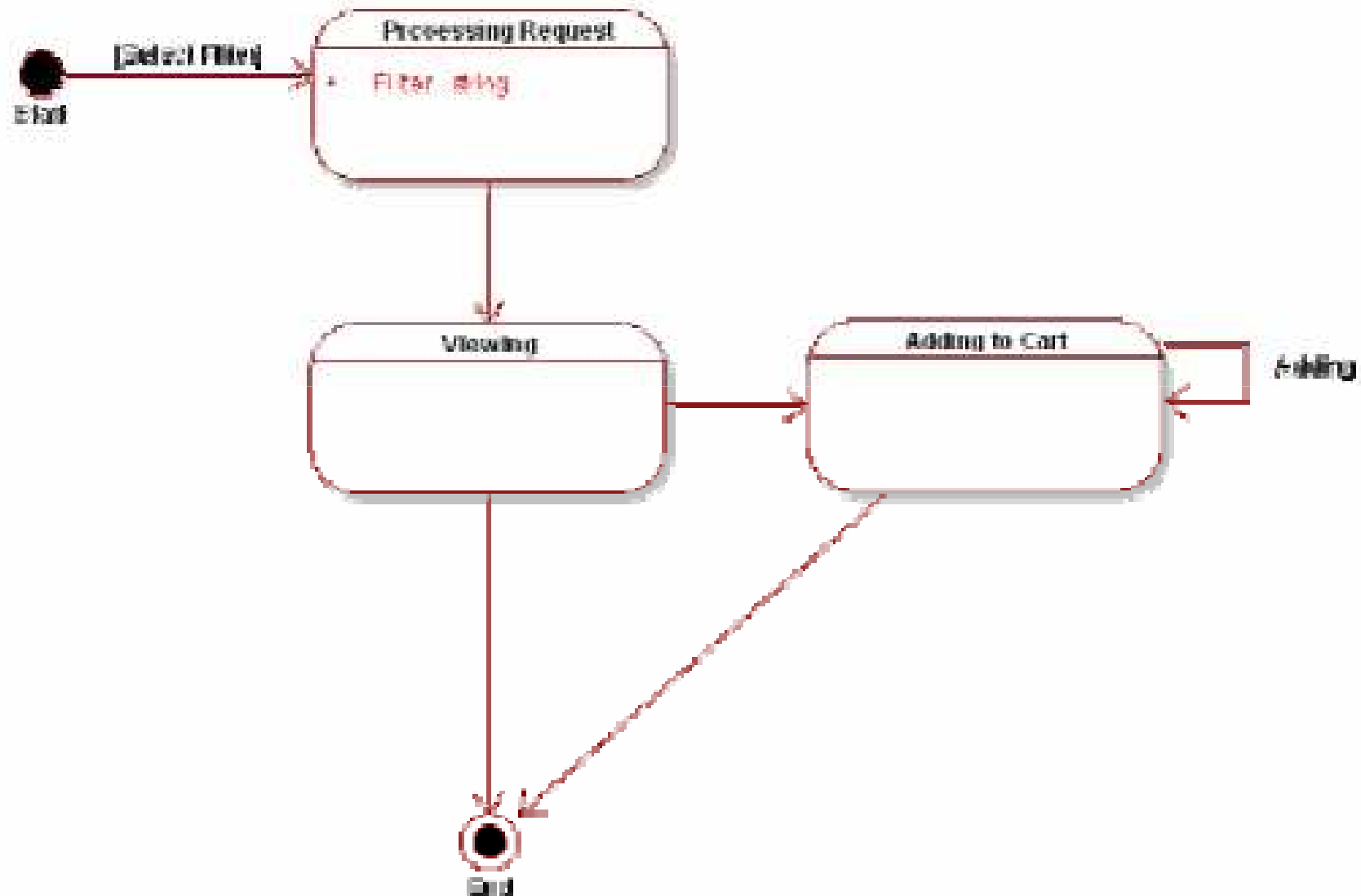
Statechart Diagram.

- *Statechart diagram/ state diagram*
digunakan untuk mendokumentasikan beragam kondisi/keadaan yang bisa terjadi terhadap sebuah *class* dan kegiatan apa saja yang dapat merubah kondisi/keadaan tersebut.

State	Notasi State menggambarkan kondisi sebuah entitas, dan digambarkan dengan segiempat yang pinggirnya tumpul dengan nama state didalamnya	
Transition	Sebuah Transition menggambarkan sebuah perubahan kondisi objek yang disebabkan oleh sebuah event. Transition digambarkan dengan sebuah anak panah dengan nama event yang ditulis diatasnya, dibawahnya atau sepanjang anak panah tersebut.	
Initial State	sebuah kondisi awal sebuah object sebelum ada perubahan keadaan. Initial State digambarkan dengan sebuah lingkaran solid. Hanya satu Initial State yang diizinkan dalam sebuah diagram	
Final State	menggambarkan ketika objek berhenti memberi respon terhadap sebuah event. Final State digambarkan dengan lingkaran solid didalam sebuah lingkaran kosong.	

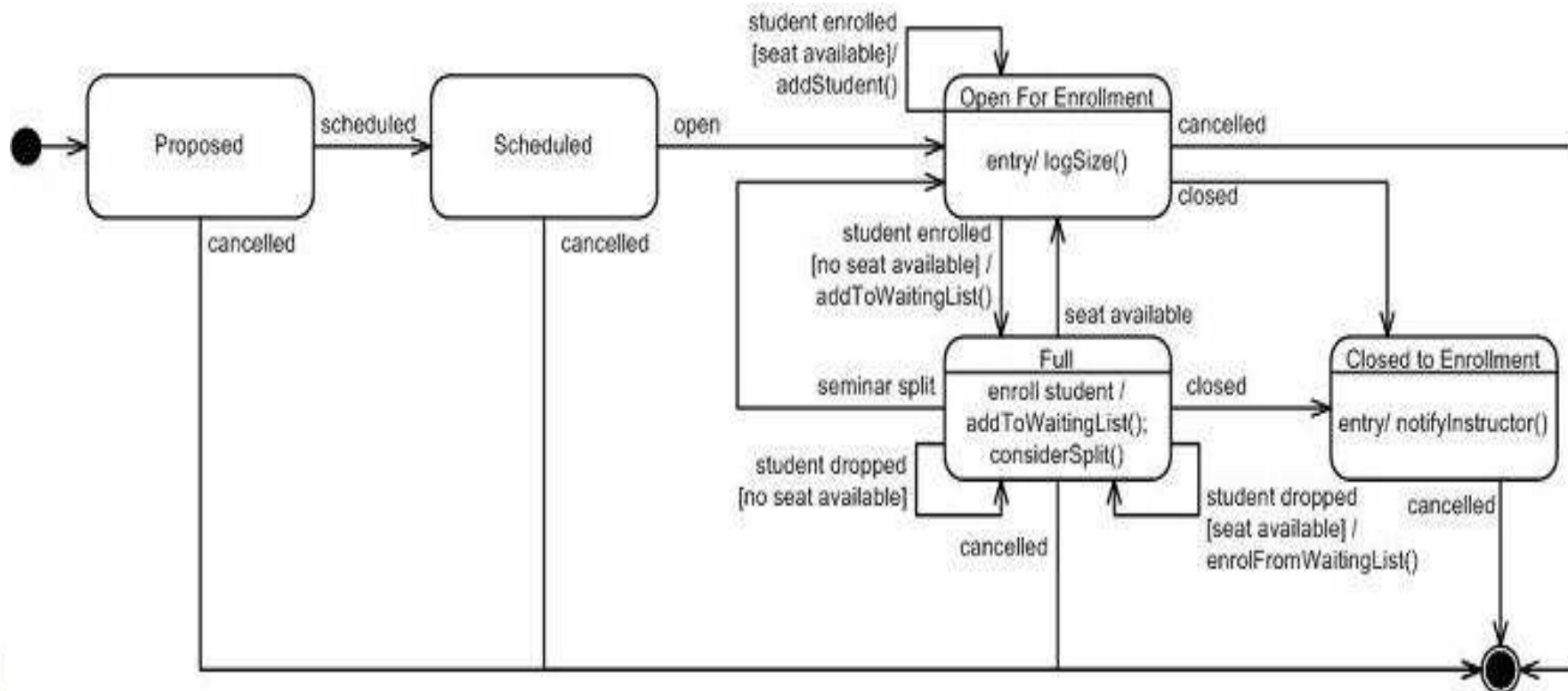
- *Statechart diagram* menggambarkan transisi dan perubahan keadaan (dari satu *state* ke *state* lainnya)
- Pada umumnya *statechart diagram* menggambarkan *class* tertentu (satu *class* dapat memiliki lebih dari satu *statechart diagram*).
- *State* digambarkan berbentuk segiempat dengan sudut membulat dan memiliki nama sesuai kondisinya
- 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.

Contoh State Diagram



State Machine Diagram (Statechart diagram in versi 1.x)

- Untuk memodelkan behavior/methode (lifecycle) sebuah kelas atau object
- Memperlihatkan urutan kejadian sesaat (state) yang dilalui sebuah object, transisi dari sebuah state ke state lainnya



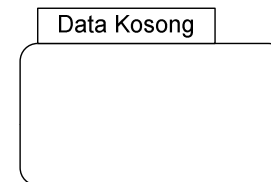
State Machine Diagram (Statechart diagram in versi 1.x)

Sebuah state machine diagram mempunyai :

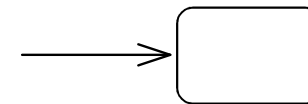
- state (kejadian sesaat) are represented by the values of attributes of an object
 - State digambarkan dengan bentuk Data Kosong



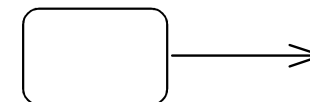
atau



- “Black Hole” states
is state has transitions into it but none out

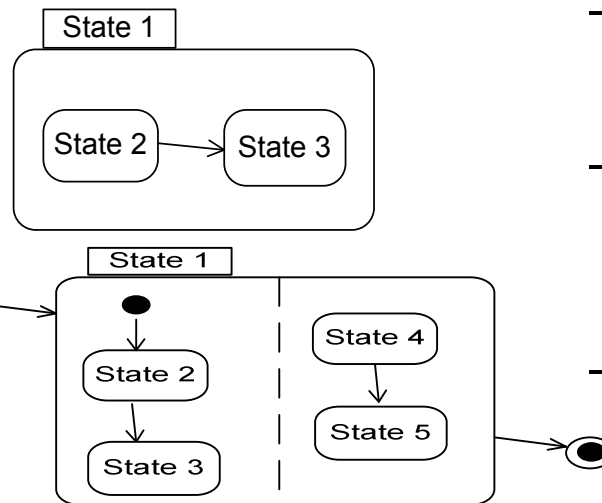


- Miracle states
is state has transitions out of it but none into it



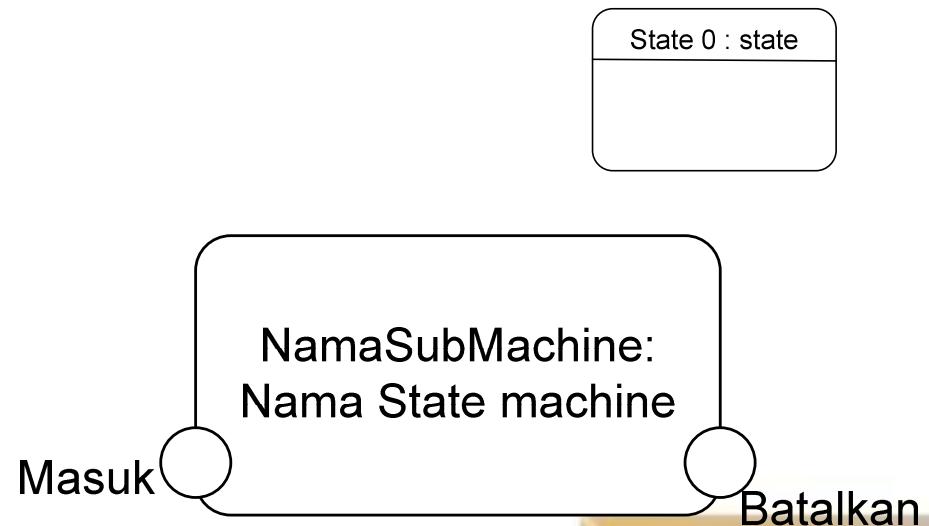
State Machine Diagram (Statechart diagram in versi 1.x)

- Composite State
 - Kumpulan dari beberapa states yang setidaknya dalam sebuah region
 - Orthogonal State, jenis composite state lebih dari 1 region



- Digunakan untuk mendukung konsep encapsulation
- Sebuah state tidak boleh mempunyai region dan submachine secara bersamaan
- Nama state mempunyai sintaks :
nama submachine state :
referenced state machine

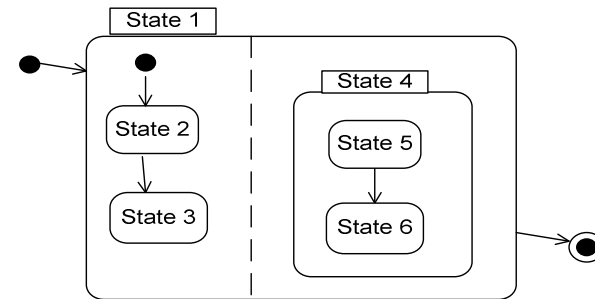
- Submachine State
 - Sejenis composite state yang isinya didefinisikan oleh state machine lain
 - State Machine yang berisi submachine state disebut “Containing state machine”
 - Sebuah state yang dihubungkan ke state machine lainnya
 - Dihubungkan ke satu/lebih entry point dan satu/lebih exit point



State Machine Diagram (Statechart diagram in versi 1.x)

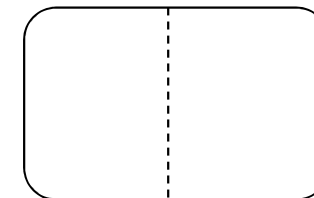
Sub States

- Sebuah state yang ada dalam sebuah region
 - Direct Substate, Sub state yang tidak berisi state lain
 - Indirect Substate, Sub state yang berisi state lain



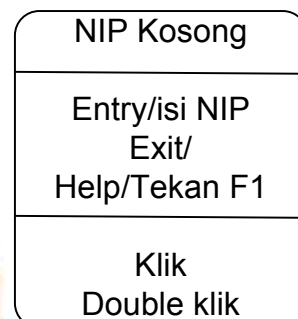
Region (kelompok state)

- Dipisahkan dengan garis terputus, yang setiap region boleh mempunyai nama sebagai optional
- Sebuah state tidak boleh mempunyai region dan submachine secara bersamaan



State terpisah menjadi 3 bagian yaitu

- Activity label bisa berupa Entry, Exit atau do
- Dimana Activity expression adalah penggunaan atribut



Nama State

Internal Activity, kegiatan yang dilakukan dalam state
sintaks : Activity label/activity expression

Internal transition

State Machine Diagram (Statechart diagram in versi 1.x)

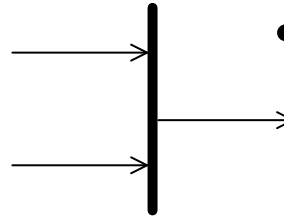
label on transition is in the format

event [guard][/*methode list()*]

- event biasa dituliskan dengan past tense
- event menyebabkan sebuah object berpindah dari satu state ke state lain
- Guard, condition that must be true for the transition to be triggered
- Guard harus konsisten dan tidak overlap
Contoh: $X < 0$, $X = 0$ dan $X > 0$ konsisten
 $X \leq 0$ dan $X \geq 0$ tidak konsisten
- Guards harus lengkap logikanya
Contoh: $X < 0$ dan $X > 0$, bagaimana jika $X = 0$?
- Methode dijalankan
 - ketika object memasuki state diindikasikan dengan methode bernama `entry()`
 - ketika object keluar state diindikasikan dengan methode bernama `exit()`
- Methode menyebabkan perubahan di sebuah state bisa juga tidak

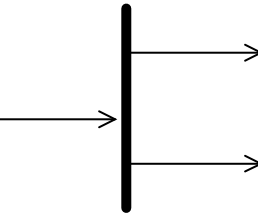
State Machine Diagram (Statechart diagram in versi 1.x)

- Join, menggabungkan beberapa transition menjadi sebuah transition

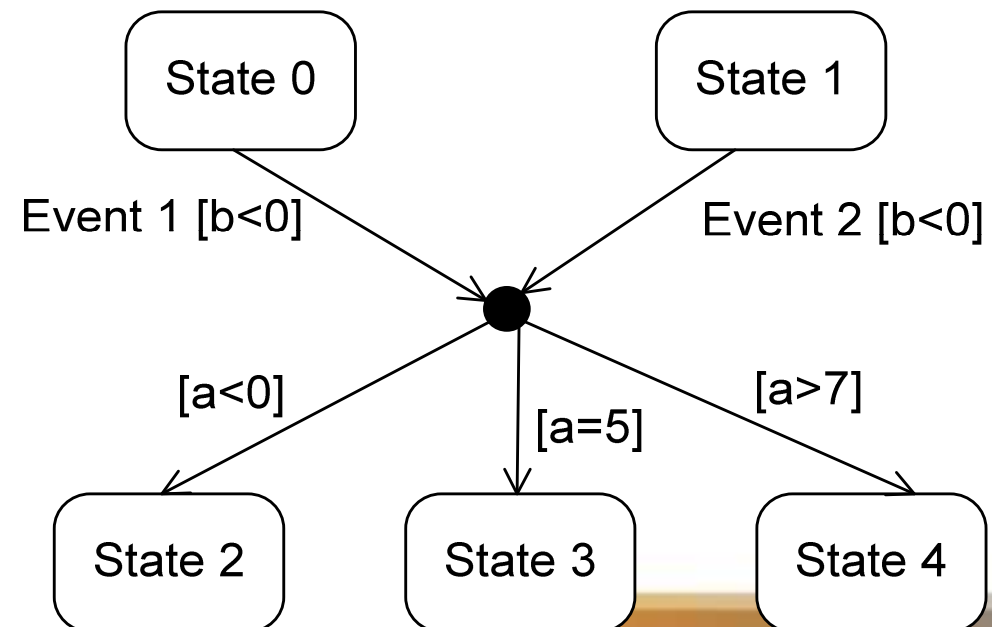


- Dimungkinkan transition ke sebuah state yang berisi beberapa state yang disebut state list

- Fork, memecah sebuah transition menjadi beberapa transition yang berkondisi AND (transition harus dilewati semuanya).

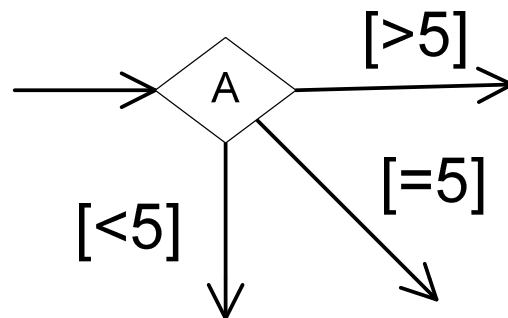


- Junction, Menggabungkan sebuah/beberapa transition dan memecahnya menjadi sebuah/beberapa transition yang berkondisi AND (transition harus dilewati semuanya). Digunakan tanda lingkaran hitam kecil Contoh:

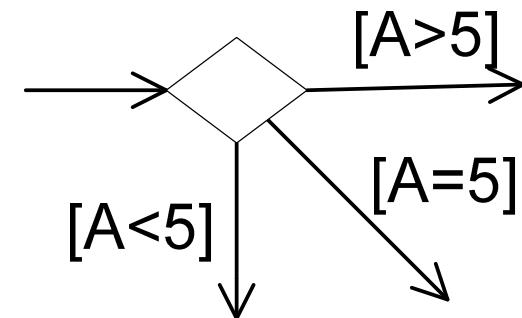


- Choice, Mengkondisikan sebuah transition menjadi sebuah/beberapa transition, yang hanya dipilih salah satu transition(choice).
 - Digunakan lambang diamond 
 - Operand dapat diletakkan didalam diamond atau pada transition

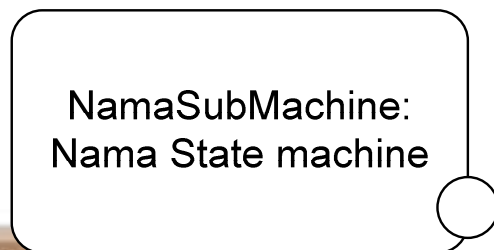
Contoh:



atau

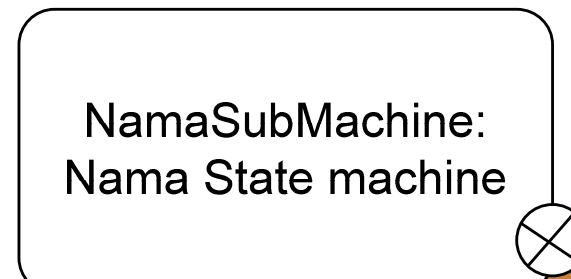


- Entry point
Dilambangkan sebuah lingkaran kecil yang ditaruh pada pinggiran state(bisa juga didalam atau diluar), dan berguna sebagai submachine state



lagi

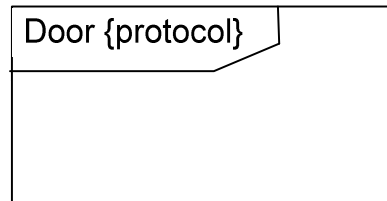
- Exit point
Dilambangkan sebuah lingkaran kecil bersilang yang ditaruh pada pinggiran state (bisa juga didalam atau diluar), dan berguna sebagai submachine state



batalan

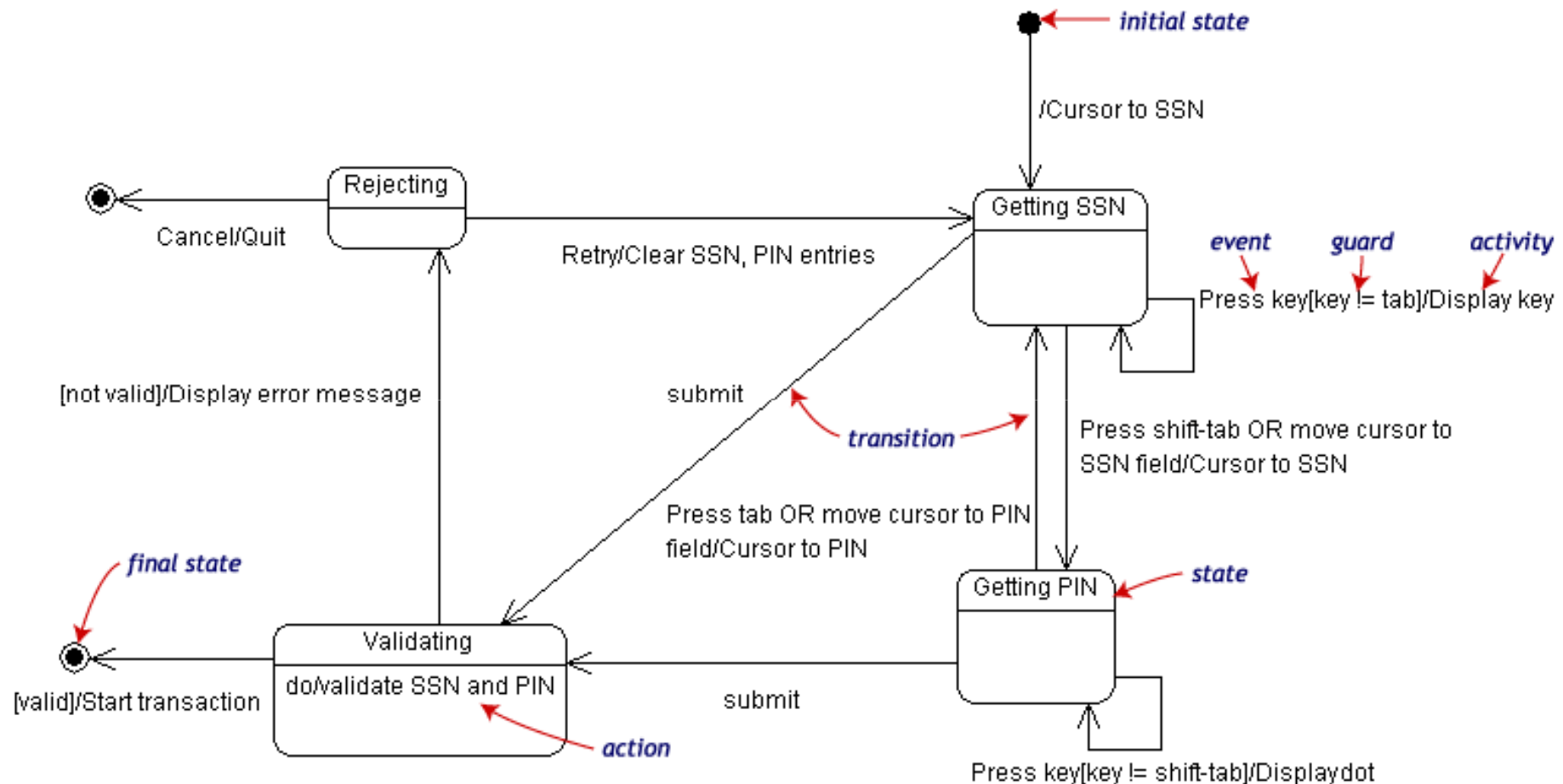
State Machine Diagram (Statechart diagram in versi 1.x)

- State Machine Diagram ada 2 jenis
 - Behavioral State Machines
 - Protocol State Machines
- Tidak adanya internal activity seperti entry, exit, do
- Transition pada Protocol State Machines harus menggunakan Protocol Transition



- Protocol Transition
 - Sintaks : [pre condition] event / [post condition]
 - precondition atau postcondition adalah guard (Guard is condition that must be true for the transition to be triggered)
 - Precondition, kondisi sebelum transition
 - Postcondition, kondisi setelah transition

Contoh State Diagram



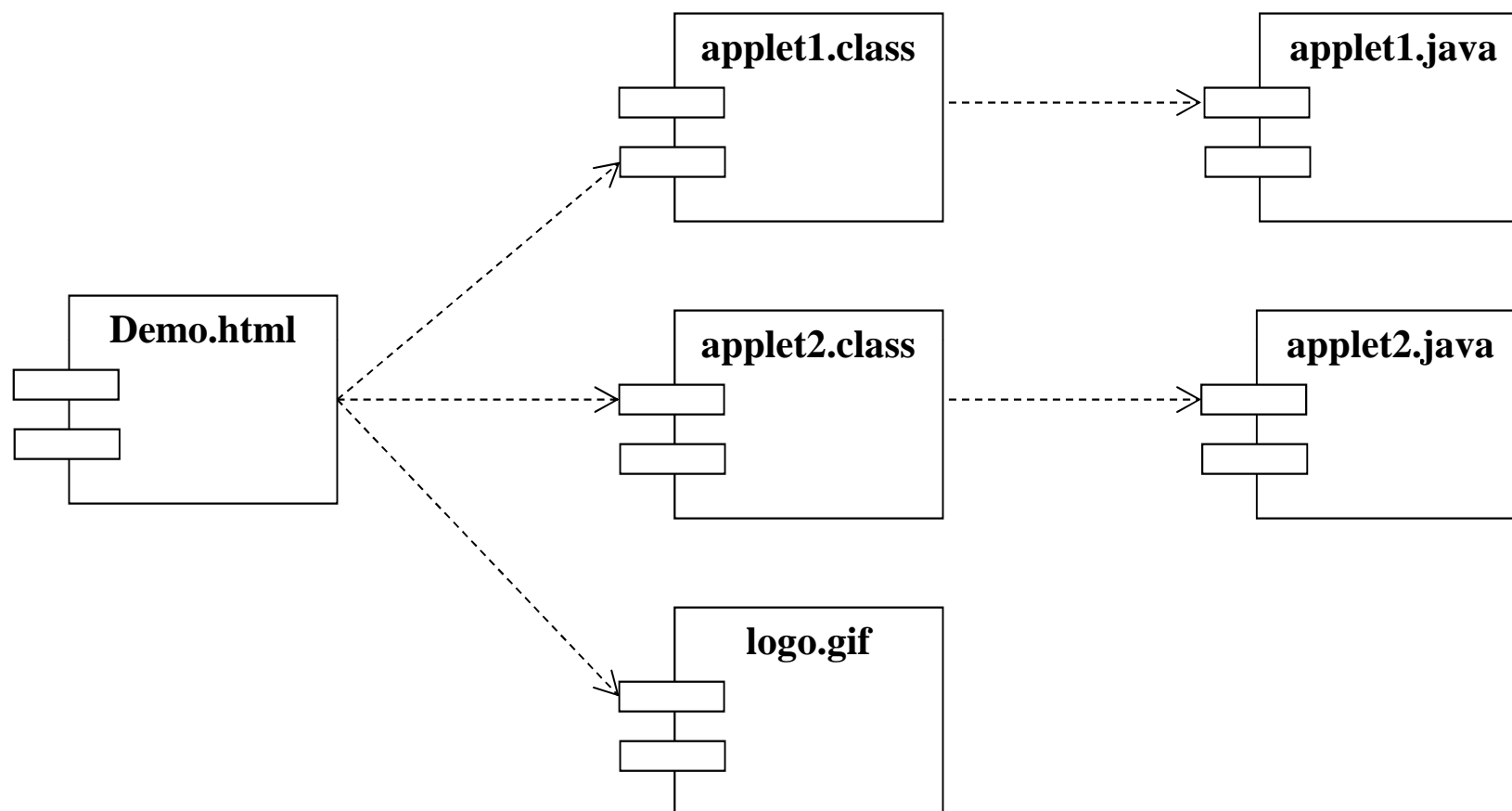
Deployment Diagram

- ***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
- ***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.

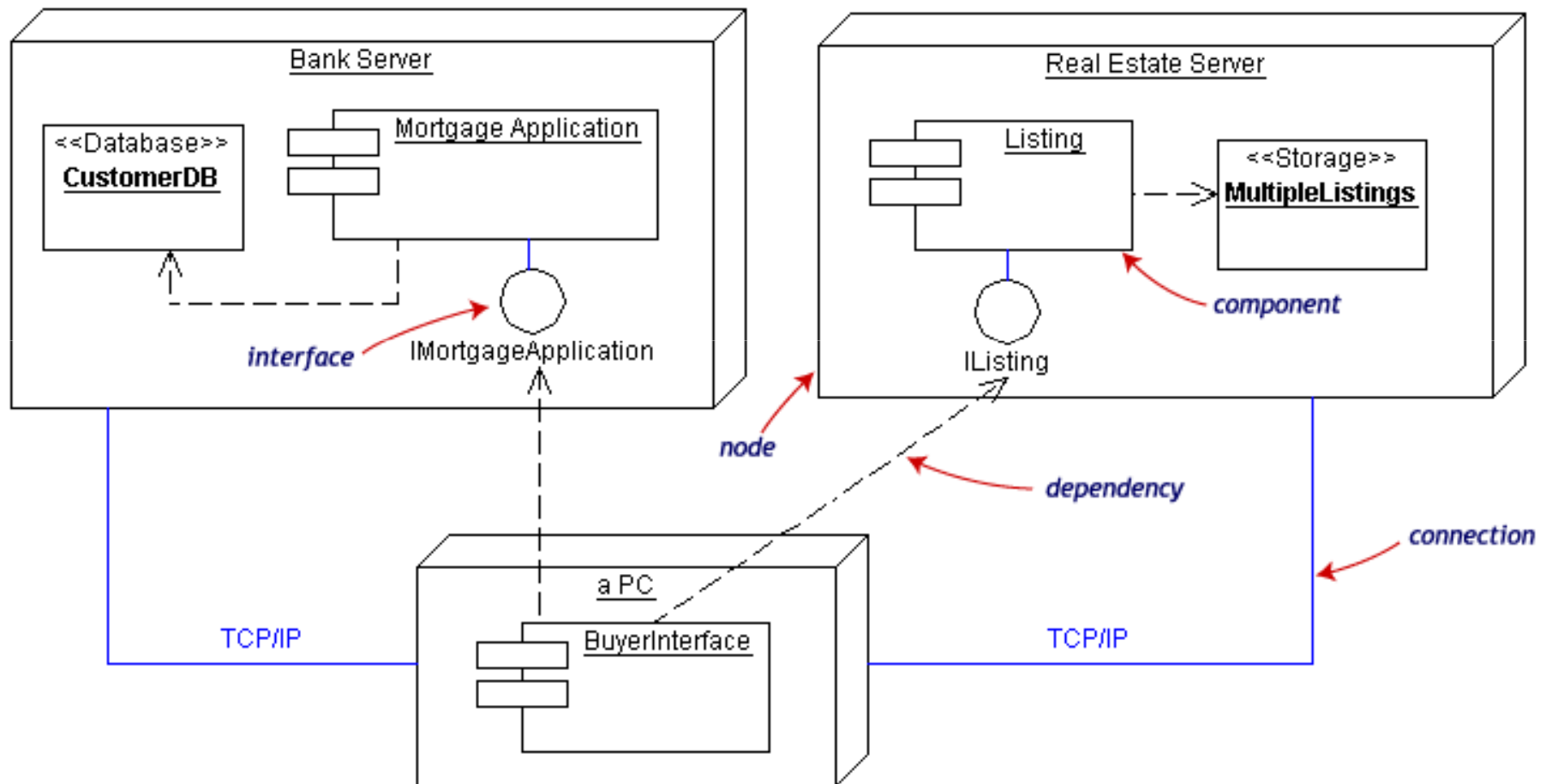
Component Diagram

- *Component diagram* menggambarkan struktur dan hubungan antar komponen piranti lunak, termasuk ketergantungan (*dependency*) di antaranya.
- Komponen piranti 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.

Contoh : Component Diagram

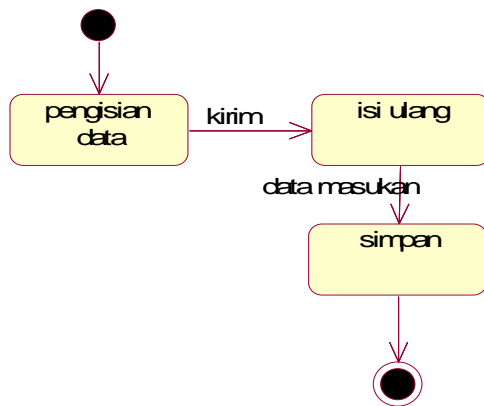


Contoh : Component & Deployment Diagram

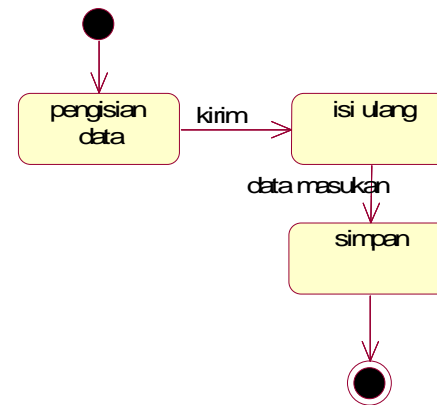


Contoh kasus (Acknowledgments Toeko triyanto)

state chart diagram pendaftaran



statechart diagram pengisian data kwitansi.



Bobot 30% (design)

Tugas : - Berdasarkan tugas pada pertemuan sebelumnya
(Pengembangan dari program yang pernah dibuat)

Buatlah design UML dari sistem usulan dengan
apakah itu berupa program desktop, web, animasi
atau sistem pakar (pilih salah satu)

- (untuk pertemuan 4, 5 dan 6) buatlah
rancangannya dengan menggunakan Tools,
misalnya : Enterprise Architect , Rational Rose,
Argo UML, Visual Paradigm dan lain-lain, sesuai
dengan Diagram yang telah dipelajari diatas.

-Dikumpulkan berupa laporan