



Unified Modeling Language

Arna Fariza



Aturan Analisa dan Desain

- ☐ Memisahkan konstruksi software
 - o Analisa kebutuhan
 - o Arsitektur software
 - o Spesifikasi (High-Level/Early design)
 - o Desain detail
 - o Implementasi dan testing
 - o Maintenance dan evolusi
- ☐ Setiap fase desain/development dipisahkan



Aturan Analisa dan Desain

- ❑ Desain
 - o Mengembangkan solusi logika
 - o Mewakili satu cara untuk penyelesaian permasalahan
 - o Menentukan sistem BAGAIMANA yang memenuhi APA!
- ❑ Desain OO
 - o Menekankan pada menentukan obyek dan komponen software logika
 - o Evaluasi desain OO alternatif
 - o Menuju implementasi untuk pemecahan kelayakan
- ❑ Peringatan: A+D adalah proses yang berkelanjutan
- ❑ Kesuksesan dan verifikasi A+D menuju ke rekayasa software yang dapat diukur



Konsep Mendefinisikan Komponen

- ❑ **Komponen** terdiri dari satu atau lebih class (atau komponen lain) dan dimaksudkan untuk mendukung unit fungsional yang dibangun
- ❑ **Class** dapat digunakan pada **banyak komponen**
- ❑ Class yang digunakan dalam banyak komponen menggunakan **semantik yang sama** dalam semua konteks
- ❑ **Terdiri dari:**
 - o Desain berbasis komponen
 - o Saling ketergantungan antar komponen
 - o Pandangan alternatif dari interaksi komponen
 - o Kerangka komponen reusable



Sejarah Desain OO

- ❑ Selama 15+ tahun terakhir, beberapa pemain dalam OOD
 - o Booch: **The Booch Method**
 - “Object-Oriented Design with Application,” Benjamin/Cummings, 1991.
 - o Rumbaugh: **OMT**
 - “Object-Oriented Modeling and Design,” Prentice-Hall, 1991.
 - o Meyer: **Client/Server Contract Approach**
 - “Object-Oriented Software Construction,” Prentice-Hall, 1988.
 - o Jacobson: **Use-Cases and Software Engrg.**
 - “Object-Oriented Software Engineering: A Use Case Driven Approach,” Addison-Wesley, 1992.



Sejarah Desain OO

- ❑ Pemain dalam OOD - lanjutan
 - o Coleman: **The Fusion Method**
 - “Object-Oriented Development - The Fusion Method,” Prentice-Hall, 1994.
 - o Lieberherr: **Adaptive OO Software**
 - “Adaptive OO Software: The Demeter Method with Propagation Patterns,” PWS, 1996.
 - o Gamma: **Design Patterns**
 - “Design Patterns: Elements of Reusable Object-Oriented Software,” Addison-Wesley, 1995.
 - o Booch and Rumbaugh: **UML Predecessor**
 - “Unified Method for Object-Oriented Development,” Rational TR, 1995



Kekuatan UML

- ❑ **Unified Modeling Language (UML)** adalah desain dan analisa OO yang ekuivalen dengan Java
- ❑ Penyeragaman Booch, Rumbaugh, dan Jacobson
- ❑ Materi
 - o Apakah UML itu?
 - o Tujuh tujuan UML
 - o Pemodelan konstruksi dan diagram
 - Diagram Use-Case
 - Diagram Class
 - Diagram Perilaku
 - Diagram Interaksi
 - Diagram Implementasi



Apakah UML itu?

- ❑ UML adalah bahasa untuk spesifikasi, visualisasi, konstruksi dan dokumentasi pembuatan software

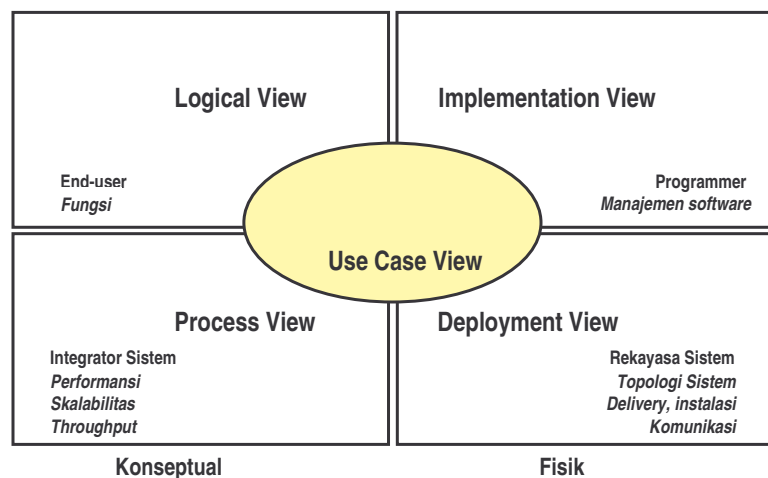


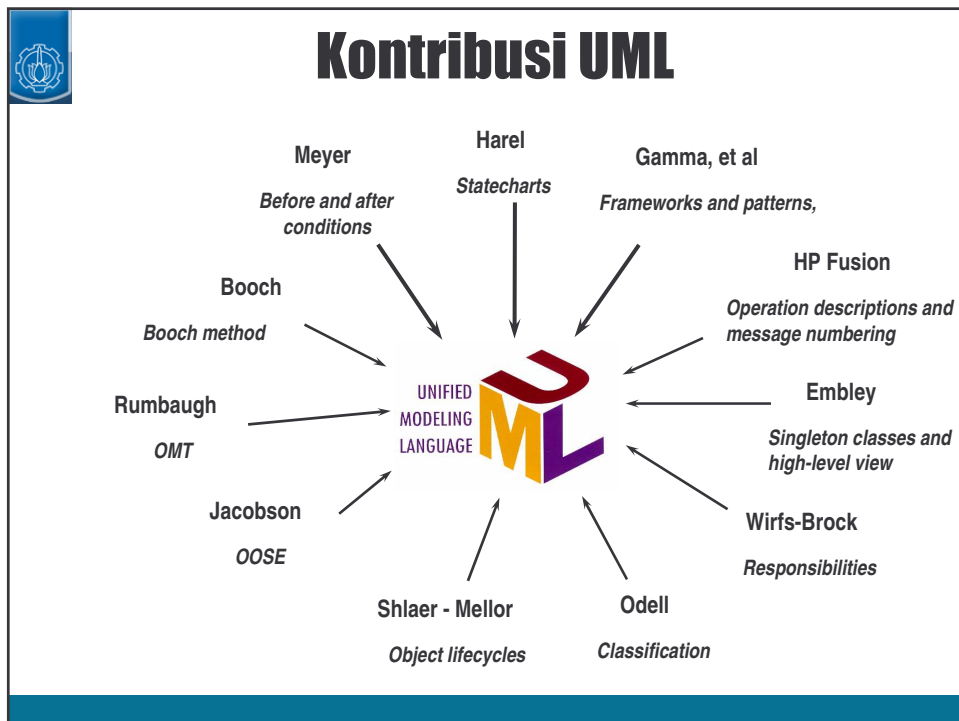
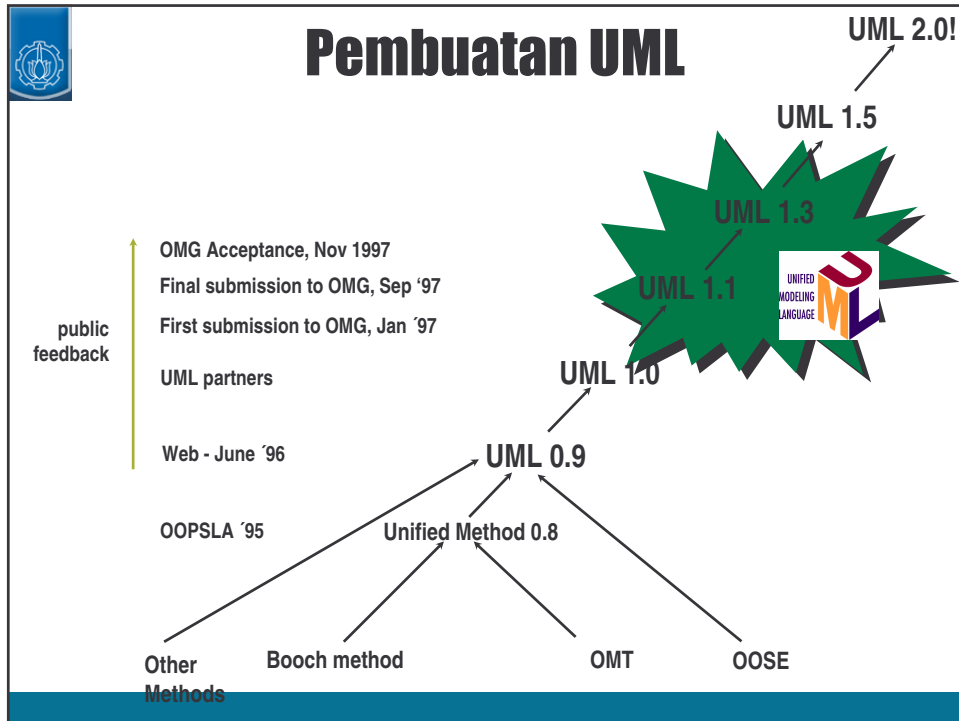
Sejarah UML

- ❑ Penyeragaman Booch dan Rumbaugh - 1994
- ❑ Versi 0.8 dikeluarkan Oktober 1995
- ❑ Ivar Jacobson dan Objectory disatukan menjadi Rational pada Fall 1995
- ❑ UML 2.0 - versi resmi - dalam fase upgrading
- ❑ UML 1.5 - versi sebelumnya - Lengkap



Representasi Arsitektur Sistem





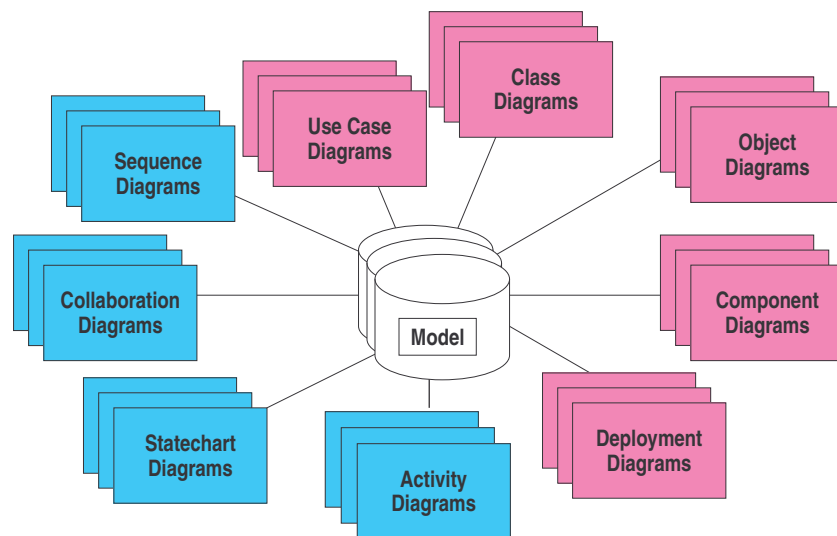


Pembangunan/Diagram Pemodelan UML Klasifikasi berdasarkan Kapabilitas/Timeline

- ☐ Diagram Use-Case
- ☐ Diagram Class dan Obyek
- ☐ Diagram Perilaku
 - o Diagram Statechart
 - o Diagram Activity
- ☐ Diagram Interaksi
 - o Sequence Diagram
 - o Collaboration Diagram
- ☐ Diagram Implementasi
 - o Component Diagram
 - o Deployment Diagram



Hubungan antara Model dan Diagram





Pemodelan Use Case



Agenda

- ☐ Apakah use case itu?
- ☐ Keuntungan Use Case
- ☐ Use Cases vs. dokumen kebutuhan
- ☐ Membangun model Use Case
 - o System
 - o Aktor
 - o Use Case
 - o Relasi Use Case
- ☐ Contoh



Apakah Use Case itu?

- ☐ Dibuat oleh Ivar Jacobson (1994)
- ☐ “use case adalah urutan transaksi dari aktor pada sistem”
- ☐ Menggambarkan APA yang sistem (“Black Box”) lakukan dari perspektif user
- ☐ Sekumpulan skenario yang menjadi satu untuk tujuan user
- ☐ Model use case BUKAN teknik pemodelan berorientasi obyek yang berjalan inherent



Keuntungan Use Case

- ☐ Menangkap kebutuhan fungsional dari perspektif user
- ☐ Memberi diskripsi jelas dan konsisten dari apa yang seharusnya dilakukan sistem
- ☐ Dasar membentuk tes sistem
- ☐ Menyediakan kemampuan untuk melacak kebutuhan fungsional ke dalam class aktual dan operasi dalam sistem
- ☐ Berfungsi sebagai unit estimasi
- ☐ Unit terkecil dari delivery
 - o Setiap increment yang direncanakan dan di-deliver digambarkan dalam use case yang akan di-deliver secara increment



Use Case vs. Kebutuhan

- ☐ Dokumen kebutuhan menyatakan apa yang seharusnya dilakukan sistem. Use case menggambarkan aksi yang dilakukan user dan respon dari sistem
- ☐ Use case biasanya digunakan untuk mengartikan kebutuhan
- ☐ Kebutuhan lebih efektif didokumentasikan sebagai use case
 - o Lebih baik dalam pelacakan
 - o Lebih mudah untuk validasi user terhadap kebutuhan fungsional
 - o Membantu struktur manual user
 - o Perangkat membangun class



Diagram Use Case pada UML

- ☐ Model use case digambarkan dalam UML sebagai satu atau lebih diagram use case
- ☐ Terdapat 4 elemen utama:
 - o **Sistem**
 - o **Aktor** yang berinteraksi pada sistem
 - o **Use case** atau servis yang tersedia pada sistem
 - o **Relasi** antar elemen



Sistem

- ☐ Bagian dari pemodelan use case, merupakan definisi batasan pengembangan sistem
- ☐ Sistem tidak perlu harus berupa software
- ☐ Mendefinisikan batasan sistem
 - o Bagian mana yang otomatis dan bagian mana yang manual?
 - o Bagian mana yang dilakukan oleh sistem lain?
 - Keseluruhan solusi yang ditampilkan dimasukkan dalam batasan sistem
 - Dapat dilakukan secara inkremental
- ☐ Sistem direpresentasikan sebagai kotak
- ☐ Nama sistem muncul dibawah atau di dalam kotak

Traffic Violations Report System

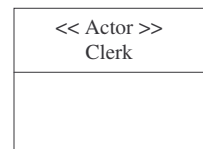
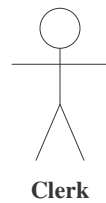


Aktor

- ☐ Seseorang atau sesuatu yang berinteraksi dengan sistem (pertukaran informasi dalam sistem)
- ☐ Aktor merupakan aturan yang dimainkan dalam sistem, bukan user individu dari sistem
- ☐ Contoh:
 - o Clerk - Menginputkan data
 - o Supervisor - mengijinkan modifikasi/menghapus data
 - o Manager - mengijinkan melihat statistik
- ☐ Satu user dapat berlaku lebih dari satu aturan
- ☐ Aktor tidak harus manusia
 - o Bisa berupa sistem eksternal yang berhubungan dengan pengembangan sistem
- ☐ Aktor mempunyai nama yang menyatakan aturan
- ☐ Use case dilakukan oleh aktor

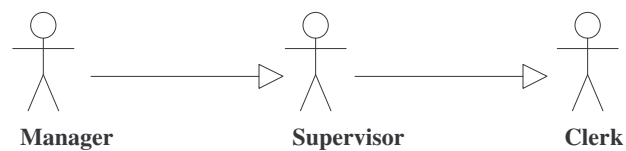


Icon Aktor



Relasi antar Aktor

- ☐ Bila beberapa aktor adalah bagian dari aturan, juga berlaku aturan yang digeneralisasi, digambarkan sebagai generalisasi
- ☐ Perilaku aturan general digambarkan dalam aktor super-class
- ☐ Aktor spesialisasi mewarisi perilaku super-class dan menambahkan beberapa cara
- ☐ Relasi antar aktor tidak selaku diperlukan





Identifikasi Aktor

- ☐ Siapa yang merupakan fungsi utama dalam sistem?
- ☐ Siapa yang memerlukan maintain, administrasi dan menjalankan sistem?
- ☐ Dengan software/hardware lain yang mana sistem perlu berinteraksi?
 - o Sistem komputer lain
 - o Aplikasi lain dari komputer yang sama (misalnya client/server)



Use Case

- ☐ Menyatakan fungsi lengkap yang dilakukan aktor
 - o Use case memenuhi tujuan aktor
- ☐ Selalu dilakukan oleh aktor
- ☐ Use case menghasilkan nilai ke aktor
- ☐ Use case lengkap
 - o Jangan membagi use case ke dalam use case yang lebih kecil yang mengimplementasikan use case lain (dekomposisi fungsional)
- ☐ Skenario use case biasanya digambarkan secara tekstual
 - o Spesifikasi sederhana dan konsisten tentang bagaimana aktor dan sistem berinteraksi
 - o Use case menggambarkan template
- ☐ Menggambarkan dimana level user dan respon sistem
 - o Bebas dari teknologi dan mekanisme detail, terutama yang berhubungan dengan antar muka user



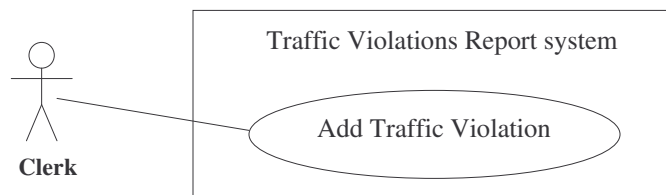
Menemukan Use Case

- ❑ Untuk setiap aktor yang sebelumnya didefinisikan:
 - o Servis apa yang dibutuhkan aktor dari sistem
 - Membaca, membuat, menghapus, memodifikasi, menyimpan informasi
 - o Event apa yang dilakukan aktor?
 - o Dapatkan pekerjaan harian dari aktor disederhanakan?
 - Tidak terkonsentrasi hanya pada sistem saat ini



Icon Use Case

- ❑ Ellips yang berisi nama use case
- ❑ Ditempatkan dalam batasan sistem yang dimodelkan
- ❑ Berhubungan setidaknya satu aktor dengan komunikasi asosiasi
 - o Kecuali untuk use case spesial/tambahan





Relasi Use Case

❑ Relasi Include relationship

- o Bila sejumlah use case mempunyai perilaku umum, perilaku ini dapat dimodelkan dalam satu use case yang digunakan oleh use case lain
- o X << includes >> Y menyatakan bahwa proses yang dilakukan X selalu melibatkan Y setidaknya satu kali
- o Berlaku sebagai dekomposisi fungsional
- o Use case include harus lengkap
- o X harus memenuhi kondisi awal dari Y sebelum memasukinya



Relasi Use Case

❑ Relasi Generalisasi

- o Digunakan bila sejumlah use case merupakan sub bagian, tetapi masing-masing mempunyai perbedaan yang tidak mungkin disatukan dalam satu use case
- o Generalisasi dan spesialisasi use case harus mempunyai tujuan yang sama
- o Use case spesialisasi menyatakan skenario alternatif dari use case generalisasi
- o Generalisasi use case harus lengkap
- o Use case spesialisasi dapat berhubungan dengan aktor baru
- o Use case spesialisasi menambahkan kondisi awal dan akhir (semantik AND)

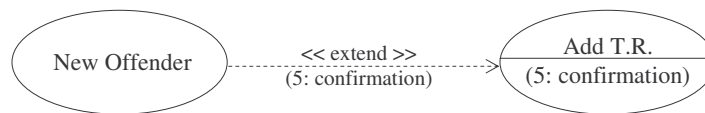




Relasi Use Case

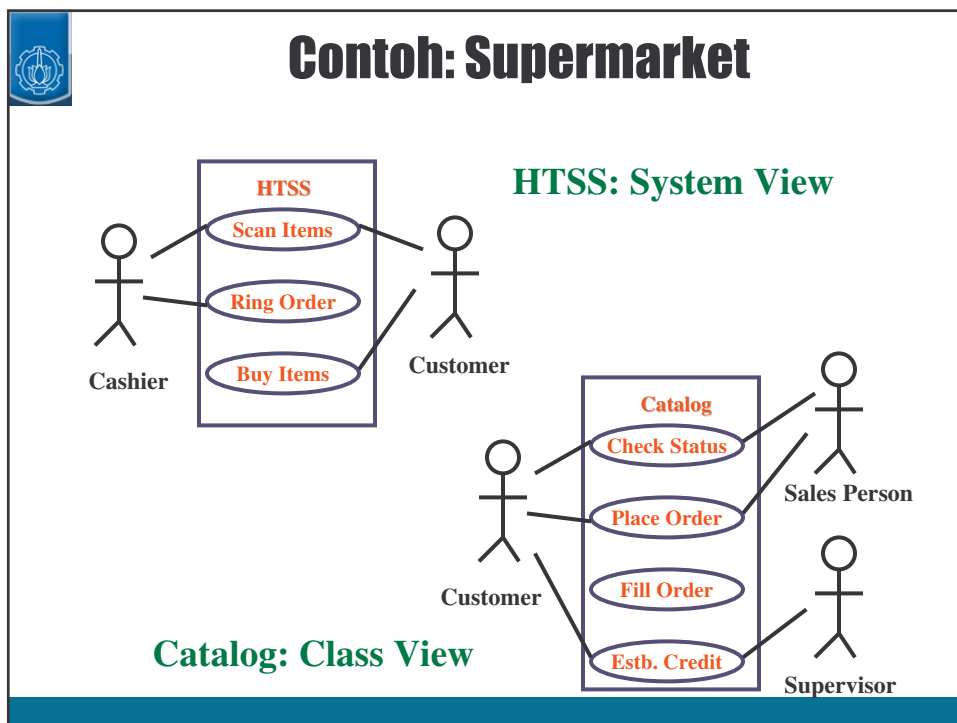
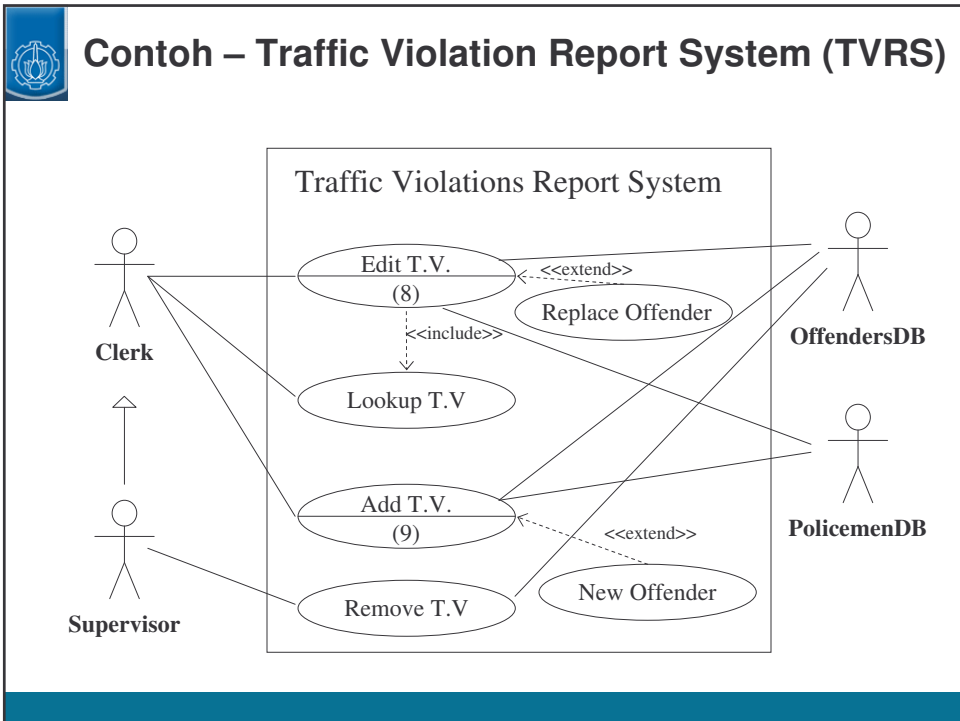
❑ Relasi Extend

- o Mirip dengan relasi generalisasi tetapi dengan aturan tambahan di dalamnya
- o Use case spesialisasi hanya **extend** use case generalisasi
- o Use case tambahan harus dideklarasikan secara eksplisit tambahannya



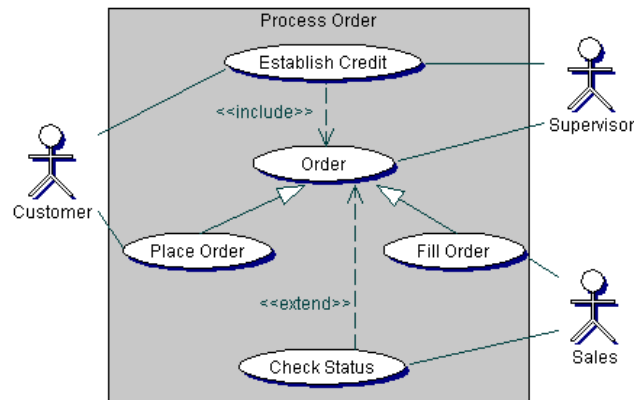
Rekomendasi Alur Kerja

1. Identifikasi aktor (dan relasinya jika diperlukan)
2. Untuk setiap aktor, identifikasi use case sampai tidak ada use case yang tertinggal
 - a. Tentukan tujuan aktor
 - b. Putuskan kegiatan utama untuk mendapatkan tujuan
 - c. Buat use case untuk setiap tujuan
 - Aktor/tujuan baru kemungkinan ditemukan
 - d. Validasi/pembetulan use case yang ada
3. Gambarkan diagram use case



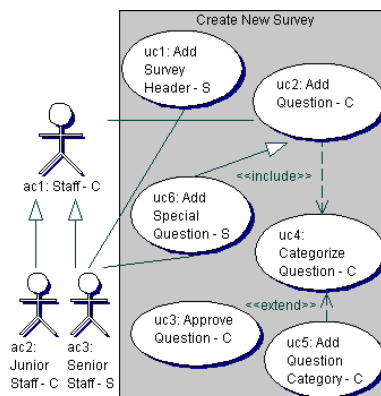


Contoh: Supermarket



Contoh: Manajemen Survey

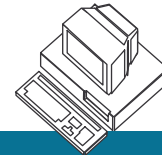
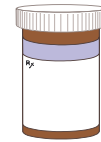
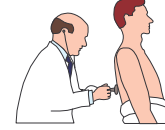
- ❑ Institusi survey yang melakukan/mengatur survey publik. Setelah data mentah dikumpulkan, staf senior menambah header survey ke database, staf senior atau junior menambah pertanyaan ke dalam survey, berupa mengelompokkan pertanyaan atau menambah katagori pertanyaan. Pertanyaan yang sensitif terbatas ke staf senior



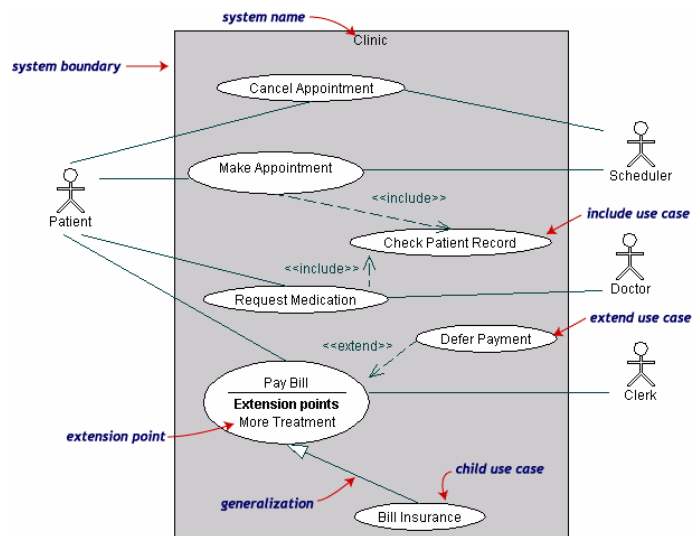


Skenario Use Case Health Care Application (HCA) - Write Rx

- ① Dokter menentukan resep untuk pasien
- ② Dokter memberikan informasi obat:
nama obat, takaran, jumlah dos dan
refill
- ③ Komputer melakukan cross-ce antara
obat dan sejarah obat/medis yang
sudah ada
- ④ Resep dikirim secara elektronik ke
apotik atau dicetak untuk pasien



Contoh: Health Care





Latihan

Buatlah diagram use case untuk rental VCD



Analisa Class



Agenda

- ☐ Tujuan
- ☐ Apakah analisa class itu?
- ☐ Tipe analisa class
 - o Class Boundary
 - o Class Entity
 - o Class Control
- ☐ Mencari analisa class
- ☐ Contoh



Tujuan

- ☐ Identifikasi kumpulan kandidat (analisis) class yang pantas digunakan sesuai perilaku yang digambarkan dalam use case
 - o Perilaku lengkap dari use case harus didistribusikan ke analisis class
 - o Kebutuhan non-fungsional tidak dimasukkan
 - o Konsentrasi pada mencari atribut dan relasi class



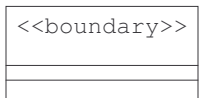
Apakah Analisis Class itu?

- ❑ Teknik mencari class menggunakan tiga perspektif sistem
 - o Batasan antara sistem dan aktor
 - o Informasi yang digunakan sistem
 - o Logika kontrol dari sistem

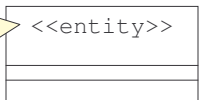
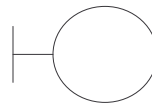


Apakah Analisis Class itu?

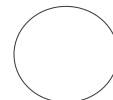
Stereotype yang mendefinisikan elemen model baru dalam hubungan dengan elemen model lain



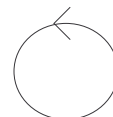
=

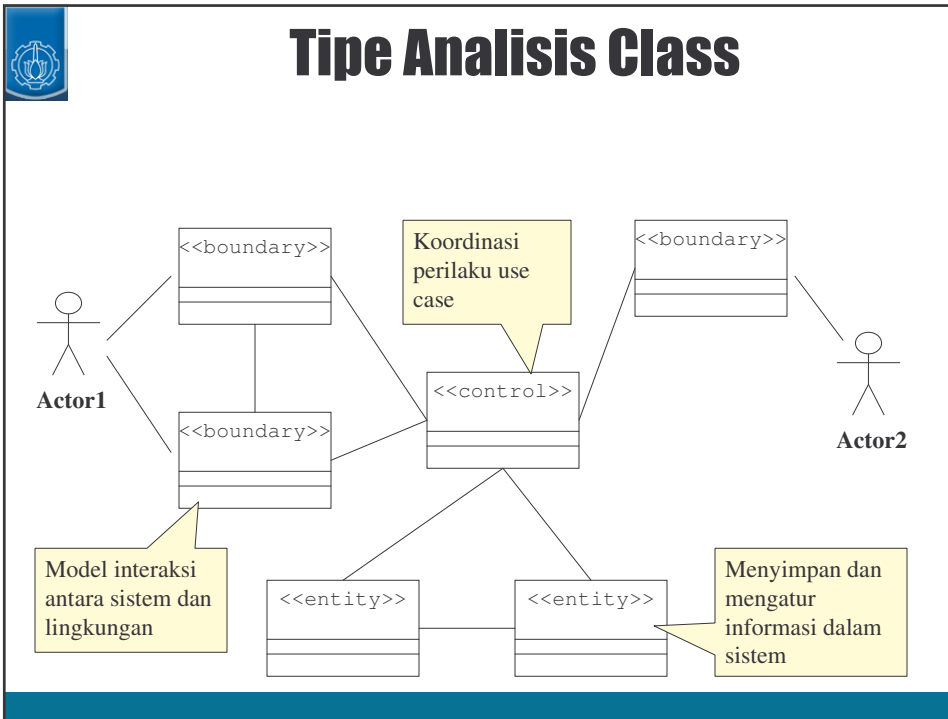


=



=





- ## Class Boundary
- ❑ Memodelkan interaksi antara sistem dan pekerjaan yang ada di dalamnya
 - o Class user interface
 - Berkonsentrasi pada informasi apa yang diberikan ke user
 - Tidak terkonsentrasi pada user interface secara detail
 - Contoh
 - ViolationsDialog
 - o Class System / Device interface
 - Konsentrasi pada protokol apa yang harus didefinisikan. Tidak terkonsentrasi pada bagaimana protokol diimplementasikan
 - ❑ Lingkungan class boundary tergantung pada:
 - o User interface
 - o Komunikasi protokol



Class Entity

- ☐ Memodelkan konsep utama dari sistem
- ☐ Biasanya informasi model bersifat persistent
- ☐ Berisi logika untuk pemecahan masalah sistem
- ☐ Tidak tergantung pada lingkungan
- ☐ Dapat digunakan banyak use case



Class Control

- ☐ Mengontrol dan mengkoordinasikan perilaku use case
- ☐ Mendelegasikan pekerjaan dari use case ke class
 - o Class control memberitahu class lain untuk mengerjakan sesuatu dan tidak pernah mengerjakan sesuatu kecuali untuk atas perintah class control
- ☐ Class control merupakan gabungan class boundary dan entity
- ☐ Meskipun, terdapat satu class control untuk satu use case
- ☐ Tergantung pada use case dan lingkungan
- ☐ Contoh:
 - o ViolationsController



Mencari Analisis Class

1. Untuk setiap use case:
 - a. Tentukan deskripsi use case
 - b. Identifikasi class boundary, entity an kontrol
 - c. Untuk setiap class identifikasikan
 - Atribut
 - Relasi
2. Validasi model, ulangi proses jika perlu



Identifikasi Class

- ☐ Class mungkin tersembunyi dalam:
- o Dokumen kebutuhan
 - o Model use case
 - o Permintaan stakeholder
 - o Domain permasalahan
 - o Dokumentasi proyek



Identifikasi Class

- ☐ Class boundary
 - o Setidaknya terdapat satu class boundary untuk setiap aktor / pasangan use case
- ☐ Class control
 - o Biasanya, terdapat satu class control per use case
 - o Jika ada dua class control yang sejenis, use case yang berhubungan harus digabung
 - Contoh: “manage traffic report” menggantikan use case “edit/add/remove traffic report”
- ☐ Class Entity
 - o Diidentifikasi dengan menggunakan kata benda dan frasa kata benda untuk menggambarkan permasalahan, dokumen kebutuhan, use case dan dokumentasi lain
 - o Kata benda berupa:
 - Obyek
 - Deskripsi state obyek (atribut)
 - Aktor
 - Bukan salah satunya



Eliminasi Kata Benda

- ☐ Class duplikat
 - o Perbedaan hanya pada nama : “System”, “TVRS”...
- ☐ Class yang tidak berhubungan
 - o Class yang tidak mengerjakan sesuatu dalam sistem (solusi): “police headquarters”...
- ☐ Atribut / operasi
 - o Beberapa kata benda yang sejenis dimodelkan sebagai atribut atau operasi daripada sebagai class: “ID”, “name”, “Report Lookup”...
- ☐ Aturan
 - o Beberapa kata benda sebagai obyek yang terlibat hubungan daripada sebagai class
 - Contoh: “Teaching Assistant” dan “Student” mungkin mempunyai aturan yang berbeda dari class “Person”
- ☐ Kata benda abstrak
 - o “identifikasi ide atau kuantitas yang tidak ada secara fisik”
 - o Jarang berhubungan dengan analisa class, tetapi sebagai atribut: “Request”, “Opinion”



Mencari Atribut Class

- ☐ Properti atau karakteristik dari class yang diidentifikasi
 - o Informasi yang ada pada class
 - o Atomik
- ☐ Kata benda yang tidak menjadi class
 - o Informasi berupa nilai yang penting untuk solusi
 - o Informasi yang secara unik dimiliki oleh obyek



Mencari Relasi

- ☐ Asosiasi biasanya berhubungan dengan kata kerja atau frasa kata kerja
 - o Lokasi fisik: next to, above, inside...
 - o Aksi langsung: drives, creates, manages...
 - o komunikasi: talks to, listens, notifies...
 - o Kepemilikan: has, part of, belongs to, contained...
 - o Lainnya: works for, married to, studies at...
- ☐ Eliminasi asosiasi yang tidak berhubungan dengan permasalahan / solusi
- ☐ Asosiasi biasanya ditemukan menggunakan diagram interaksi



Contoh: Analisa Class TVRS

- ❑ Daftar kata benda dari kebutuhan dan use case TVRS (Kandidat class entity):

Traffic report	Offender	ID
Supervisor	Policeman	Password
Report lookup	Vehicle number	Police headquarters
Confirmation	License number	Shutdown
TVRS	Fault	Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	Traffic Violation
System	Violation	Clerk



Eliminasi Class Duplikat

Traffic report	Offender	ID
Supervisor	Policeman	Password
Report lookup	Vehicle number	Police headquarters
Confirmation	License number	Shutdown
TVRS	Fault	Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	Traffic Violation
System	Violation	Clerk



Eliminasi Class Duplikat

Traffic report	Offender	ID
Supervisor	Policeman	Password
Report lookup	Vehicle number	Police headquarters
Confirmation	License number	Shutdown
TVRS	Fault	Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	Traffic Violation
System	Violation	Clerk

Clerk dan
Supervisor
Diganti
dengan User



Eliminasi Class yang tidak Relevan

Traffic report	Offender	ID
User	Policeman	Password
Report lookup	Vehicle number	Police headquarters
Confirmation	License number	Shutdown
TVRS		Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	
	Violation	



Eliminasi Class yang tidak Relevan

Traffic report	Offender	ID
User	Policeman	Password
Report lookup	Vehicle number	Police headquarters
Confirmation	License number	Shutdown
TVRS		Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	
	Violation	



Eliminasi atribut dan operasi

Traffic report	Offender	ID
User	Policeman	Password
Report lookup	Vehicle number	
Confirmation	License number	Shutdown
TVRS		Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	
	Violation	



Eliminasi atribut dan operasi

Traffic report	Offender	ID
User	Policeman	Password
Report lookup	Vehicle number	
Confirmation	License number	Shutdown
TVRS		Date
Offender Details Form	Traffic policeman	Speed
Traffic report addition	Commander	
	Violation	



Eliminasi kata benda abstrak

Traffic report	Offender
User	Policeman
Confirmation	
TVRS	
Offender Details Form	Traffic policeman
	Violation



Eliminasi kata benda abstrak

Traffic report

Offender

User

Policeman

Confirmation

TVRS

Offender Details Form

Traffic policeman

Violation



Class Entity (partial)

- ☐ Traffic report
- ☐ User
- ☐ TVRS
- ☐ Offender Details Form
- ☐ Offender
- ☐ Policeman
- ☐ Traffic policeman
- ☐ Violation
- ☐ ...

Daftar yang tersisa biasanya berisi class yang **bukan** class entiti, seperti "Offender Details Form". Class boundary dan control lebih mudah ditemukan dengan analisa use case secara langsung.



Class Boundary

- ☐ ReportDetailsForm
- ☐ PolicemanDetailsForm
- ☐ LookupReportForm
- ☐ ConfirmationDialog
- ☐ OffendersDBProxy
- ☐ PolicemanDBProxy
- ☐ ...

database proxy
merupakan API
level tinggi dan
membungkus
komunikasi dan
bahasa
pemrograman
secara detail

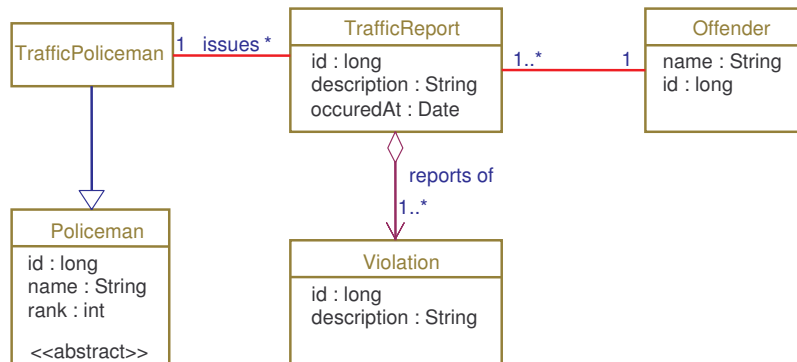


Class Control

- ☐ AddReportController
- ☐ RemoveReportController
- ☐ LookupReportController
- ☐ EditReportController
- ☐ AuthenticationController
- ☐ ...



Analisis Diagram Class I



Analisis Diagram Class II

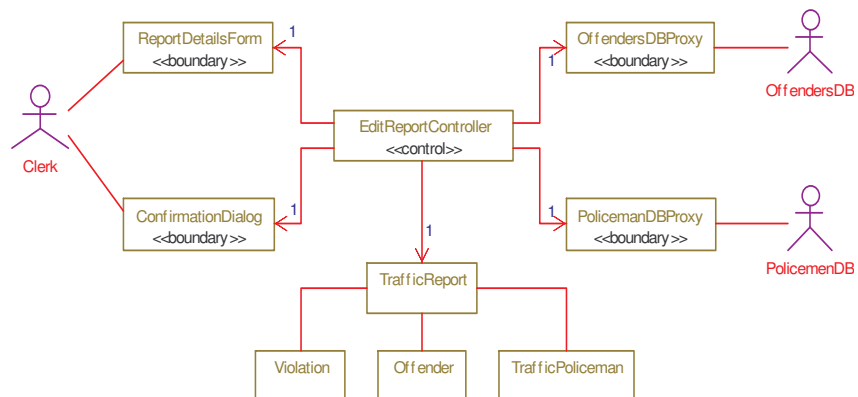




Diagram Class



Agenda

- ☐ Apakah class itu?
- ☐ Relationship dan Multiplicity
- ☐ Contoh diagram class
- ☐ Package
- ☐ Contoh package
- ☐ Diagram obyek



Apakah Class itu?

- ❑ Class adalah gambaran himpunan obyek yang menggunakan atribut, operasi, metode, relasi dan semantik yang sama
- ❑ Class direpresentasi secara grafis sebagai kotak untuk
 - o Nama Class, atribut Private dan operasi Public
 - o Properti, responsibiliti, aturan, sejarah modifikasi dll.
- ❑ Desainer mengembangkan Class sebagai himpunan ruang yang berkembang sepanjang waktu untuk meningkatkan fungsionalitas dan fitur

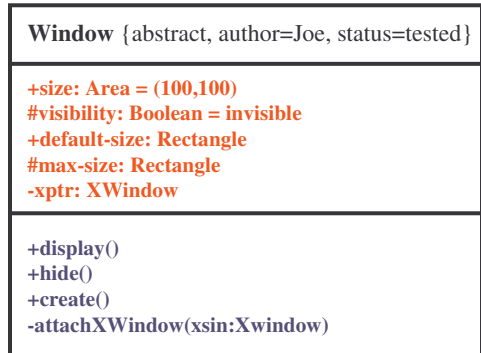


Relationship dan Multiplicity

- ❑ Relationship:
 - o Asosiasi - antar 2 class jika anggota satu class harus mengetahui anggota class lain untuk melakukan pekerjaan
 - o Agregasi - asosiasi dimana satu class milik sekumpulan class
 - o Generalisasi - hubungan pewarisan yang menyatakan satu class adalah superclass dari class lain
- ❑ Multiplicity
 - o 0..1 nol atau satu anggota
 - o $n \dots m$ mengindikasikan anggota n ke m
 - o 0..* atau * tidak ada batasan jumlah anggota (termasuk tanpa anggota)
 - o 1 tepat satu anggota
 - o 1..* paling sedikit satu anggota



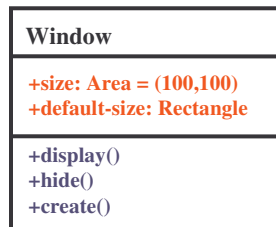
Contoh Diagram Class



What do +, #, - Represent?

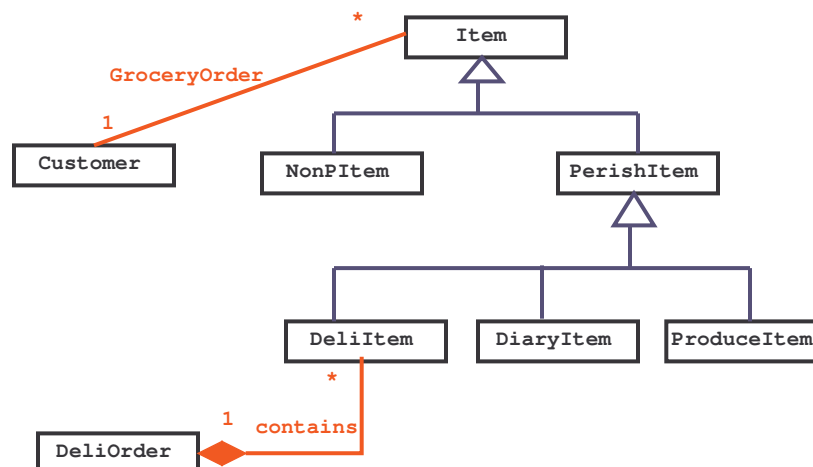
- + Public**
- # Protected**
- Private**

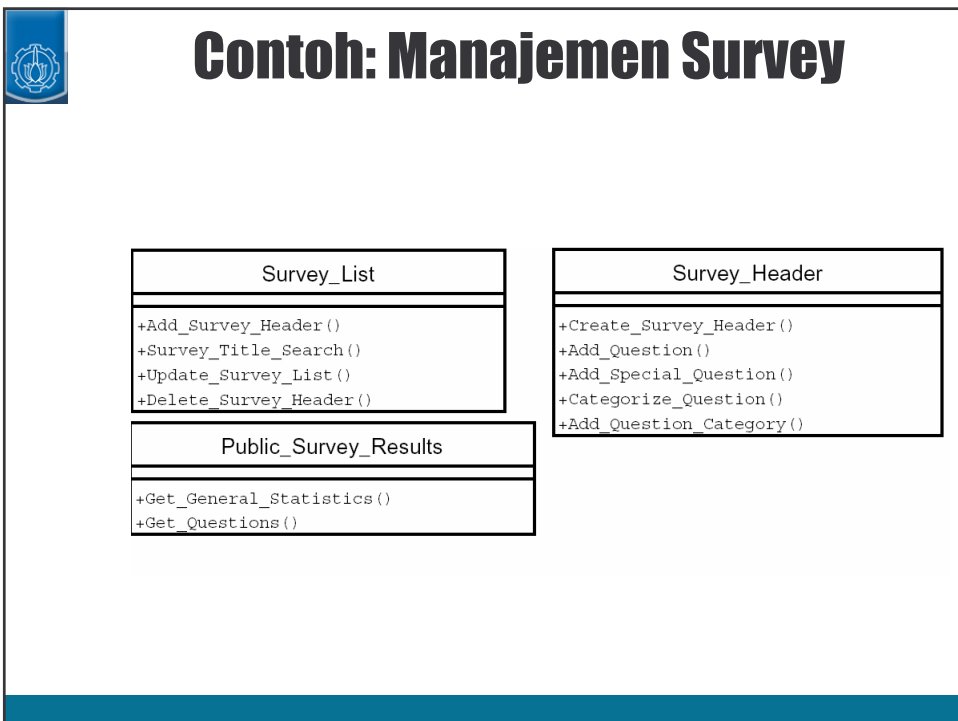
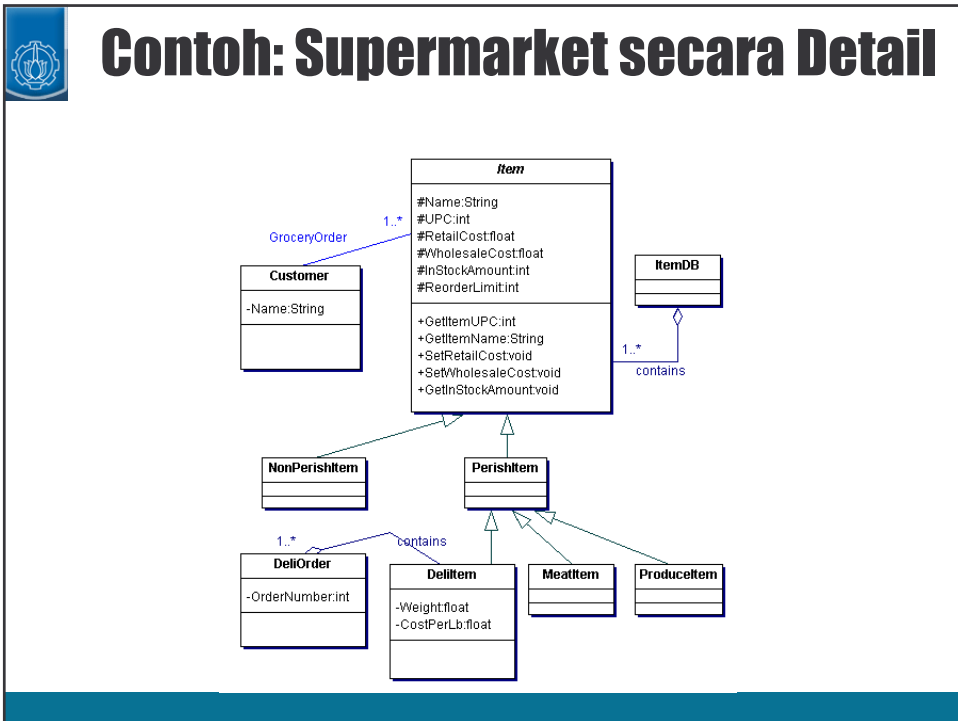
Providing Specialized Views

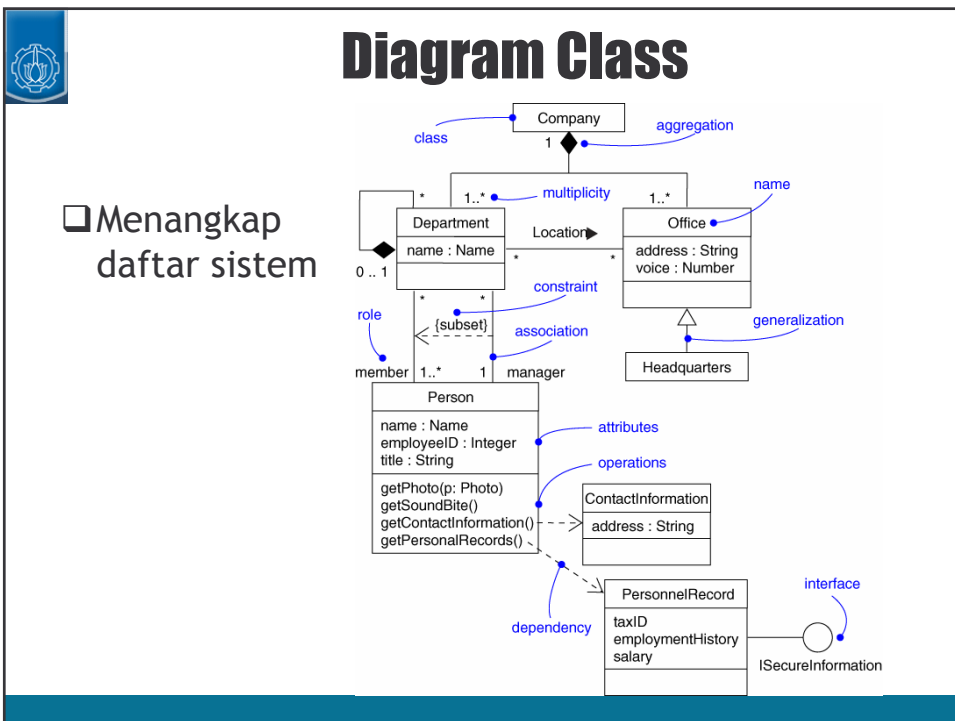
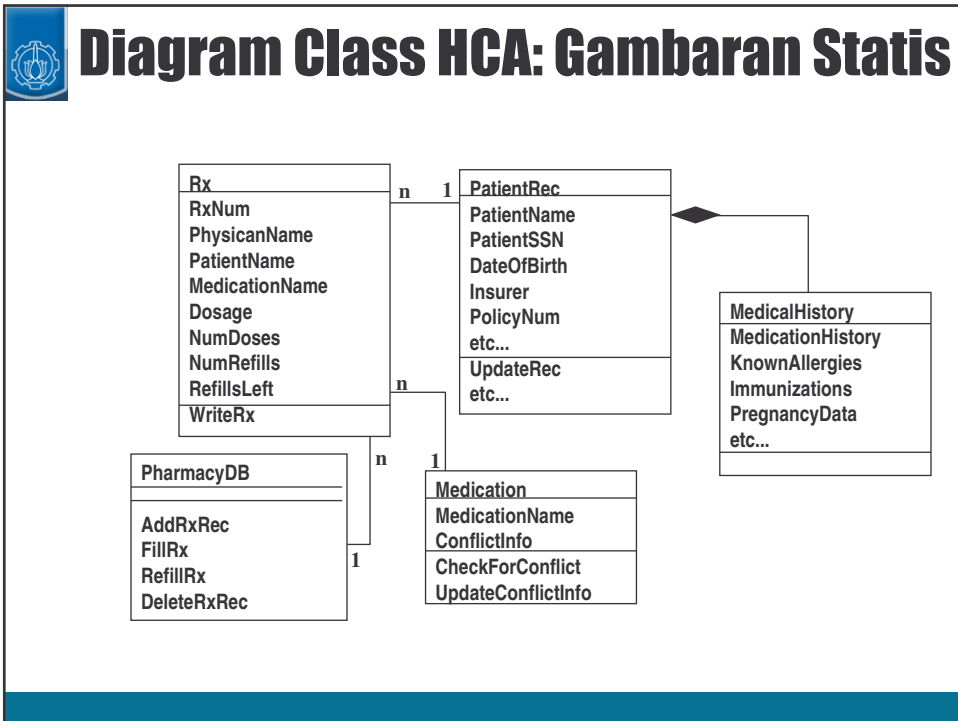


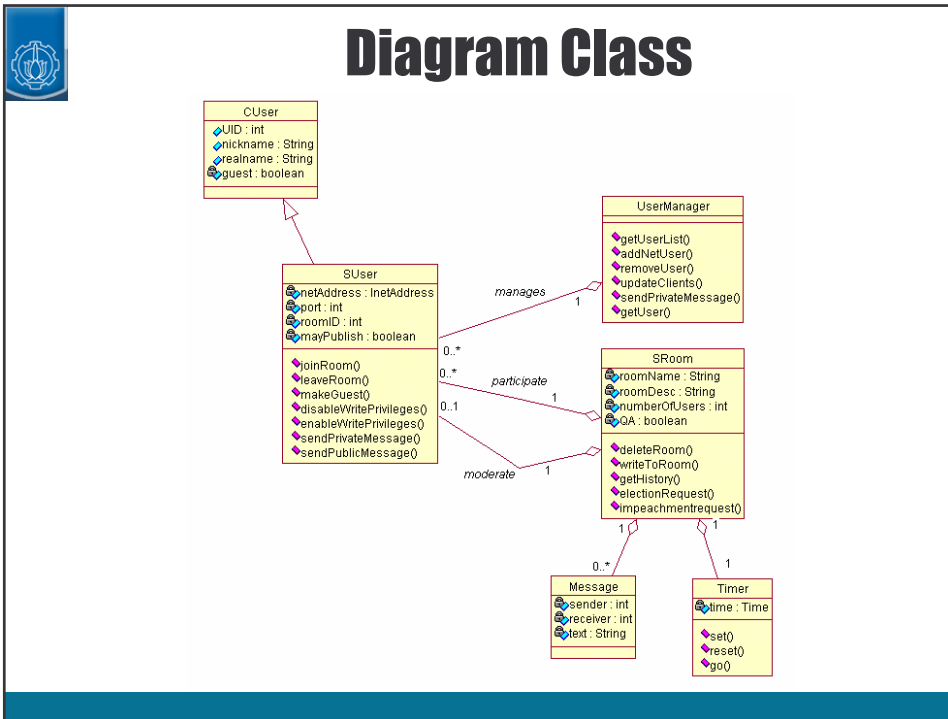
Generalisasi dan Asosiasi

Contoh: Supermarket

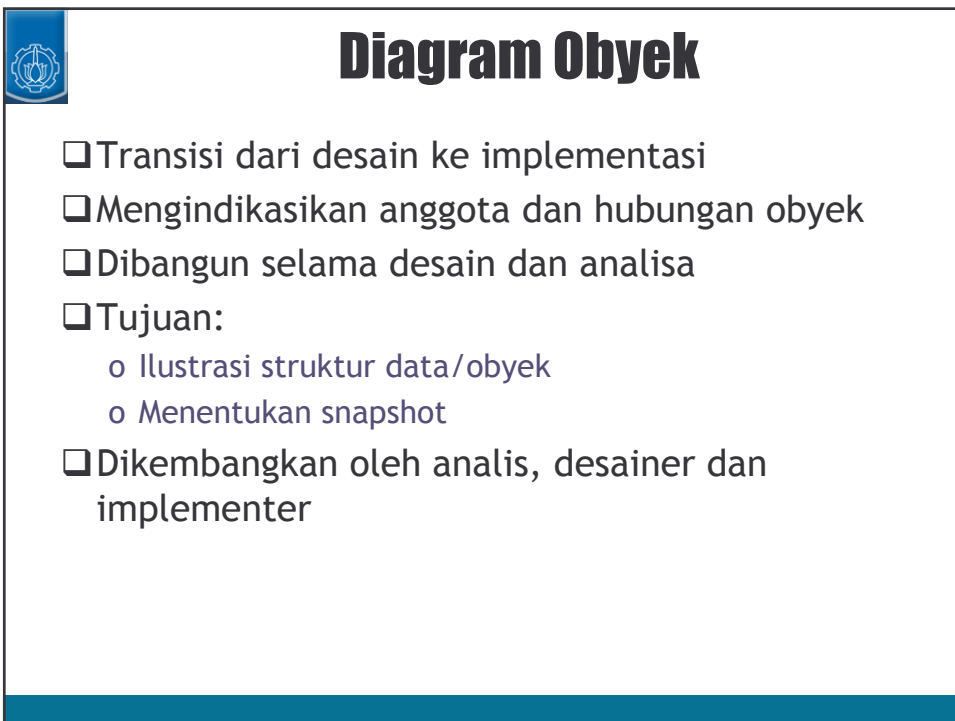
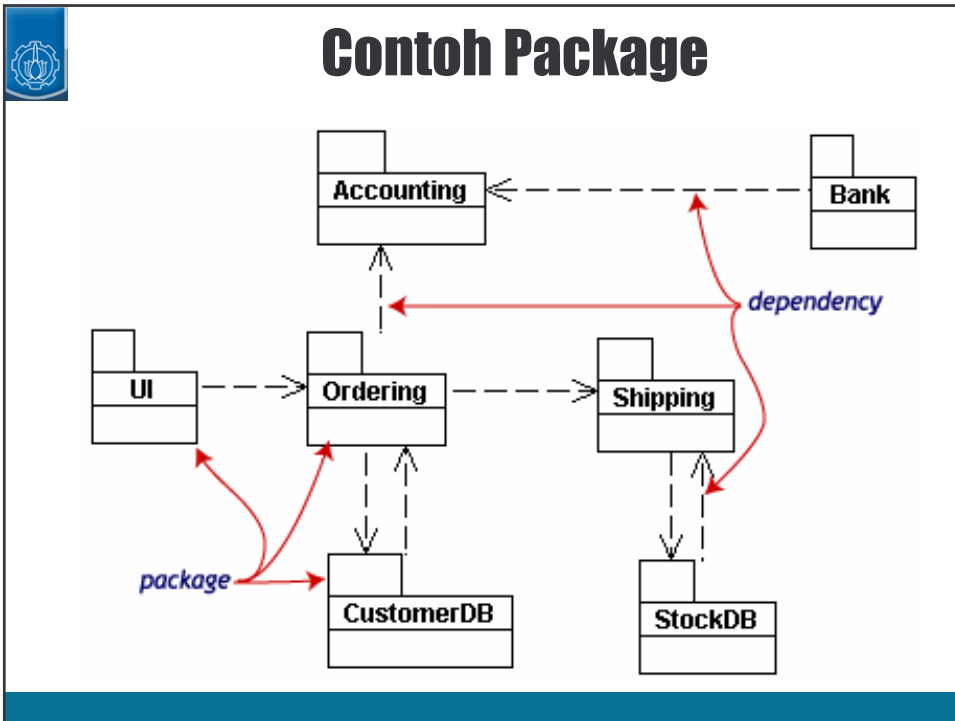








- ## Package pada Diagram Class
- ☐ Diagram class kompleks bersifat abstrak
 - ☐ Package terdiri dari beberapa class dan diasosiasikan dan dihubungkan satu sama lain
 - o Panah Dependency dalam bentuk garis putus-putus
 - o Mengindikasikan bahwa satu Package bergantung pada Package lain
 - o Perubahan tujuan (kepala panah) menyebabkan perubahan asal (ekor panah)
 - ☐ Mendukung konsep arsitektur software yang belum sempurna
 - ☐ Tetapi, tidak ada pemeriksaan/pelaksanaan dependency dalam diagram berikutnya





Perilaku Antar muka jejak

Diagram Class

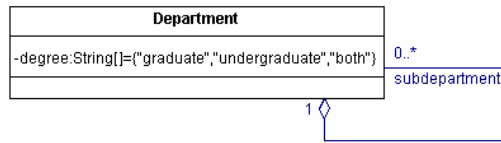


Diagram Anggota

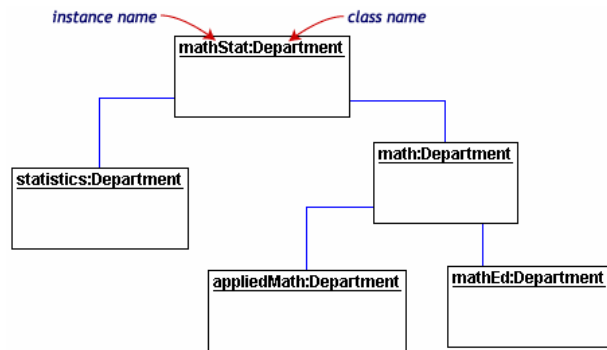
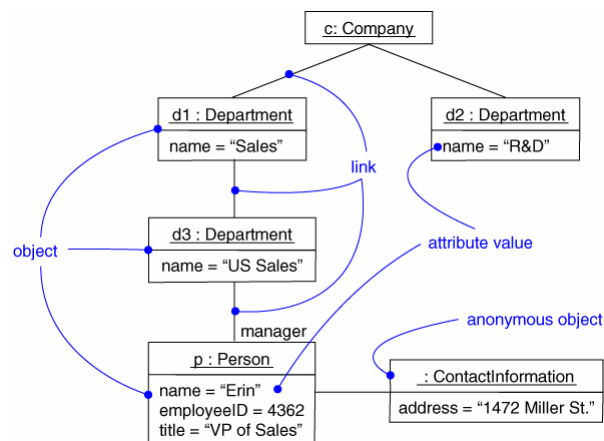


Diagram Obyek

Menangkap Instance dan Link





Latihan

Buatlah diagram class untuk rental VCD



Sequence Diagram



Agenda

- ☐ Diagram Interaksi
- ☐ Apakah Sequence Diagram?
- ☐ Obyek
- ☐ Message
- ☐ Kontrol Informasi
- ☐ Contoh



Diagram Interaksi

- ☐ Beberapa diagram untuk menggambarkan *perilaku dinamis* pada sistem berorientasi obyek
 - o Beberapa pesan ditukas pada beberapa obyek dalam konteks untuk mencapai tujuan
- ☐ Biasanya digunakan pada model dimana use case direalisasikan ke dalam urutan pesan antar obyek



Diagram Interaksi

- ❑ Tujuan diagram interaksi adalah:
 - o Model interaksi antar obyek
 - o Menuntun untuk mengerti bagaimana sistem (use case) bekerja
 - o Verifikasi bahwa diskripsi use case didukung oleh class yang ada
 - o Identifikasi tanggung jawab/operasi dan menyatakannya dalam class



Diagram Interaksi

- ❑ Pada UML, ada 2 macam
 - o Collaboration Diagram
 - Penekanan pada struktur relasi antar obyek
 - o Sequence Diagram
 - Penekanan pada pertukaran pesan pada obyek

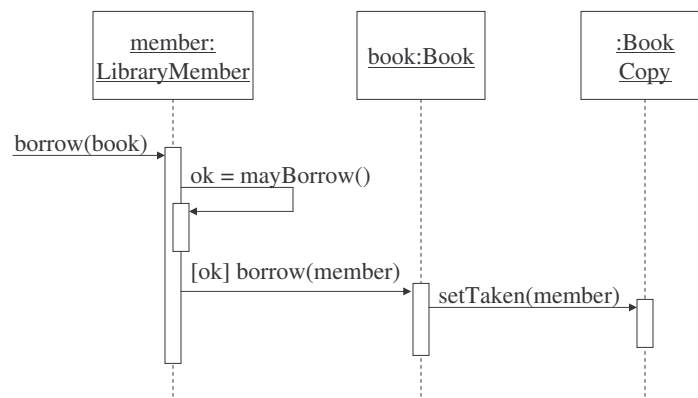


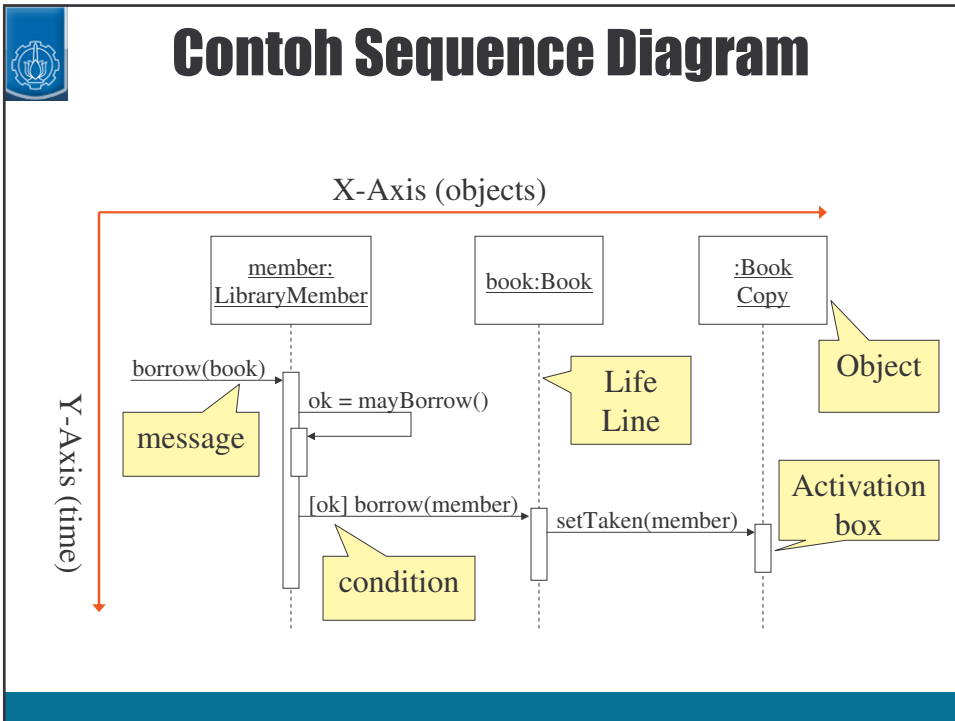
Apakah Sequence Diagram itu?

- ❑ Menggambarkan bagaimana obyek saling berinteraksi
- ❑ Penekanan pada urutan waktu pesan
- ❑ Dapat memodelkan secara sederhana urutan kerja, pencabangan, iterasi, rekursi dan konkurensi



Contoh Sequence Diagram





Obyek

- ❑ Penamaan Obyek:
 - o Sintak : *[NamaAnggota][:NamaClass]*
 - o Nama class konsisten dengan diagram class (class yang sama)
 - o Berisi nama anggota jika obyek diacu atau jika beberapa obyek dengan tipe yang sama ada dalam diagram
- ❑ Garis putus-putus (*Life-Line*) menyatakan obyek digunakan selama interaksi

```

sequenceDiagram
    participant myBirthdy as myBirthdy :Date
  
```



Pesan

- ❑ Interaksi antara 2 obyek dibentuk sebagai pesan yang dikirim dari satu obyek ke obyek yang lain
 - o Biasanya diimplementasikan dengan operasi pemanggilan yang sederhana
 - o Dapat berupa pesan aktual dikirim melalui beberapa mekanisme komunikasi, baik melalui network atau secara internal dalam komputer
 - Komunikasi antar proses (pengiriman sinyal, ...)
 - Remote Procedure Call (RMI, CORBA, ...)
- ❑ Jika obyek obj_1 mengirim pesan ke obyek lain obj_2 , harus terdapat hubungan antar dua obyek tersebut
 - o Ketergantungan struktural
 - o obj_2 adalah daerah global dari obj_1
 - o obj_2 adalah daerah lokal dari obj_1 (method argument)
 - o obj_1 dan obj_2 adalah obyek yang sama








Pesan

- ❑ Pesan digambarkan sebagai panah antara life line dari 2 obyek
 - o Pemanggilan diri sendiri diperbolehkan
 - o Waktu yang dibutuhkan oleh obyek penerima untuk memproses pesan dinyatakan dengan *activation-box*.
- ❑ Pesan dilabeli secara minimal dengan nama pesan
 - o Argumen dan informasi kontrol (kondisi, iterasi) kemungkinan dimasukkan
 - o Lebih dipilih menggunakan diskripsi tekstual bila aktor adalah sumber atau target pesan



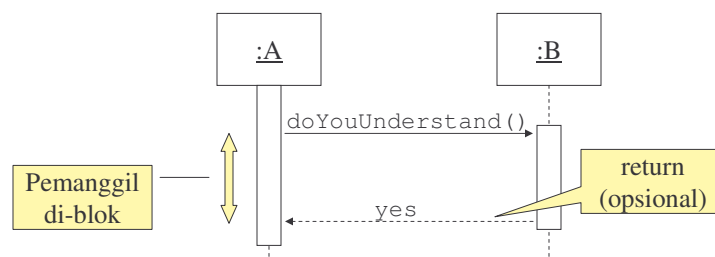
Tipe Pesan

Synchronous	
Asynchronous	
Simple	
Create	
Destroy	



Pesan Synchronous

- ❑ Sekumpulan aliran kontrol, biasanya diimplementasikan sebagai pemanggilan operasi
 - o Rutin yang menangani pesan dilengkapi sebelum pemanggil melakukan eksekusi





Nilai Return

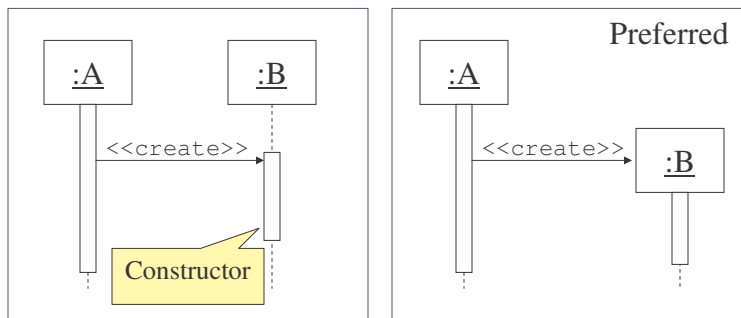


- ❑ Digunakan opsional dengan panah putus-putus yang dilabeli nilai return
 - o Jangan memodelkan nilai return jika ternyata dikembalikan, misalnya `getTotal()`
 - o Modelkan nilai return hanya jika diperlukan untuk diacu oleh tempat yang lain, misalnya sebagai parameter yang dilewatkan ke pesan yang lain
 - o Lebih dipilih memodelkan nilai return sebagai bagian dari method invocation, seperti `ok = isValid()`



Pembuatan Obyek

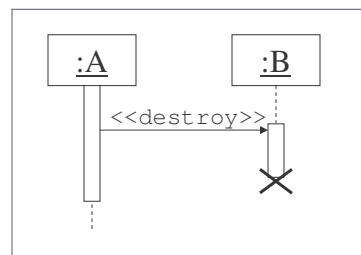
- ❑ Obyek dapat membuat obyek lain melalui pesan `<<create>>`





Penghapusan Obyek

- ❑ Obyek dapat menghapus obyek lain melalui pesan `<<destroy>>`
 - o Obyek dapat menghapus dirinya sendiri
 - o Menghindari memodelkan panghapusan obyek kecuali manajemen memori kritis



Pesan Asynchronous

- ❑ Digunakan untuk memodelkan sistem yang konkuren
- ❑ Pemanggil harus menunggu pesan ditangani sebelum melanjutkan eksekusi
 - o Seperti jika menghasilkan nilai segera
- ❑ *Obyek aktif* mempunyai thread eksekusi dan dapat menginisialisasi aktifitas kontrol
- ❑ Pesan asynchronous dapat
 - o Membuat thread baru (daftar aktifikasi baru)
 - o Membuat obyek baru
 - o Komunikasi dengan thread yang sedang berjalan



Informasi Kontrol

❑ Kondisi

- o Sintak: '[' ekspresi ']' label-pesan
- o Pesan dikirim hanya jika kondisi benar
- o Contoh: [ok] borrow(member) →

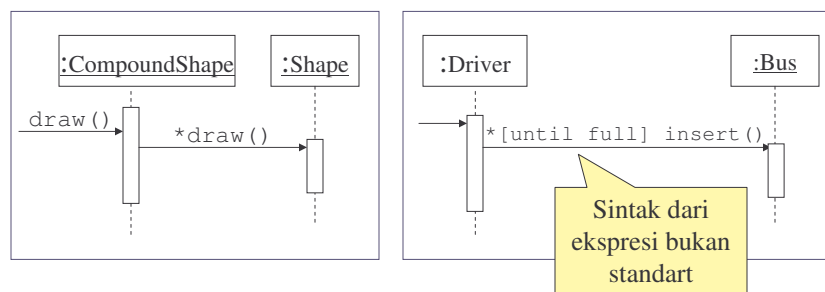
❑ Iterasi

- o Sintak: * ['[' ekspresi ']'] label-pesan
- o Dikirim beberapa kali sehingga dimungkinkan untuk obyek penerima lebih banyak



Informasi Kontrol

❑ Contoh iterasi:



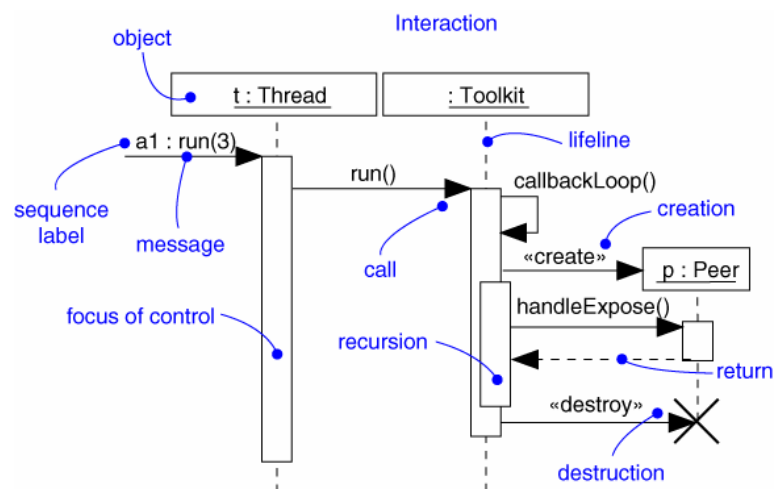


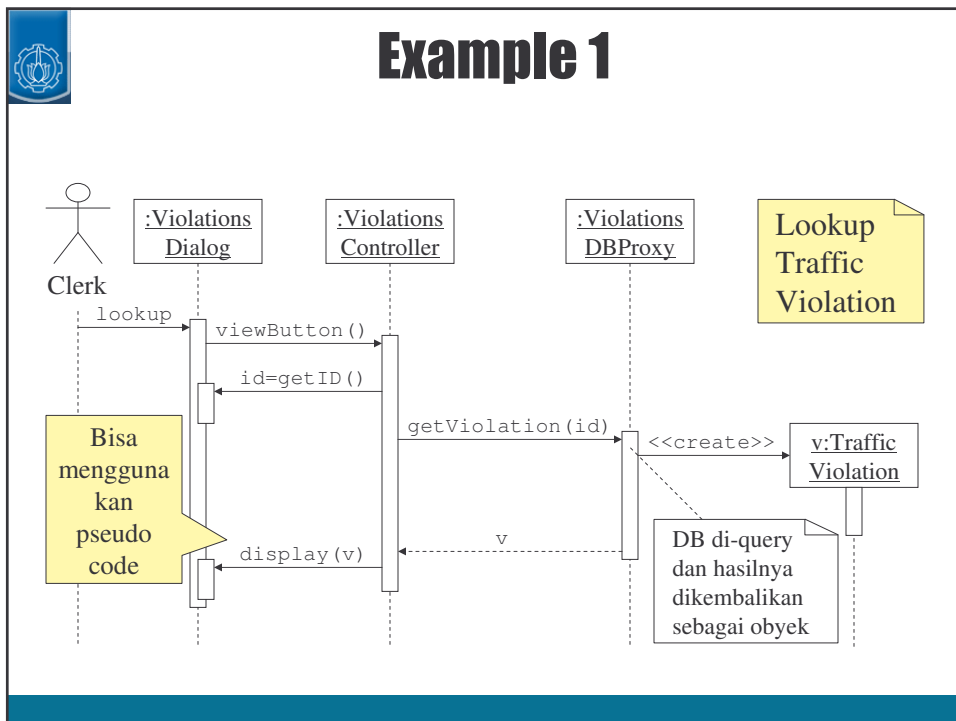
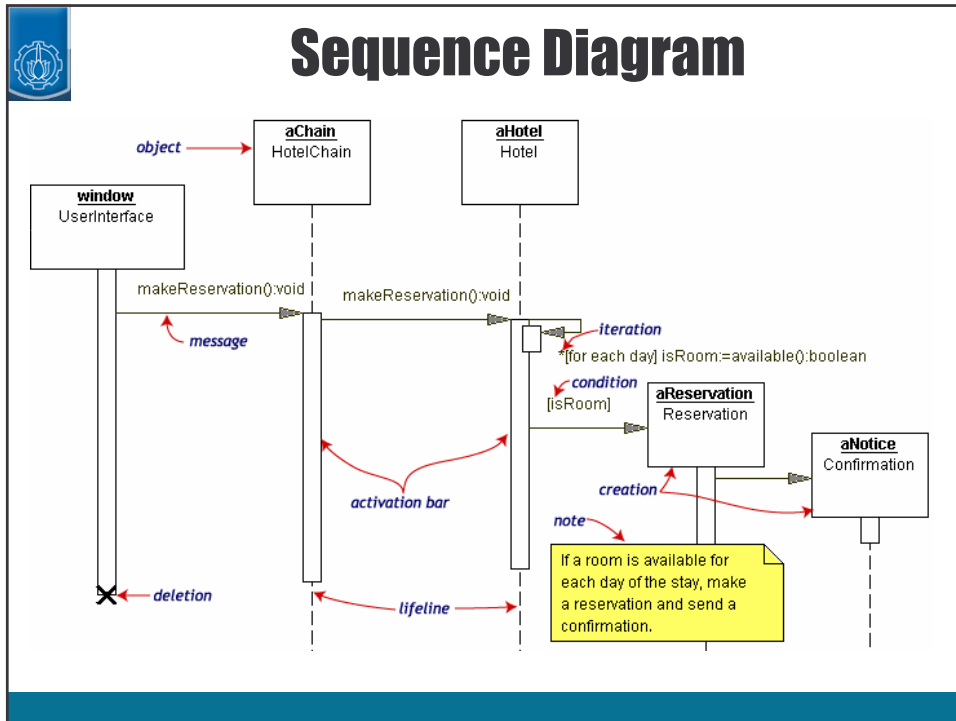
Informasi Kontrol

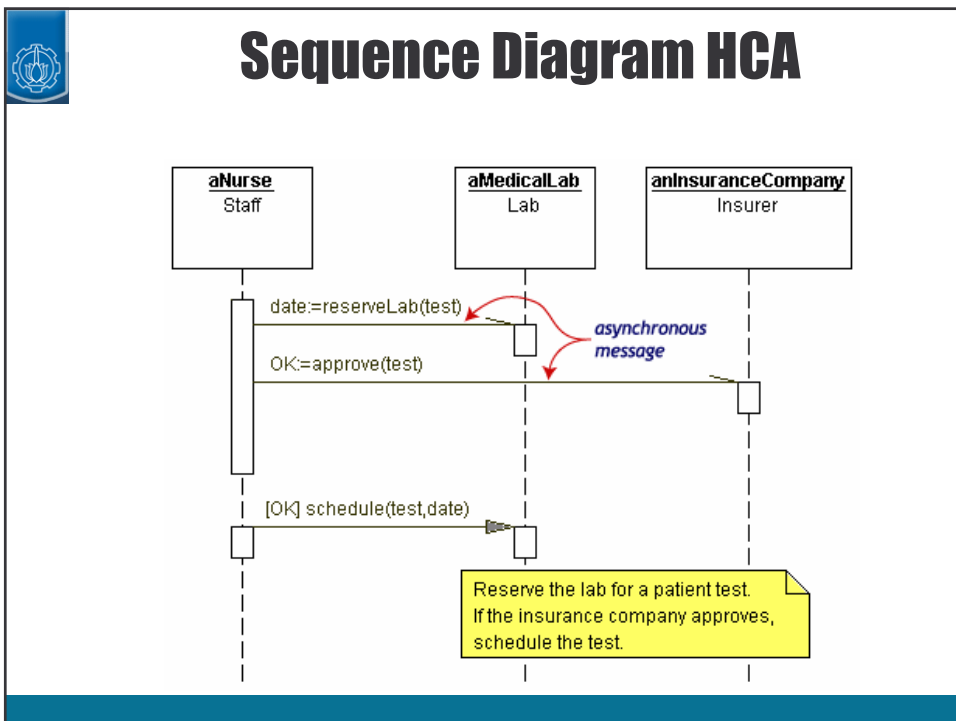
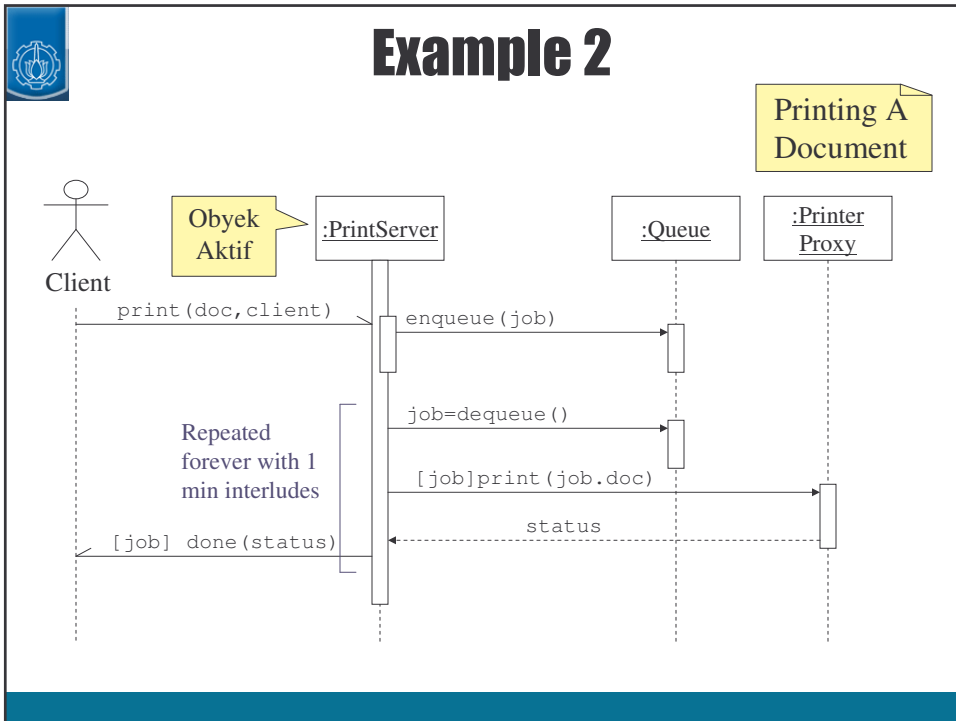
- ❑ Mekanisme kontrol dari sequence diagram hanya cukup untuk pemodelan alternatif sederhana
 - o Perlu membuat beberapa diagram untuk skenario pemodelan yang kompleks
 - o Jangan menggunakan sequence diagram untuk pemodelan algoritma secara detail (lebih baik menggunakan *diagram aktifitas*, *pseudo-code* atau *diagram state-chart*)

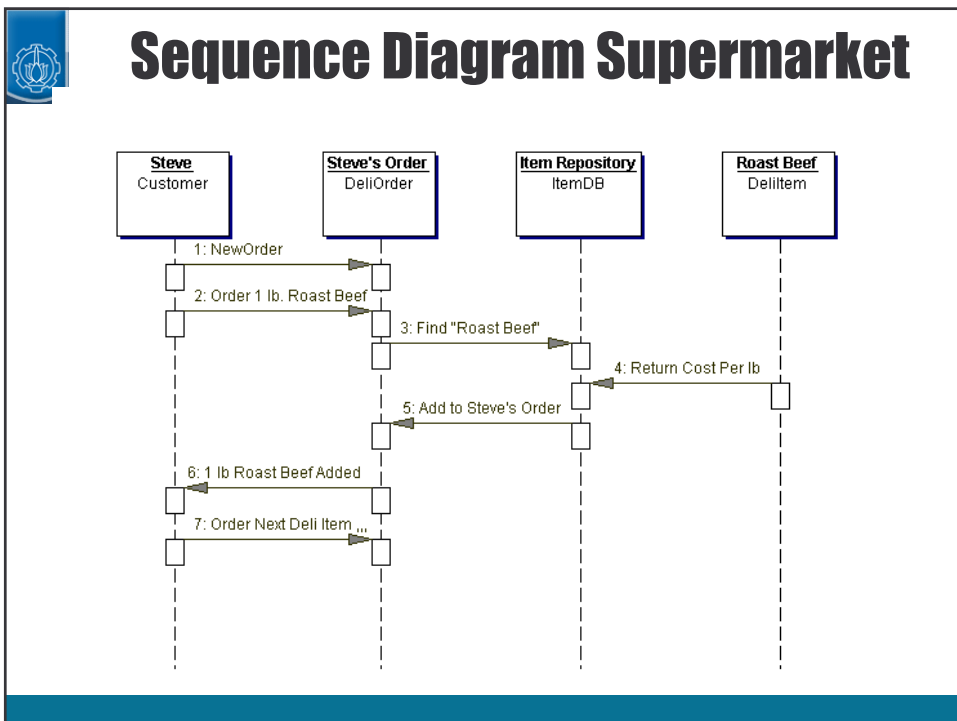
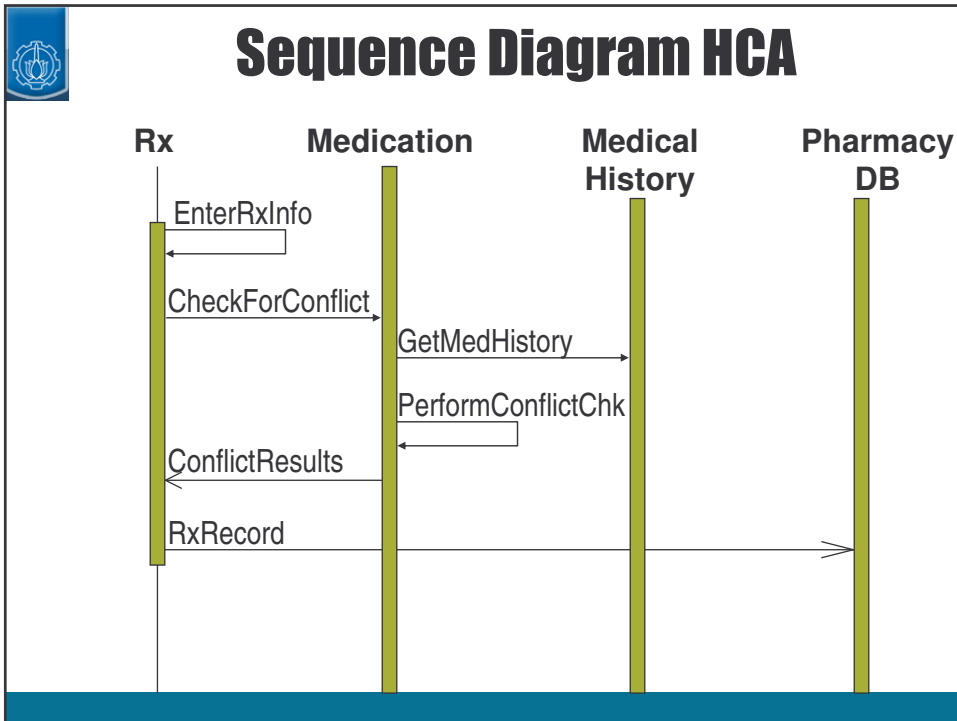


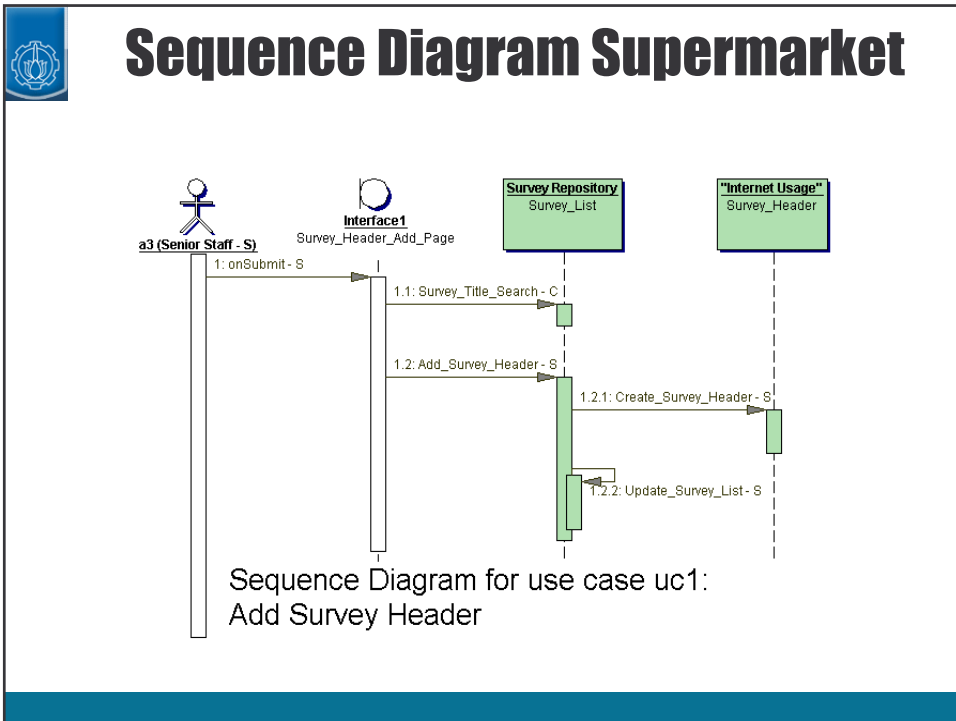
Sequence Diagram











Latihan

Buatlah sequence diagram untuk rental VCD



Collaboration Diagram

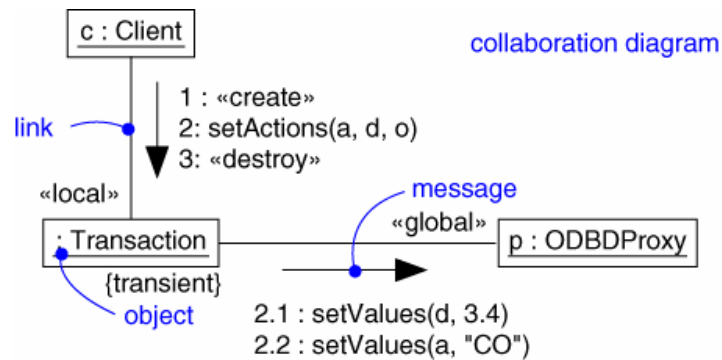


Collaboration Diagram

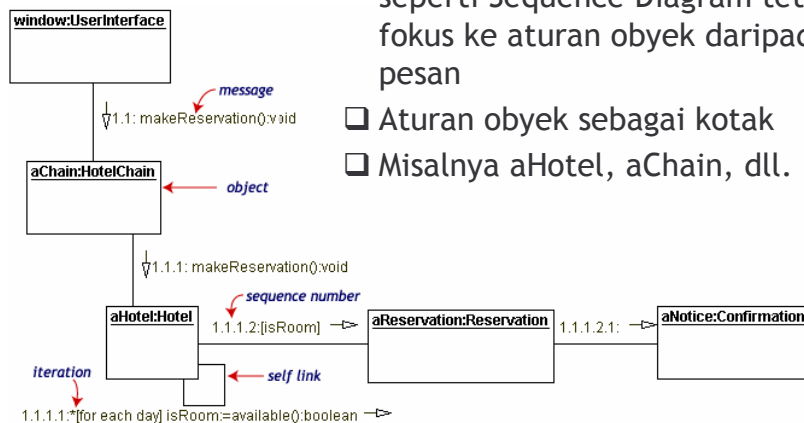
- ☐ Collaboration Diagram: dibentuk dari perspektif interaksi antar obyek
- ☐ Menangkap perilaku dinamis (berorientasi pesan)



Collaboration Diagram



Collaboration Diagram



- ❑ Menyampaikan info yg sama seperti Sequence Diagram tetapi fokus ke aturan obyek daripada pesan
- ❑ Aturan obyek sebagai kotak
- ❑ Misalnya aHotel, aChain, dll.

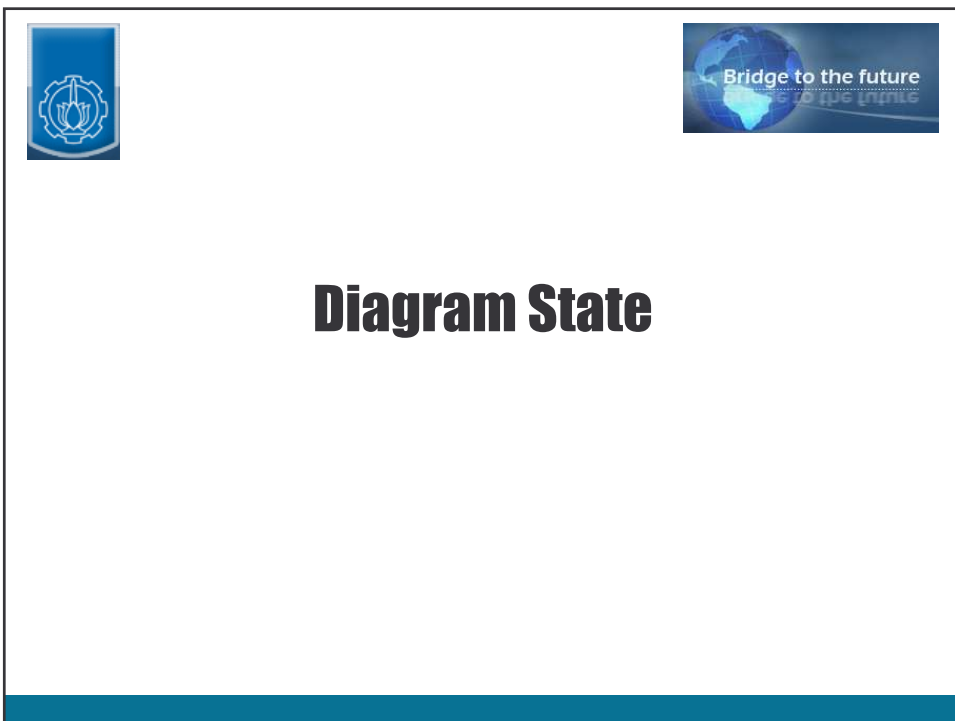
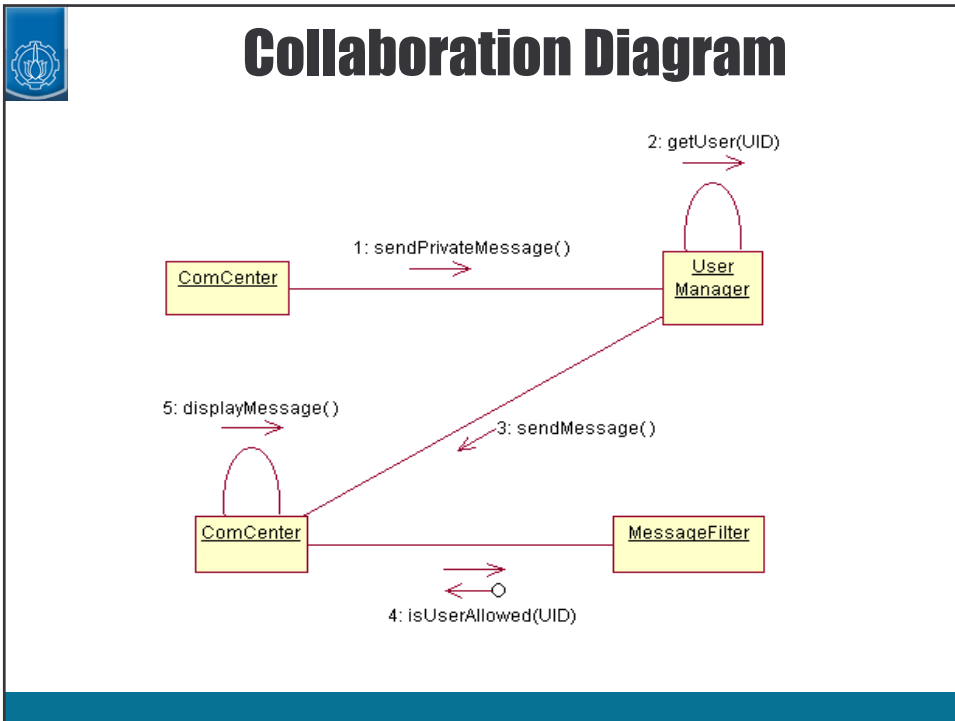


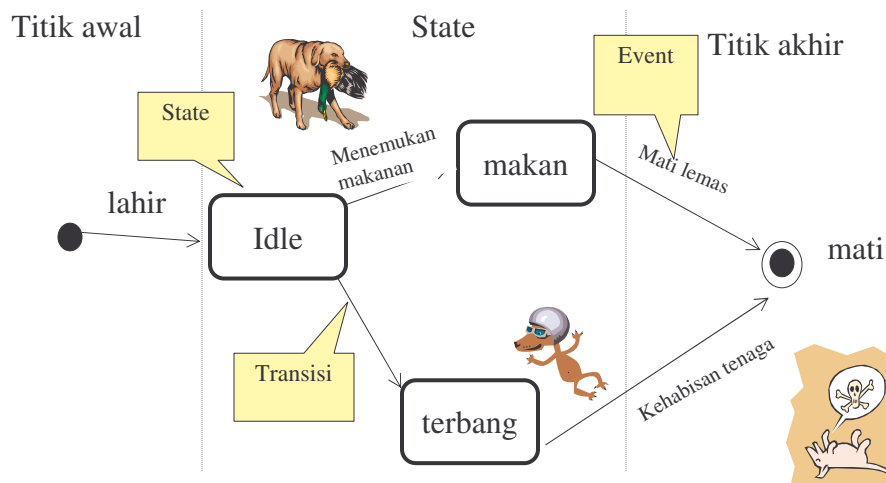


Diagram State

- ❑ Tipe finite state machine.
- ❑ Berguna untuk pemodelan suatu class yang perilakunya tergantung pada state tertentu
 - o Memodelkan state obyek dan bagaimana berpindah dari satu state ke state yang lain selama daur hidup
 - o Menggambarkan perilaku obyek melewati beberapa use case
- ❑ Suatu class mempunyai diagram state sendiri jika mempunyai perilaku dinamis yang menarik
 - o Pengontrol
 - o Perangkat (contoh: modem – dialing, sending, receiving)
 - o Obyek yang mengubah state atau tugas (contoh: buku dapat dipinjam, terlambat atau berada di perpustakaan)



Contoh



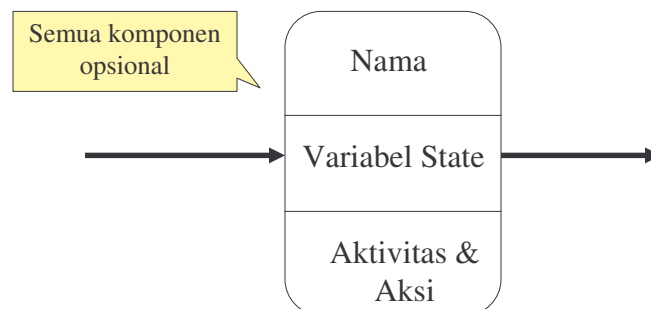


Event

- ❑ “Event adalah sesuatu yang terjadi yang menyebabkan beberapa aksi”
- ❑ Stimuli yang berpengaruh pada obyek
 - o Dasarnya komunikasi antar obyek
- ❑ Memindahkan obyek dari satu state ke state lain
- ❑ Berhubungan dengan
 - o Operasi
 - o Modifikasi nilai atribut
 - o Melintasi periode waktu
- ❑ Event asal tidak ditentukan



State





Variabel State

- ☐ Nilai gabungan semua atribut dan hubungan dilakukan oleh obyek yang merupakan identitas obyek
- ☐ Mengabaikan atribut yang tidak berpengaruh pada perilaku obyek
- ☐ Kesatuan nilai yang mempunyai respon yang sama ke event
- ☐ Atribut sementara (yang membangkitkan event) kemungkinan juga ditentukan



Aktifitas & Aksi

- ☐ **Aktifitas**
 - o Memakan waktu
 - o Dihubungkan hanya dengan state
 - o Mungkin di-interrupt oleh beberapa event
 - o Dapat dimodelkan sebagai diagram state berulang
 - o Contoh:
 - Download file
 - Mencetak
- ☐ **Aksi**
 - o Mempunyai durasi pendek
 - o Diasosiasikan dengan state dan transisi
 - o Atomik
 - o Contoh:
 - Terdengar bunyi beep
 - Menampilkan menu
 - Mengeset flag
 - Pembakaran event



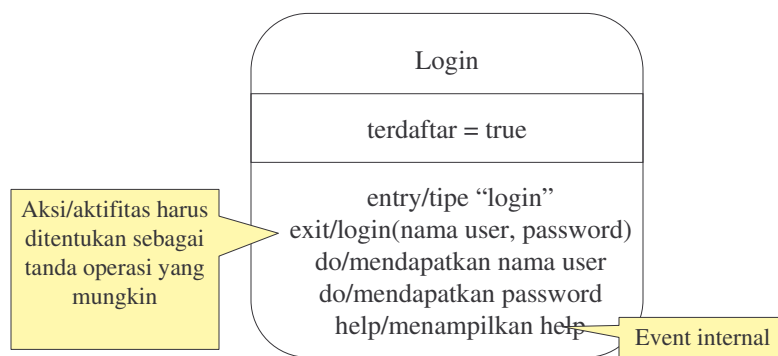
Aktifitas & Aksi

- ❑ Aksi dapat ditimbulkan pada *entry* dan *exit* ke state
 - o entry/ entry-action
 - o exit/ exit-action
- ❑ Aktivitas mungkin dinyatakan
 - o do/ activity
- ❑ Event internal mungkin dinyatakan (Internal events may be indicated (termasuk transisi berarah ke dirinya sendiri))
 - o internal-event/ action



Aktifitas & Aksi

❑ Contoh:





Transisi

- ☐ Transisi adalah hubungan antara state asal dan state target yang menyatakan bahwa obyek mempunyai state asal akan mengubah state ke state target, jika kumpulan event tertentu terjadi dan/atau kumpulan kondisi terpenuhi
- ☐ Transisi diri sendiri adalah transisi dimana state asal dan target sama
- ☐ Aksi terdapat pada transisi
- ☐ Transisi dari state yang sama harus *mutually exclusive*. Sebaliknya, state obyek tidak terdefinisi



Transisi

- ☐ Direpresentasikan dengan garis berarah antara dua state
- ☐ Sintak transisi (umum):
 - o event (atribut) [kondisi] / aksi ^ kirim-ketentuan
 - o Semua komponen opsional
- ☐ Semantik transisi:
 - o JIKA event terjadi DAN semua kondisi tertentu benar MAKA tempatkan aksi tertentu dan ubahlah state obyek ke state target
 - o Jika event pencetus tidak ditentukan, transisi akan dicoba jika semua state aktifitas lengkap

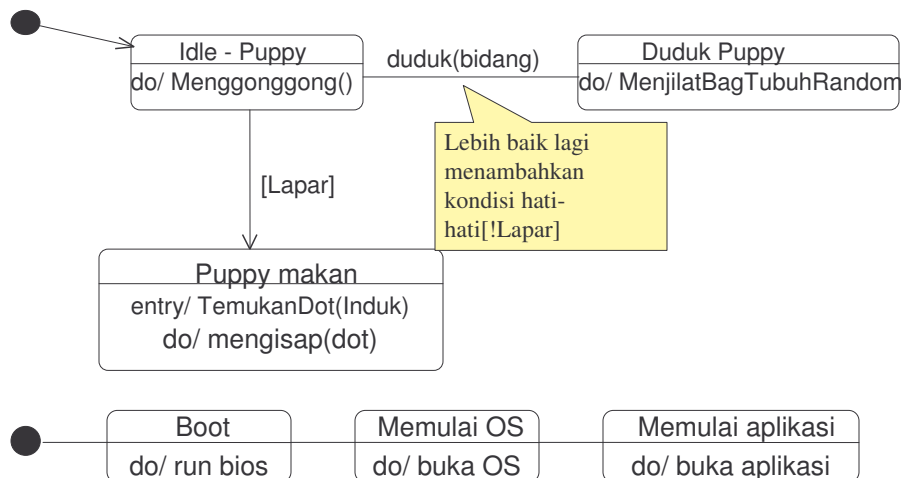


Transisi

- o Kirim-ketentuan digunakan untuk mengirimkan pesan (event) ke obyek lain (atau ke obyek yang sama)
 - Sintak: ekspresi-tujuan '.' nama-event-tujuan(argumen)
 - Contoh:



Contoh

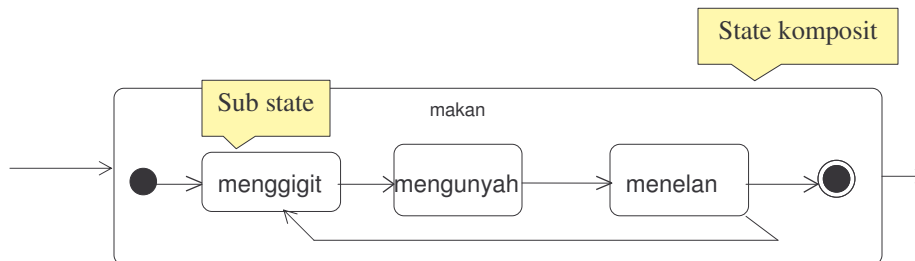




Sub-state



- ❑ Sub-state adalah state yang berisi state lain (state komposit)
- ❑ Sub-state dapat diulang ke semua level
- ❑ Contoh:



Transisi ke/dari state komposit

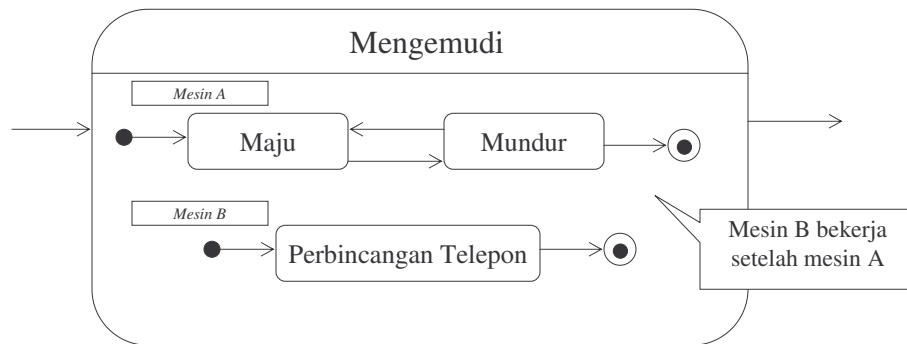
- ❑ Jika transisi adalah ke state komposit, diagram state berulang harus mempunyai state inisial
- ❑ Jika transisi ke adalah ke sub-state, aksi masukan sub-state dieksekusi setelah semua aksi masukan dari state komposit dieksekusi
- ❑ Jika transisi adalah dari sub-state ke aksi keluar sub-state dieksekusi dilanjutkan dengan aksi keluar dari state komposit
- ❑ Transisi dari state komposit bisa terjadi dari sembarang sub-state dan harus didahulukan daripada transisi dari sub-state saat itu



Sub-state



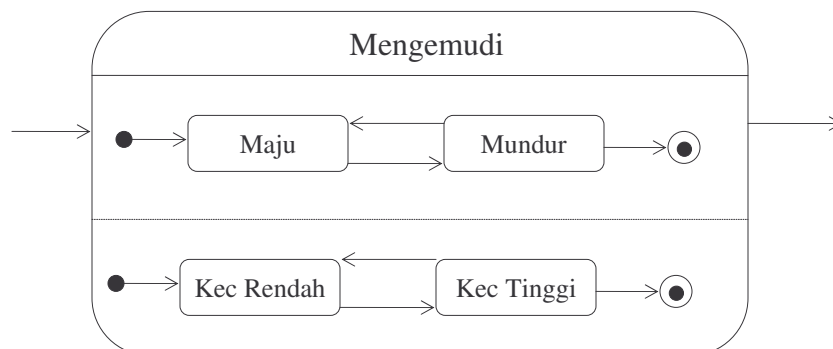
- ❑ 'atau' sub-state (biasanya disebut 'cluster')
 - o Sub-state dapat terjadi hanya satu kali

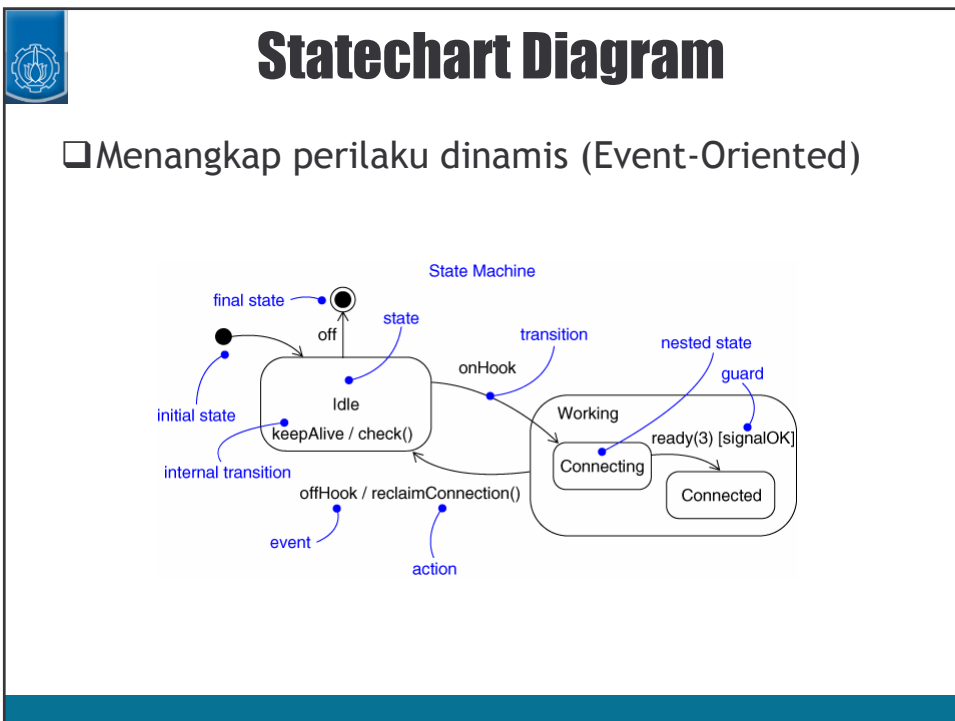
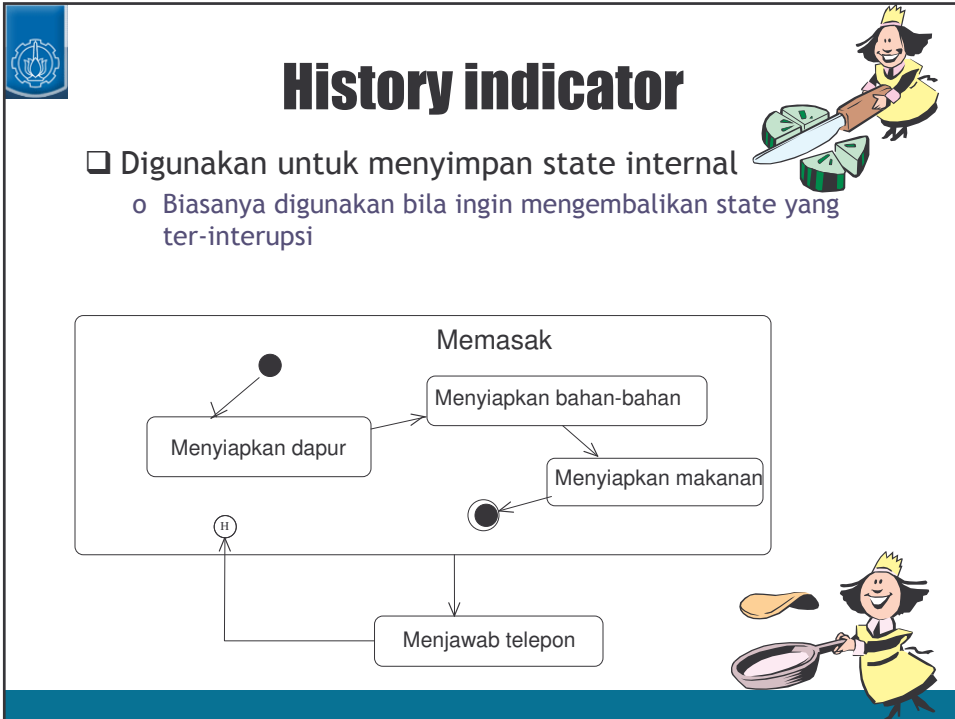


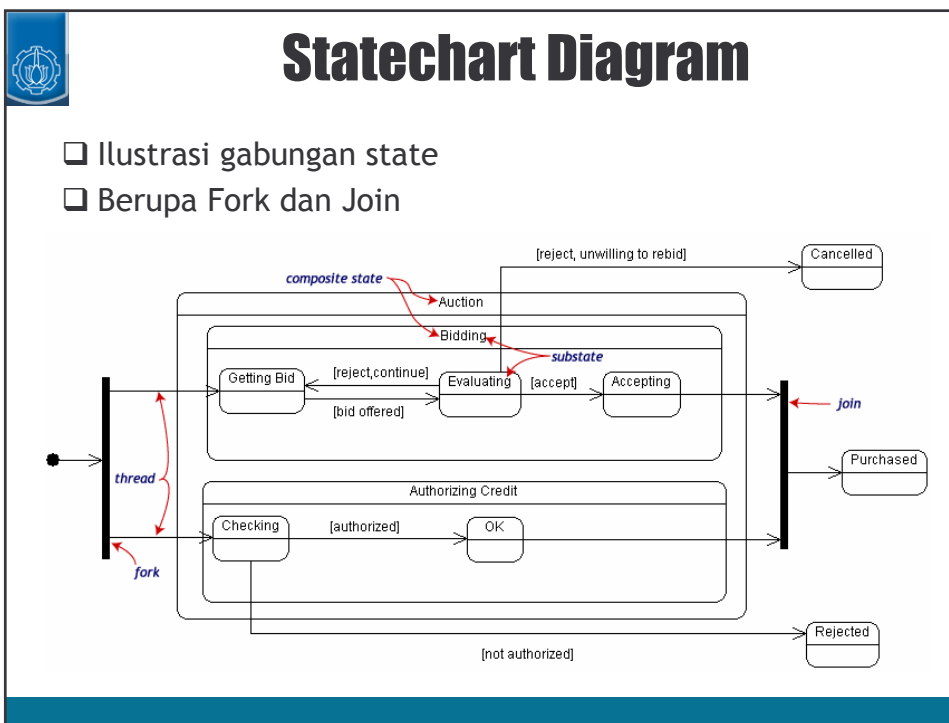
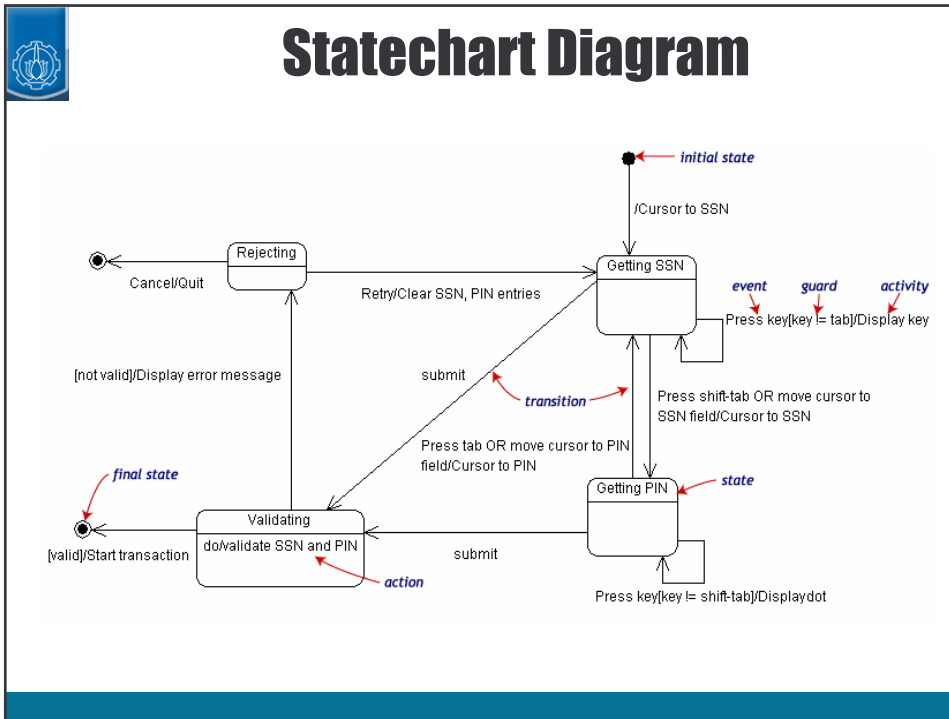
Sub-state



- ❑ 'dan' sub-state (biasanya disebut 'set')
 - o Beberapa su-state berada secara simultan







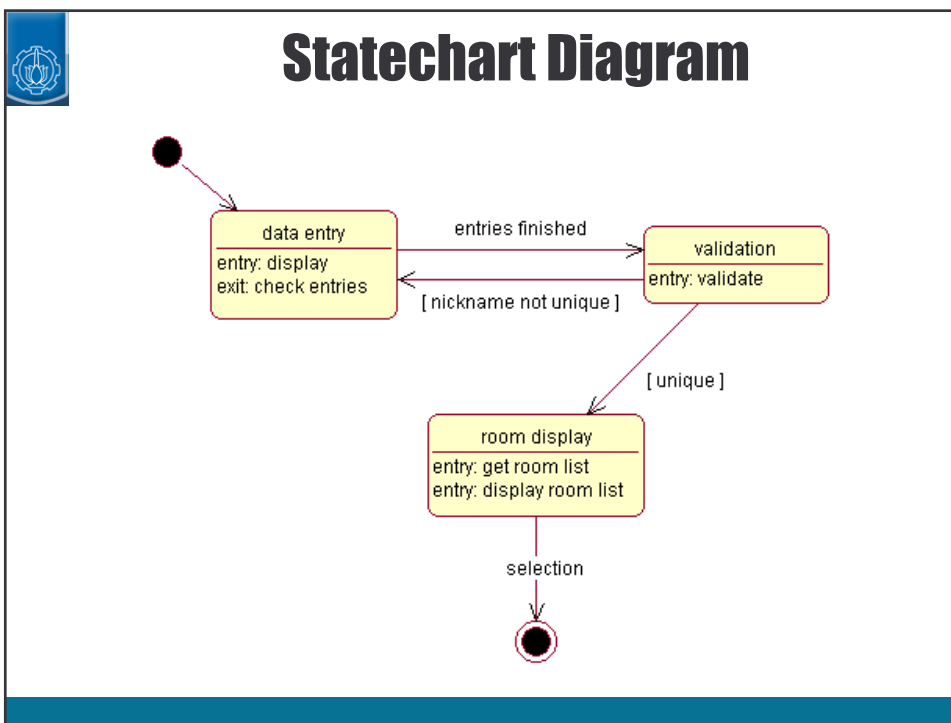
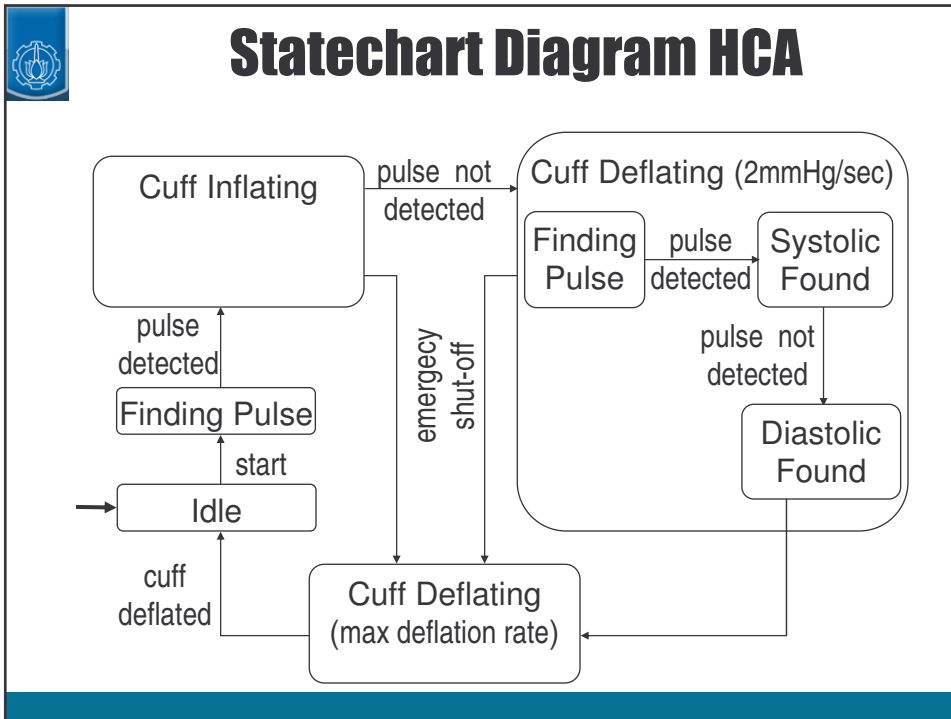




Diagram Aktivitas

- ❑ Aktivitas adalah eksekusi task yang berupa aktivitas fisik atau eksekusi kode.
- ❑ Diagram Aktivitas menggambarkan aliran perilaku pada system.
- ❑ Diagram aktivitas menunjukkan urutan aktivitas yang dilakukan.
- ❑ Seperti flow chart, diagram aktivitas mendukung perilaku kondisional, dengan tambahan dukungan untuk eksekusi parallel.
- ❑ Start : setiap activity diagram mempunyai satu start (letaknya diatas) pada saat memulai aksi
- ❑ End : setiap activity diagram mempunyai satu finish pada saat mengakhiri aksi.



- ❑ Activity : aktivitas yang dihubungkan oleh transisi. Transisi berupa panah berarah yang mengalir dari aktivitas sebelumnya ke aktivitas berikutnya. Biasanya diikuti label tekstual yang opsional dalam bentuk

[guard] label

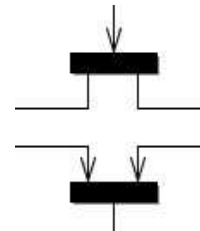
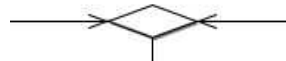
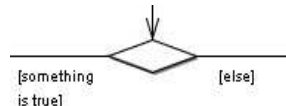
- ❑ Guard adalah ekspresi kondisional apabila transisi terjadi maka bernilai true. Label juga opsional dan bentuknya bebas.

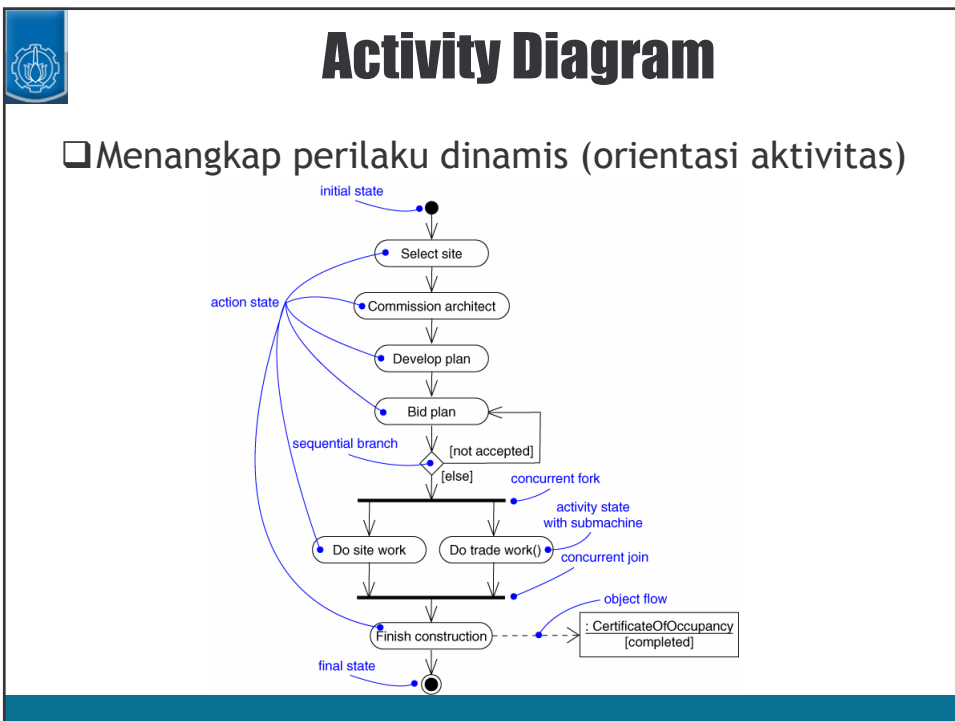
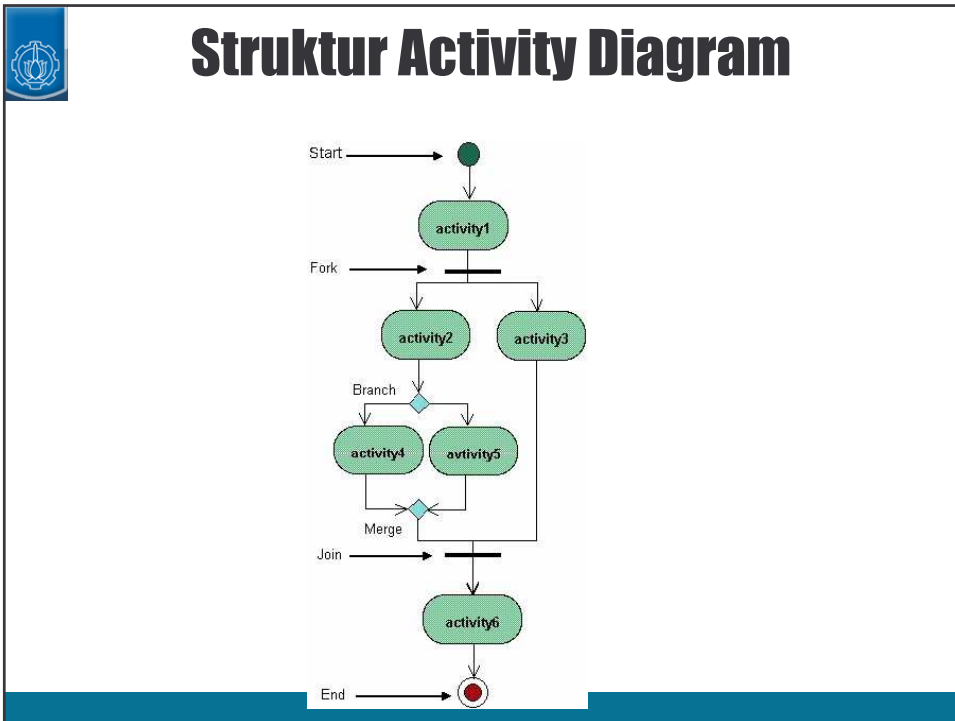
Activity

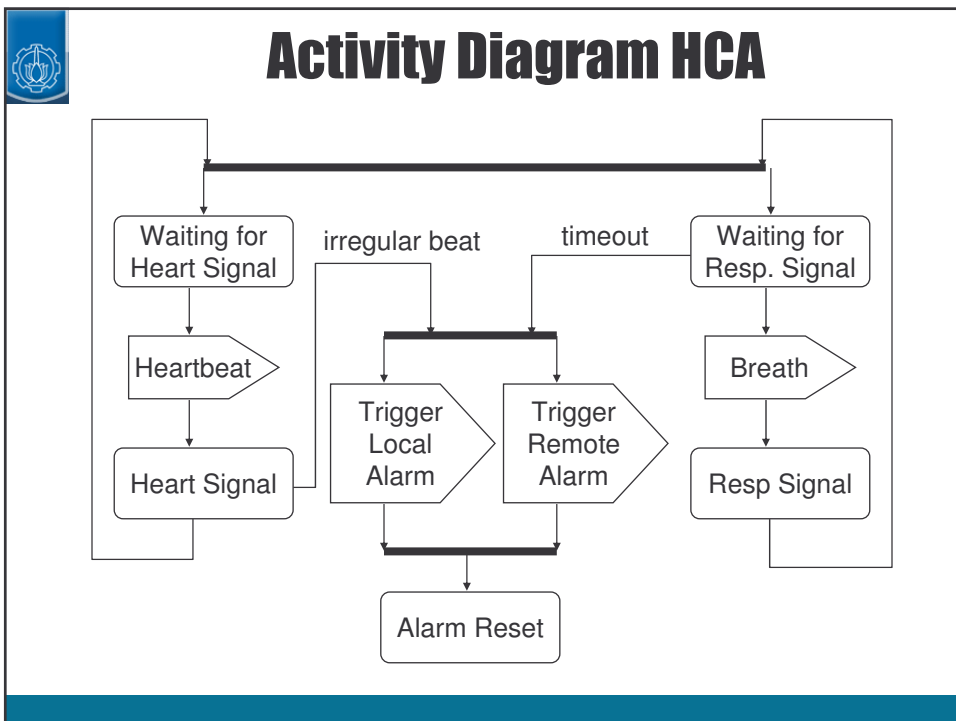
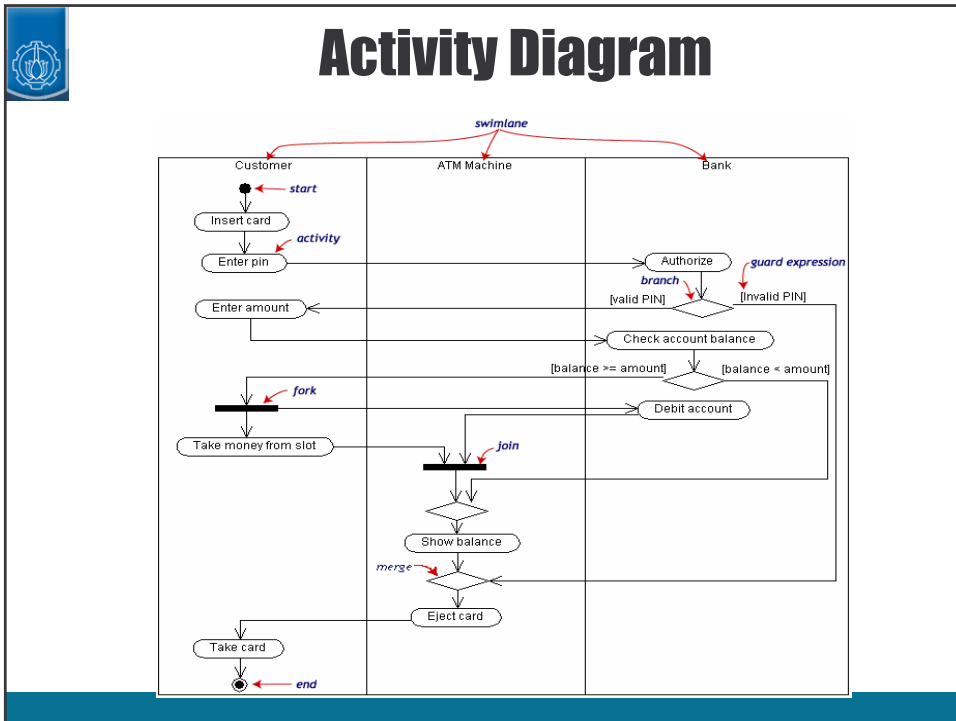


Diagram Aktivitas

- ❑ Untuk menunjukkan perilaku kondisional menggunakan cabang dan gabungan.
- ❑ Untuk menunjukkan perilaku parallel gunakan fork dan join.
- ❑ Fork mempunyai satu transisi yang masuk dan beberapa transisi yang keluar, semua akan dilakukan.
- ❑ Join merepresentasikan akhir dari perilaku parallel dan mempunyai beberapa transisi yang masuk dan hanya satu yang keluar.









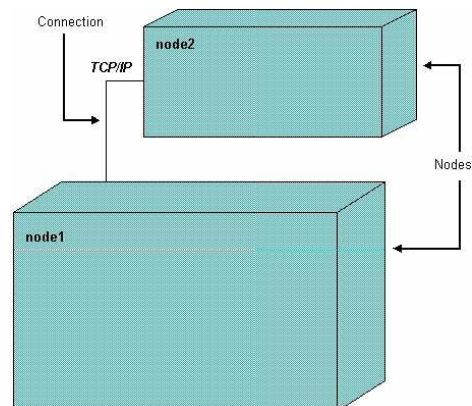
Implementation / Diagram Fisik

- ❑ Garis besar sistem dapat dilihat pada implementation diagram.
- ❑ Ada dua tipe implementation diagram yaitu deployment diagram dan component diagram.
- ❑ Deployment diagram menunjukkan relasi fisik antara hardware dan software dalam sistem.
- ❑ Component diagram menampilkan komponen software dari sistem dan bagaimana hubungannya dengan komponen software lain.
- ❑ Deployment dan component diagram dapat dikombinasikan dalam implementation diagram. Kombinasi ini akan menggabungkan fitur dari kedua diagram ini ke dalam satu diagram.



Deployment Diagram

- ❑ Deployment diagram terdiri dari titik dan koneksi. Titik biasanya merepresentasikan bagian hardware dari sistem. Koneksi menggambarkan bagian komunikasi yang digunakan oleh hardware untuk berkomunikasi dan biasanya menggunakan suatu metode misalnya TCP/IP.



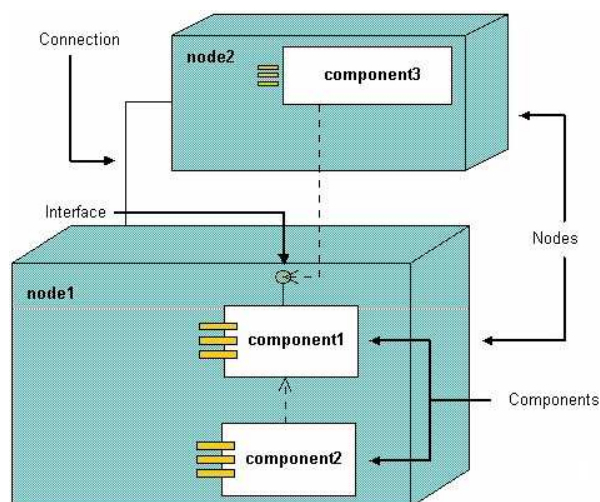


Component Diagram

- ❑ Component diagram berisi komponen dan dependensi.
- ❑ Component merepresentasikan paket fisik dari modul pengkodean.
- ❑ Dependensi antar komponen menunjukkan bagaimana perubahan dilakukan pada satu komponen berakibat pada komponen lain dalam system.
- ❑ Dependency pada component diagram direpresentasikan dengan garis putus-putus antara dua atau lebih komponen.
- ❑ Component diagram juga menunjukkan interface yang digunakan untuk berkomunikasi antar komponen.



Kombinasi Deployment dan Component diagram





Contoh Kombinasi Deployment dan Component diagram

