Perancangan Perangkat Lunak

UNIFIED MODELLING LANGUAGE

Pengantar

- 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.

Model Konseptual

- *Building block* (blok pembangun)
  sintaks (dan semantik dari sintaks) dari bagian model dengan UML
- *Rules*
  aturan untuk membangun model dari berbagai bagian model
- *Common mechanism*
  mekanisme pemodelan umum yang diterapkan di seluruh UML

Blok Pembangun pada UML

- Things

  abstraksi dari apa yang akan dimodelkan

- Relationship

  hubungan antar abstraksi (*things*)

- Diagrams

  mengelompokkan kumpulan sejumlah abstraksi yang dihubungkan

1. Things

- *Structural* (berpadanan dengan kata benda)
  merepresentasikan aspek statis sistem
- *Behavioural* (berpadanan dengan kata kerja)
  merepresentasikan aspek dinamis sistem
- *Grouping*
  menyatakan pengelompokkan sejumlah abstraksi dengan organisasi tertentu
- *Annotational*
  memberikan keterangan atas suatu abstraksi

Structural Things

Class

Window
origin size
open( ) close( ) move( )

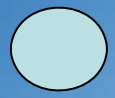
Collaboration

Chain of Responsibility

Active Class

Event Mgr
thread time
suspend( ) flush( ) stop( )

Node

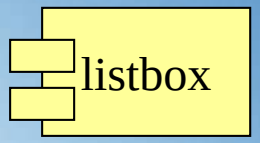


IWindow

Interface

Place Order

Use Case



Component

Structural Things

- **Class**

  deskripsi dari kumpulan objek dengan atribut, operasi, relasi, dan semantik yang sama
- **Interface**

  koleksi operasi yang menyatakan layanan dari kelas/komponen
- **Collaboration**

  mendefinisikan interaksi dan merupakan kumpulan peran dan elemen yang bekerja sama untuk menyediakan perilaku kooperatif agregat
- **Use case**

  deskripsi dari himpunan langkah aksi yang dilakukan sistem yang menghasilkan luaran kepada aktor tertentu

Structural Things

- **Active Class**

  Kelas yang mempunyai satu atau lebih proses / *thread* sehingga dapat memulai aktivitas kontrol
- **Component**

  Bagian fisik sistem yang dapat diganti-ganti yang sesuai dan menyediakan realisasi interface tertentu
- **Node**

  Elemen fisik yang ada saat *run time* dan mewakili sumber daya komputasi (kemampuan memori dan pemroses)

Behavioral Things

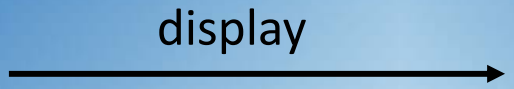
Bagian dinamik dari model UML

Biasanya terhubung dengan model struktural. Ditulis dalam kata kerja.

Ada 2 macam:

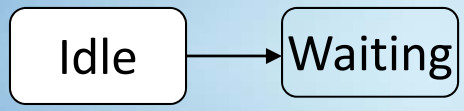
Interaksi

perilaku yang terdiri dari sekumpulan pesan yang saling dipertukarkan antar sekumpulan objek dalam konteks tertentu untuk mencapai tujuan tertentu



State Machine

perilaku yang menspesifikasikan urutan *state* dari objek atau interaksi yang terjadi selama hidup objek tersebut dalam menyikapi *event* dan tanggapannya terhadap event-event tersebut

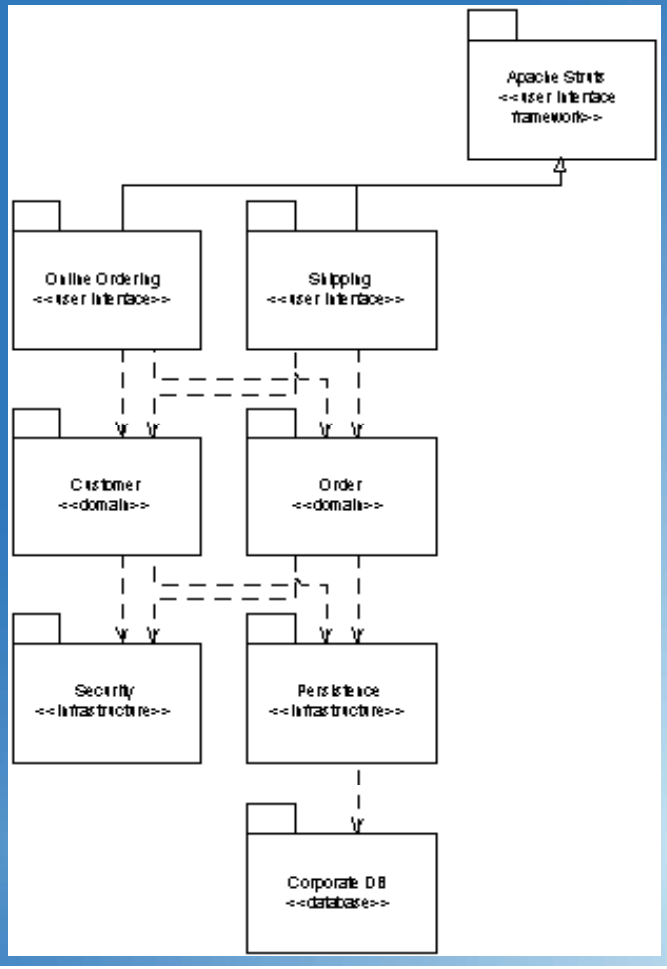


Package Diagram

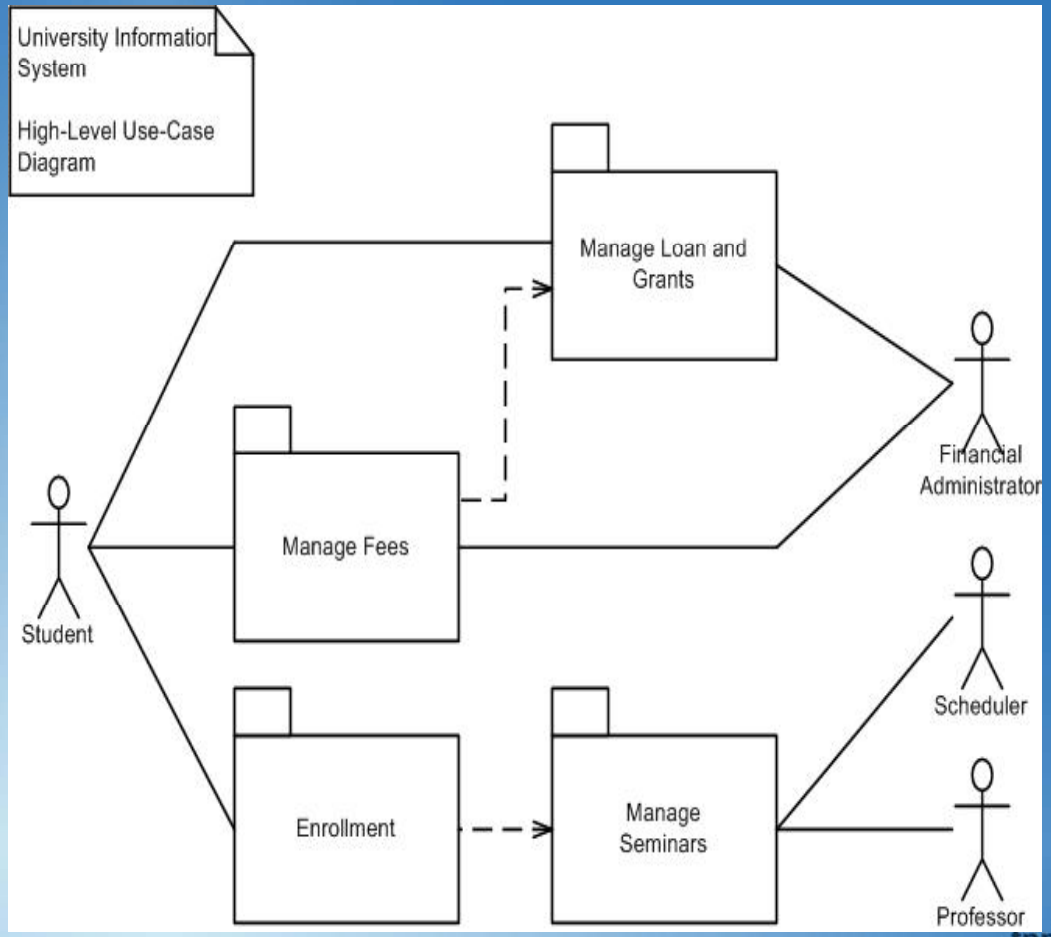
- Biasanya dipakai pada ***use case diagram*** atau ***class diagram***
- Packages digambarkan sebagai sebuah direktori (file folders) yang berisi model-model elemen
- Walaupun package secara resmi bukanlah diagram UML, namun kegunaannya cukup signifikan
- Packages dibuat untuk :
  - Menggambarkan high level overview kebutuhan sistem
  - Menggambarkan high level overview design
  - Memecah sebuah diagram yang mempunyai banyak bubbles
  - Mengorganisasikan source code programming
- Digambarkan dengan lambang

Package Diagram

Class diagram Package



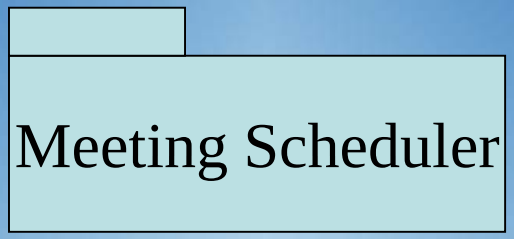
Use case diagram Package



Grouping & Annotational Things

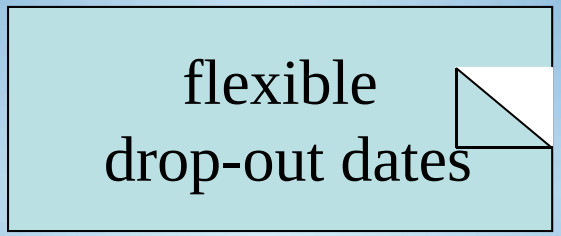
Packages

- Mekanisme untuk mengorganisasi elemen
- Konseptual, hanya ada pada waktu pengembangan
- Berisi structural dan behavioral things
- Dapat bersarang
- Variasi package: framework, model, & subsystem.



Notes

Elemen UML (Note) yang digunakan untuk menerangkan elemen lain pada model



2. Relationships

4 jenis

- Dependensi
- Asosiasi
- Generalisasi
- Realisasi

Relationships

Dependensi

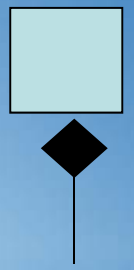
merupakan hubungan semantik antara 2 things sedemikian sehingga perubahan pada satu thing mengakibatkan perubahan pada thing lainnya

Asosiasi

merupakan hubungan struktural yang menggambarkan himpunan link antar objek

Aggregasi

jenis khusus dari asosiasi (menyatakan *whole part*)



Relationships

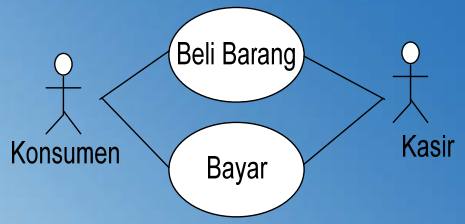
Association - Use Case Diagram

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

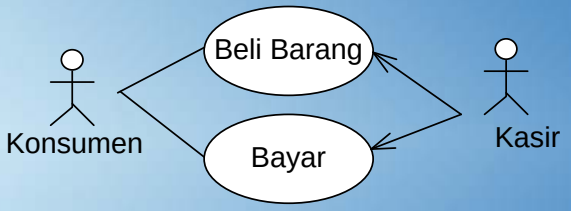
Relationships

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

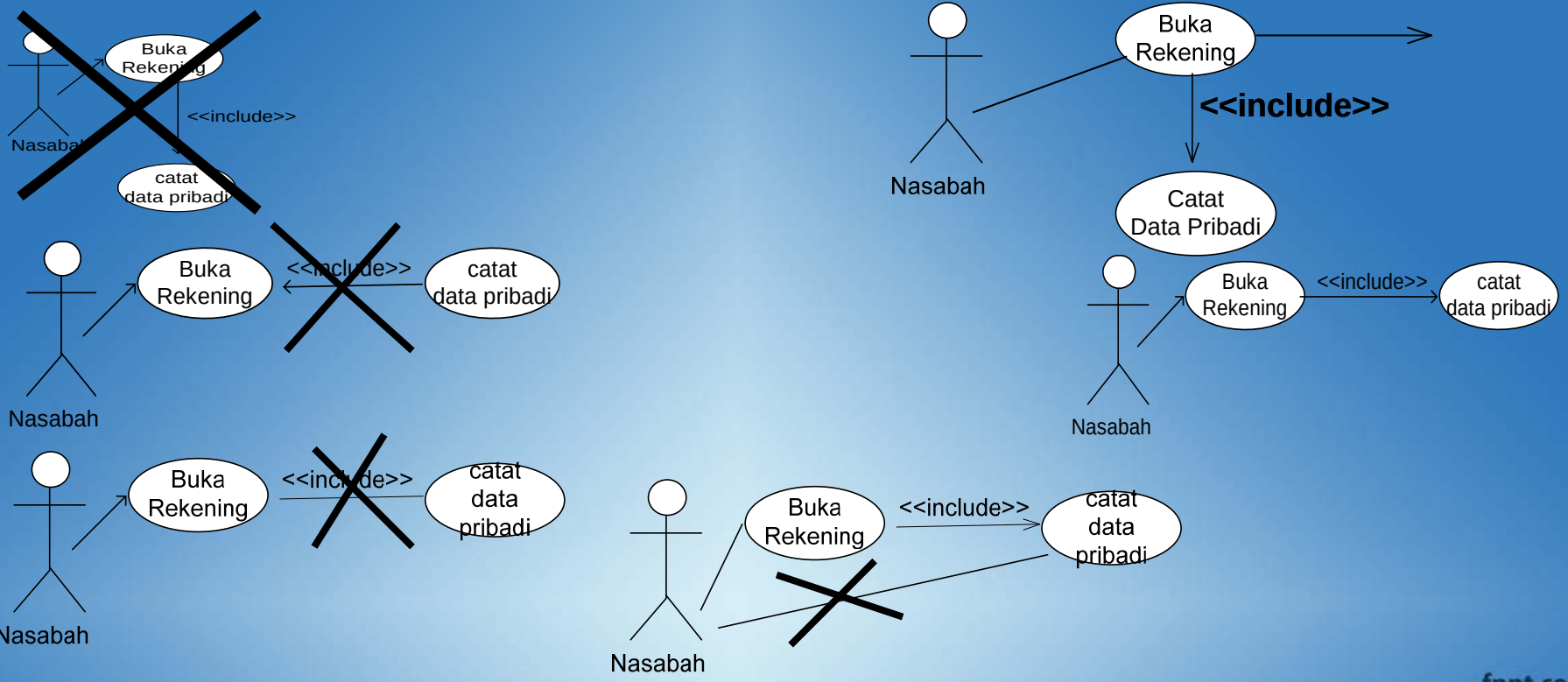


- association antara actor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara ***pasif*** dengan system anda.



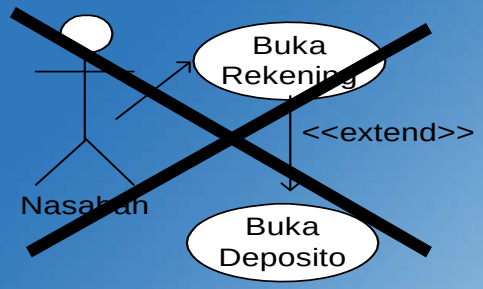
Association antara use case

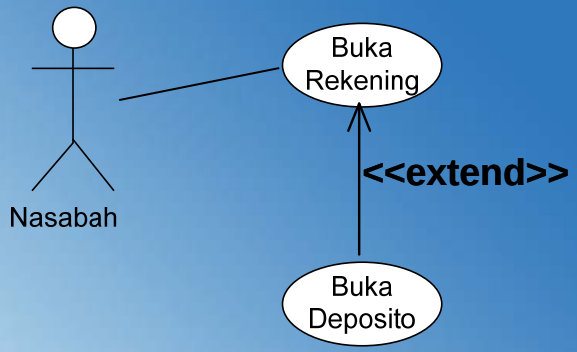
- <<include>>
  - termasuk didalam use case lain (required) / (diharuskan)
  - Pemanggilan use case oleh use case lain
  - contohnya adalah Pemanggilan sebuah fungsi program
  - *Gambarkan association <<include>> secara horizontal*
  - Tanda panah terbuka harus terarah ke sub use case
  - Tidak boleh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

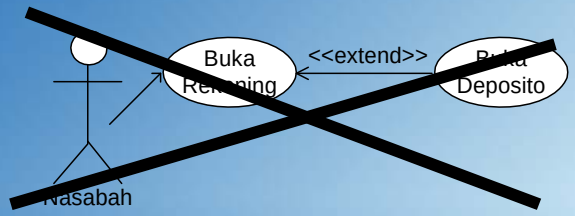


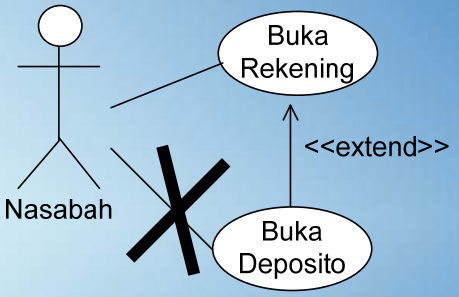
Association antara use case

- <<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 (picture extending use case below than base/parent use case)*
  - Tidak boleh actor dihubungkan pada use case <<extend>>









Relationship

Generalisasi

Relasi antar objek yang memiliki hubungan general-spesial

Realisasi

Relasi semantik antara 2 elemen dimana 1 elemen melaksanakan apa yang diharapkan dari elemen lain. Biasanya antara antarmuka dan kelas yang merealisasikannya, atau pada use case dan kolaborasi yang merealisasikannya

Relationships

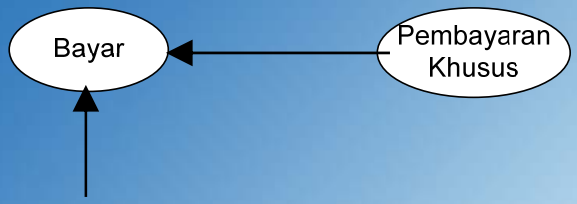
Generalization/inheritance

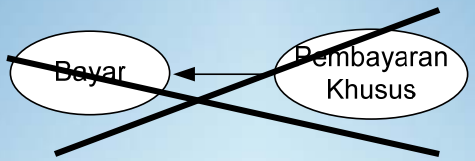
- Generalization/inheritance digambarkan dengan sebuah garis berpanah tertutup pada salah satu ujungnya yang menunjukkan lebih umum

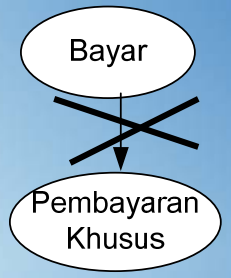
- Harus digambarkan secara vertikal

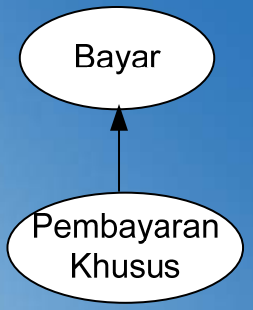
Generalization/inheritance antara use case

- Dibuat ketika ada sebuah keadaan yang lain/perlakuan khusus
- Inheriting use case dibawah base/parent use case





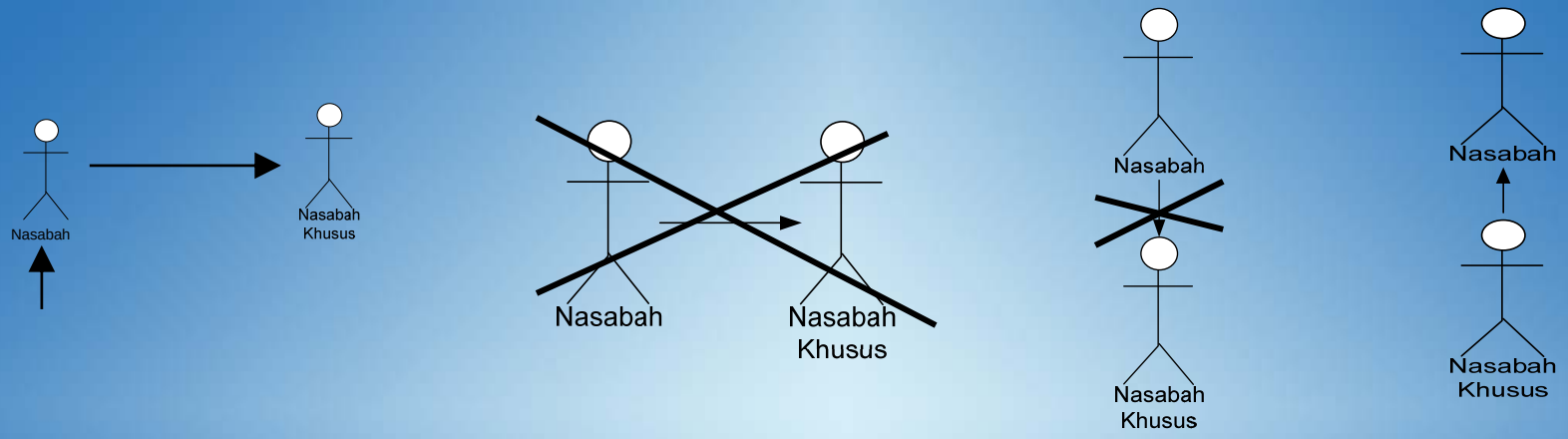




Relationships

Generalization/inheritance antara actor

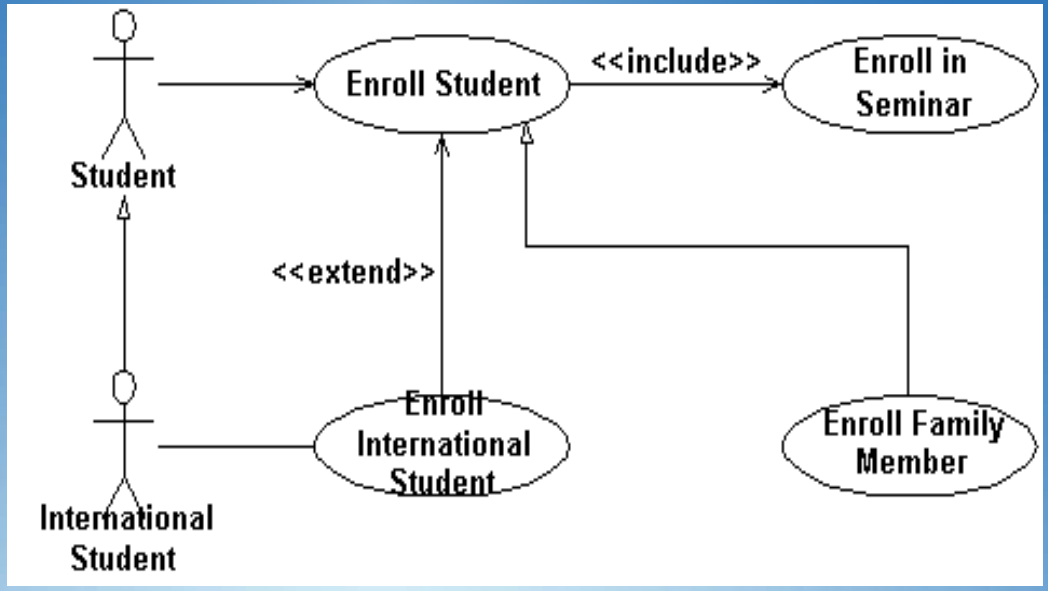
- Dibuat ketika ada sebuah actor baru terbentuk dan mempunyai atribut dan methode yang sama dengan actor yang sudah ada
- Inheriting actor dibawah base/parent actor



Relationships

Generalization/inheritance

Contoh :



3. Diagrams

- Representasi grafik dari sekumpulan elemen.
- Direpresentasikan dalam sebuah graf dimana node adalah thing dan busur adalah behavior
- Ada 9 diagram:

Class Diagram; Object Diagram; Use case Diagram

Sequence Diagram; Collaboration Diagram; Statechart Diagram

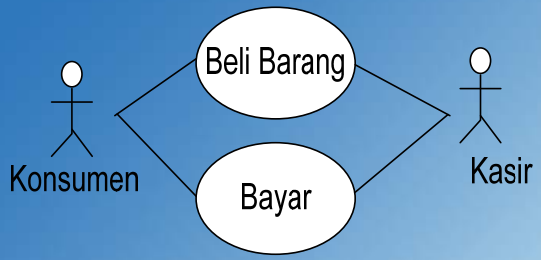
Activity Diagram; Component Diagram; Deployment Diagram

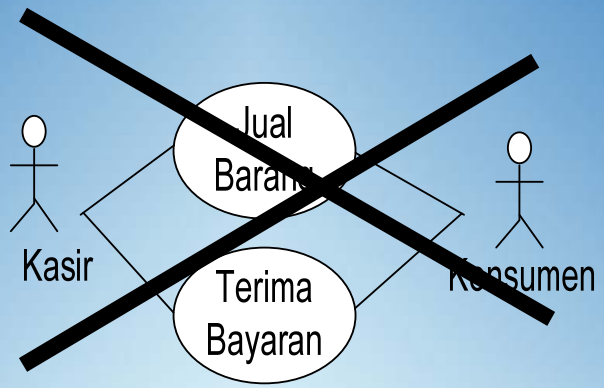
Use Case Diagram

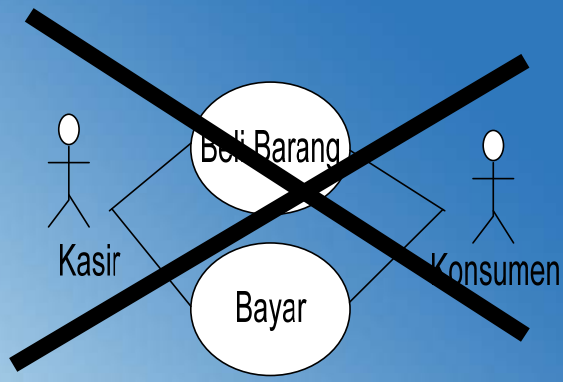
- Use case diagram menggambarkan fungsionalitas yang diharapkan dari sebuah sistem. Yang ditekankan adalah “apa” yang diperbuat sistem, dan bukan “bagaimana”. Sebuah use case merepresentasikan sebuah interaksi antara aktor dengan sistem.

Use Case Diagram

- Menggambarkan kebutuhan sistem dari sudut pandang user
- Mengfokuskan pada proses komputerisasi (automated processes)
- Menggambarkan hubungan antara use case dan actor
- Use case menggambarkan proses sistem
  (kebutuhan sistem dari sudut pandang user)







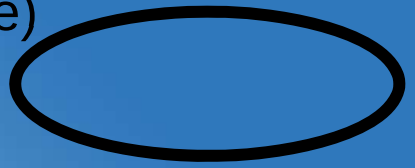
Use Case Diagram

- Secara umum use case adalah:
  - Pola perilaku sistem
  - Urutan transaksi yang berhubungan yang dilakukan oleh satu actor
- Use case diagram terdiri dari
  - Use case
  - Actors
  - Relationship
  - System boundary boxes (optional)
  - Packages (optional)
- Every actor is involved with at least one use case, and every use case is involved with at least one actor

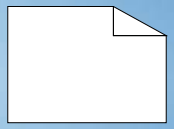
Use Case Diagram

Use Case : Use Case Diagram

- 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 verb
- Nama use case boleh terdiri dari beberapa kata dan tidak boleh ada 2 use case yang memiliki nama yang sama
- Sebuah use case bisa mempunyai dokumentasi
- Gunakan dengan lambang dibawah ini dan ditarik dengan garis putus tanpa panah



Use Case Diagram

- Letakkan use case utama anda pada pojok kiri atas dari diagram (in western culture people read from left to right, top to bottom, starting in the top-left corner)
- Use case diagram tidak terpengaruh urutan waktu, meskipun demikian supaya mudah dibaca perlu penyusunan use case

  (increase the readability of your use case diagram by arranging use cases to imply timing, one such way is to stack them. So that the use case that typically occur first are shown above those that appear later)

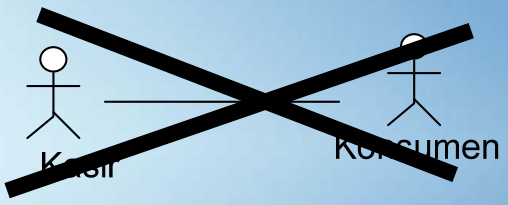
Use Case Diagram

Actor - Use Case Diagram

- Actor menggambarkan orang, sistem atau external entitas / stakeholder yang menyediakan atau menerima informasi dari system
- Actor memberi input atau menerima informasi dari system
- Actor biasanya menggunakan Noun
- Actor digambarkan dengan gambar stick figure atau dengan gambar visual

 atau  atau  dll

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



Use Case Diagram

- Indikasi <<system>> untuk sebuah actor yang merupakan sebuah system

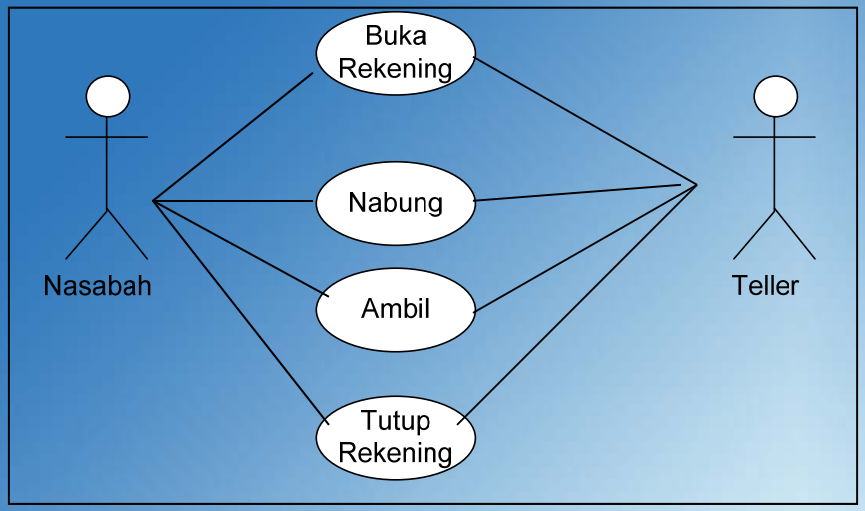
<<System Keuangan>>

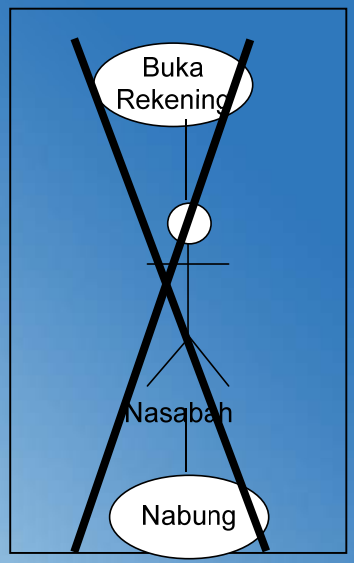
- Adanya actor bernama “Time” yang mengindikasikan scheduled events (suatu kejadian yang terjadi secara periodik/bulanan)

Time

Use Case Diagram

- Letakkan actor utama anda pada pojok kiri atas dari diagram (in western culture people read from left to right, top to bottom)
- (place your primary Use case in the top left corner of the diagram because your primary actor is often directly involved with your primary/critical use case)
- Actor jangan digambarkan ditengah-tengah use cases





- Actors menggambarkan sebuah tugas/peran dan bukannya posisi sebuah jabatan (Actors model roles, not positions. A good indication that you are modeling positions instead of roles is a use case diagram depicting several actors with similar names that have associations to the same use case)

Contoh: Sistem Registrasi Universitas

The UTD wants to computerize its registration system

- The Registrar sets up the curriculum for a semester
  - One course may have multiple course offerings
- Students select four (4) primary courses and two (2) alternate courses
- Once a student registers for a semester, the billing system is notified so the student may be billed for the semester
- Students may use the system to add/drop courses for a period of time after registration
- Professors use the system to set their preferred course offerings and receive their course offering rosters after students register
- Users of the registration system are assigned passwords which are used at logon validation

Contoh: Sistem Registrasi Universitas

- Aktor adalah seseorang atau sesuatu yang berinteraksi dengan sistem yang dikembangkan

Registrar

Professor

Student

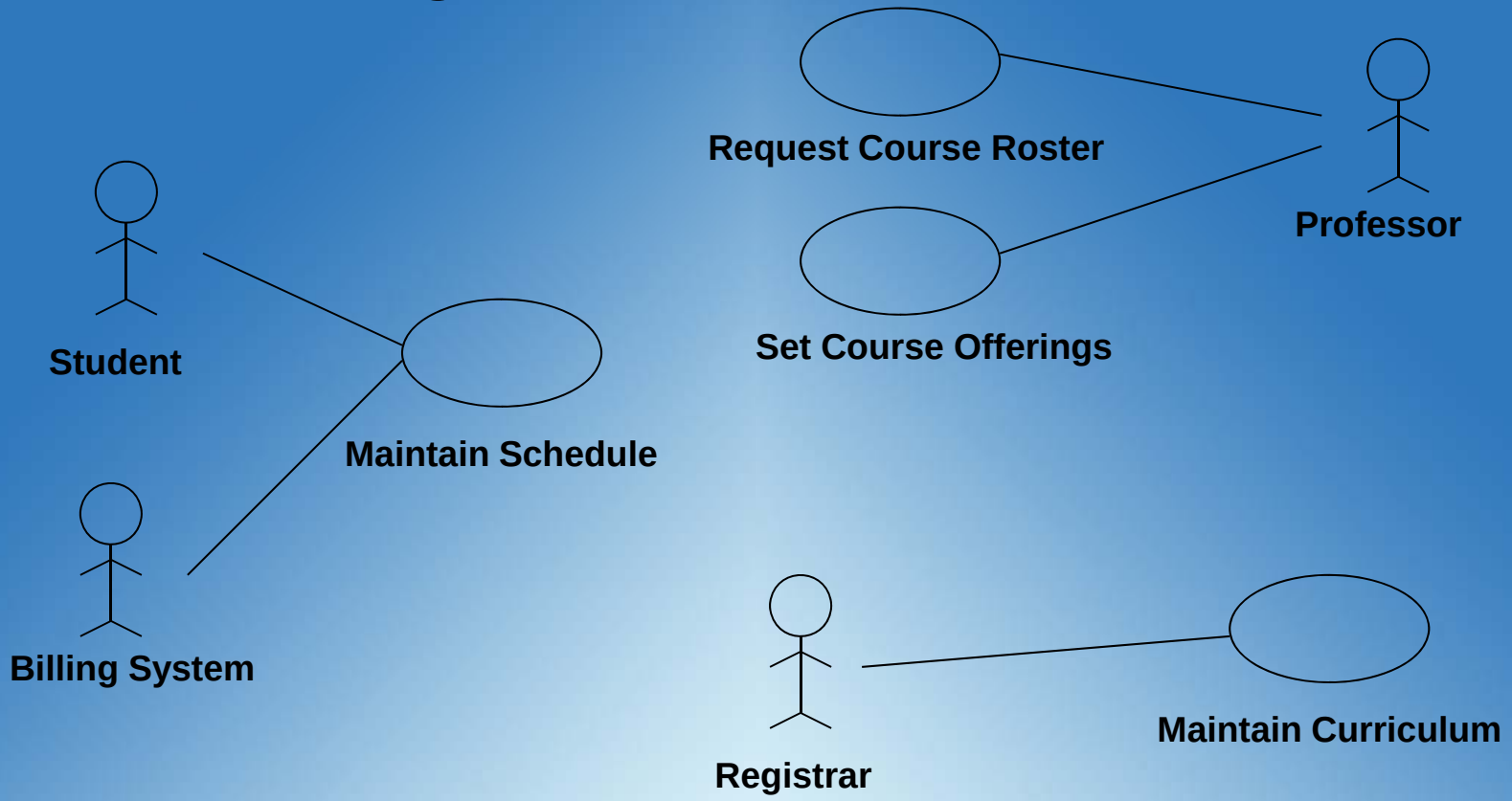
Billing System

Contoh: Sistem Registrasi Universitas

- Sebuah use case menunjukkan perilaku sistem
- Aktor:
  - Registrar -- mengelola kurikulum
  - Professor – menentukan MK yang akan ditawarkan dan meminta daftar MK
  - Student – mengelola jadwal
  - Billing System – menerima informasi tagihan

Contoh: Sistem Registrasi Universitas

- Use case diagrams



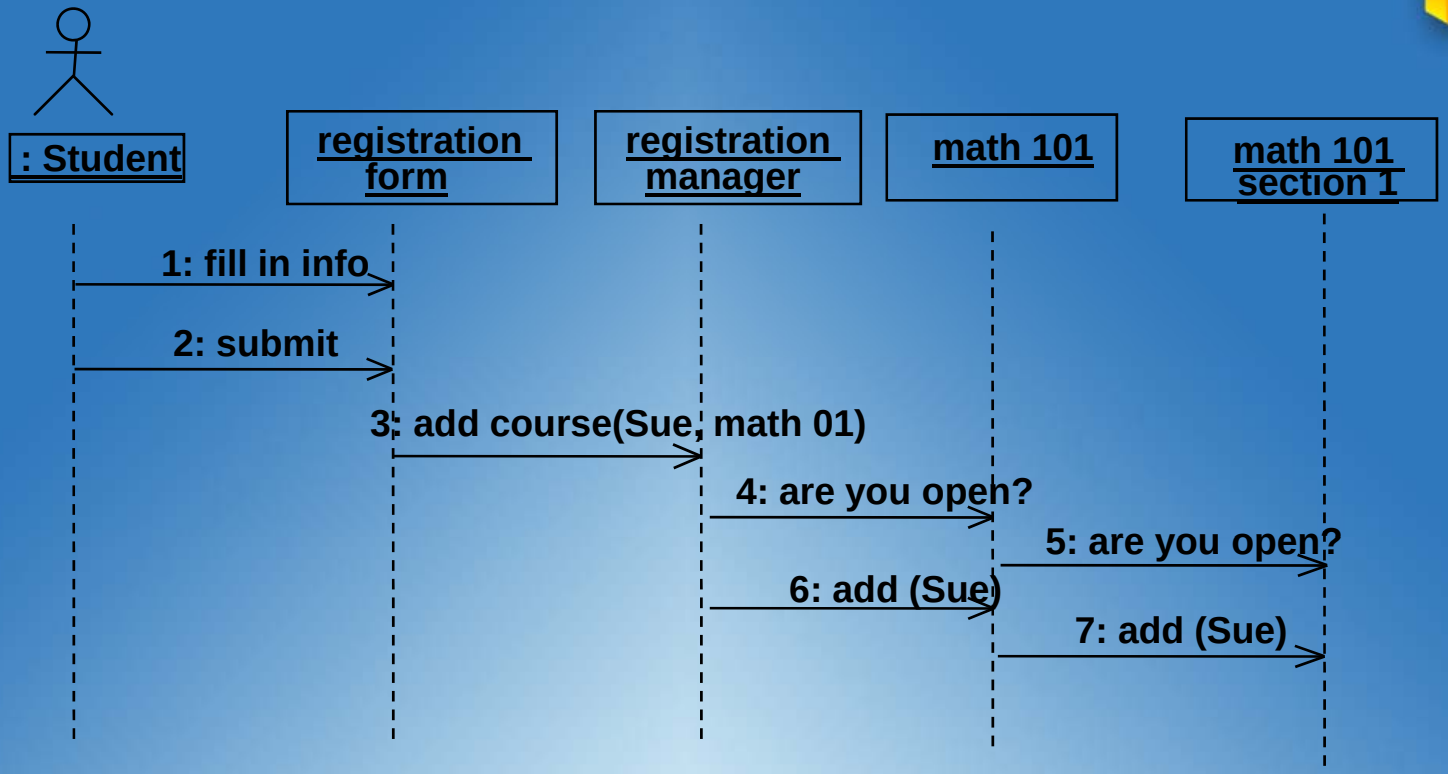
Realisasi Use Case

- Diagram use case menggambarkan outside view dari sistem
- Inside view dari sistem digambarkan dengan diagram interaksi
- Diagram interaksi menggambarkan bagaimana use case direalisasikan sebagai interaksi antar sekumpulan objek dengan mempertukarkan message. Diagram interaksi menggambarkan pandangan dinamis dari sistem
- Ada 2 jenis:
  - Sequence Diagram
  - Collaboration Diagram

Sequence Diagram

- Sequence diagram menggambarkan interaksi antar objek di dalam & di sekitar sistem (termasuk pengguna, display, dan sebagainya) berupa message yg digambarkan terhadap waktu. Sequence diagram terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).
- Sequence diagram biasa digunakan untuk menggambarkan 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

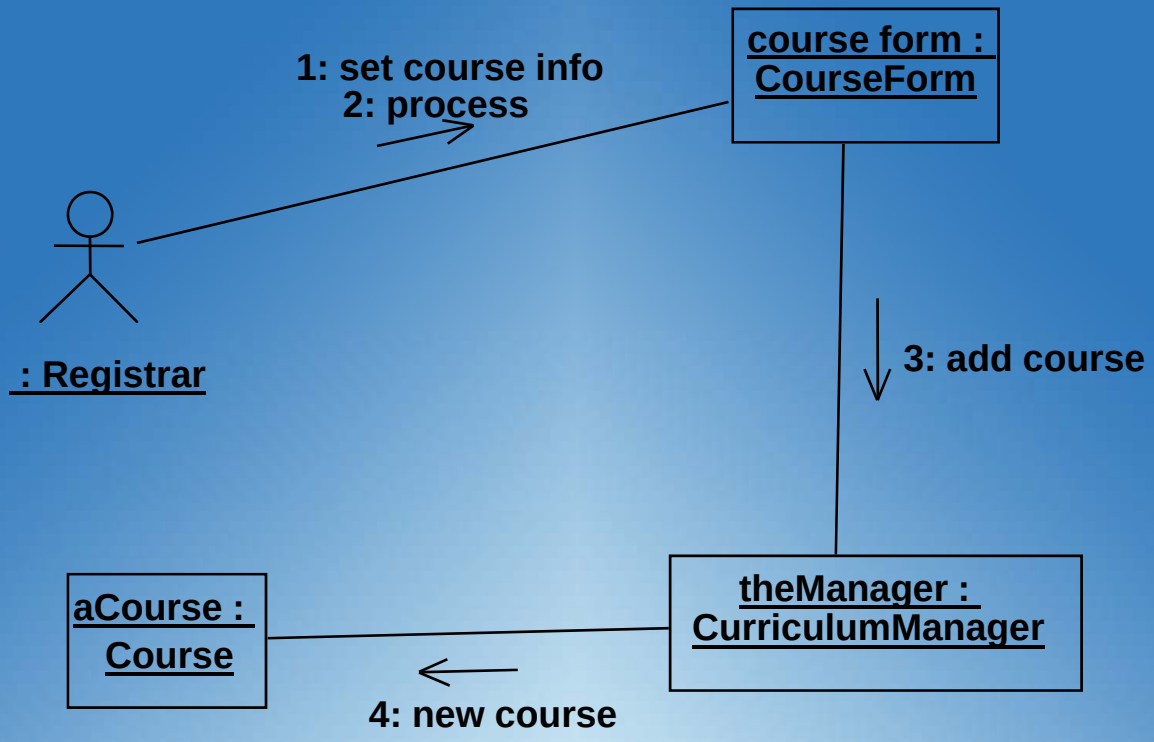
Sequence Diagram



Collaboration Diagram

- Collaboration diagram juga menggambarkan interaksi antar objek seperti sequence diagram, tetapi lebih menekankan pada peran masing-masing objek dan bukan pada waktu penyampaian message.
  Setiap message memiliki sequence number, di mana message dari level tertinggi memiliki nomor 1. Messages dari level yang sama memiliki prefiks yang sama.

Collaboration Diagram



Class Diagram

- Sebuah class diagram menunjukkan keberadaan class-class dan relasinya
- Elemen-elemen class diagram : class, relasi, multiplicity, dan *role names*
- Sebuah class adalah abstraksi dari sekumpulan objek yang memiliki struktur yang sama, perilaku yang sama, relasi yang sama dan semantik yang sama
- Beberapa class muncul pada sequence diagram atau collaboration diagram
- Harus dibuat standar penamaan class, misalnya: class ditulis dalam kata benda dengan huruf kapital di awal kata

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

Class Diagram

RegistrationForm

RegistrationManager

ScheduleAlgorithm

Course

Student

Professor

CourseOffering

Class Diagram

RegistrationForm

ScheduleAlgorithm

RegistrationManager

addStudent(Course, StudentInfo)

Course

name
numberCredits

open()
addStudent(StudentInfo)

Student

name
major

Professor

name
tenureStatus

CourseOffering

location

open()
addStudent(StudentInfo)

Diagram Objek

Menunjukkan sekumpulan objek dan relasinya

Harry (Student)
Name: “Harry Mat” **Major: CS**

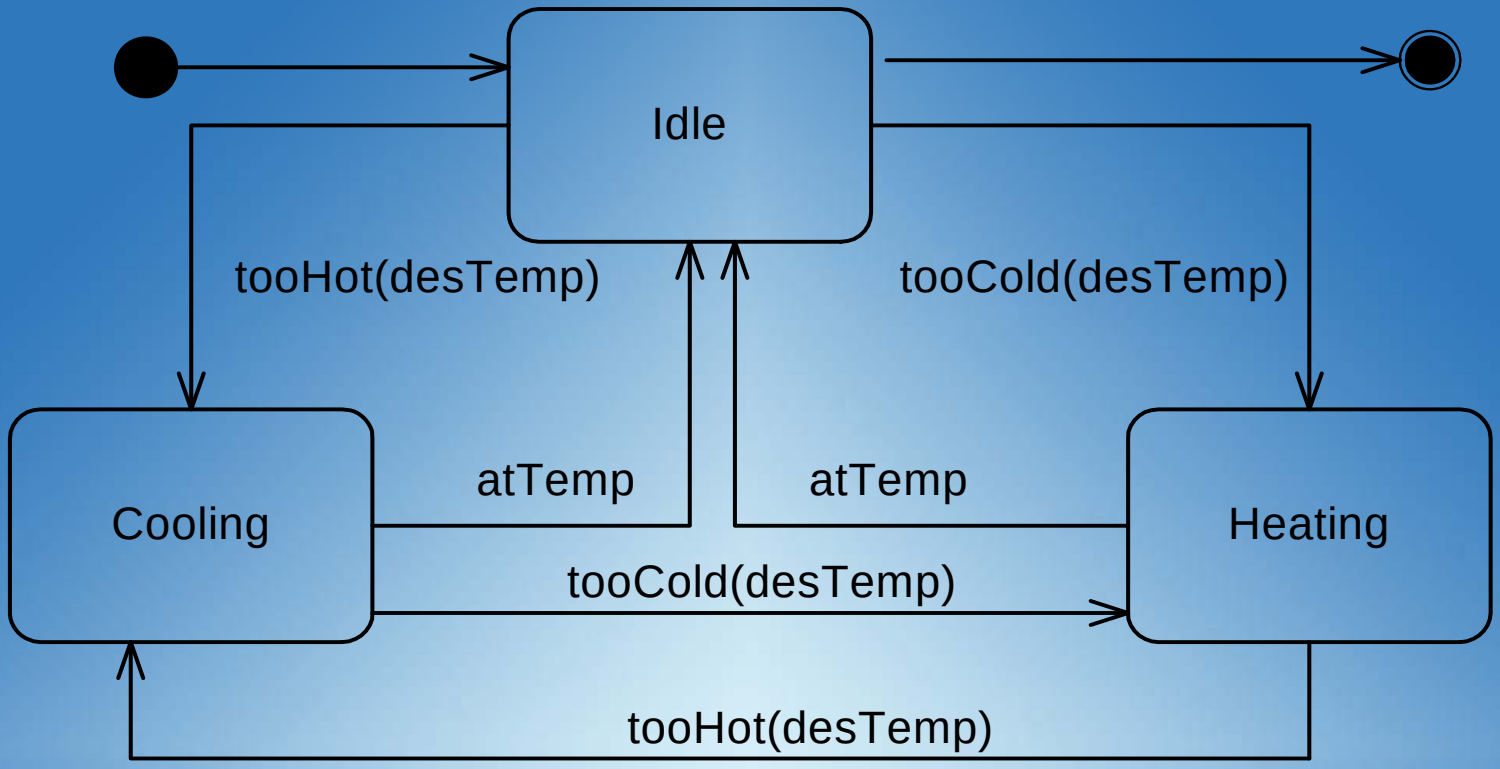
Sue (Professor)
Name: “Sue Becker” **tenureStatus: true**

Statechart Diagram

- Statechart diagram menggambarkan transisi dan perubahan keadaan (dari satu state ke state lainnya) suatu objek pada sistem sebagai akibat dari stimuli yang diterima. Pada umumnya statechart diagram menggambarkan class tertentu (satu class dapat memiliki lebih dari satu statechart diagram).

Statechart Diagram

- Terdiri dari status, transisi, kejadian, dan aktivitas



Activity Diagram

- Activity diagrams menggambarkan berbagai alir anaktivitas 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.

Diagram Aktivitas

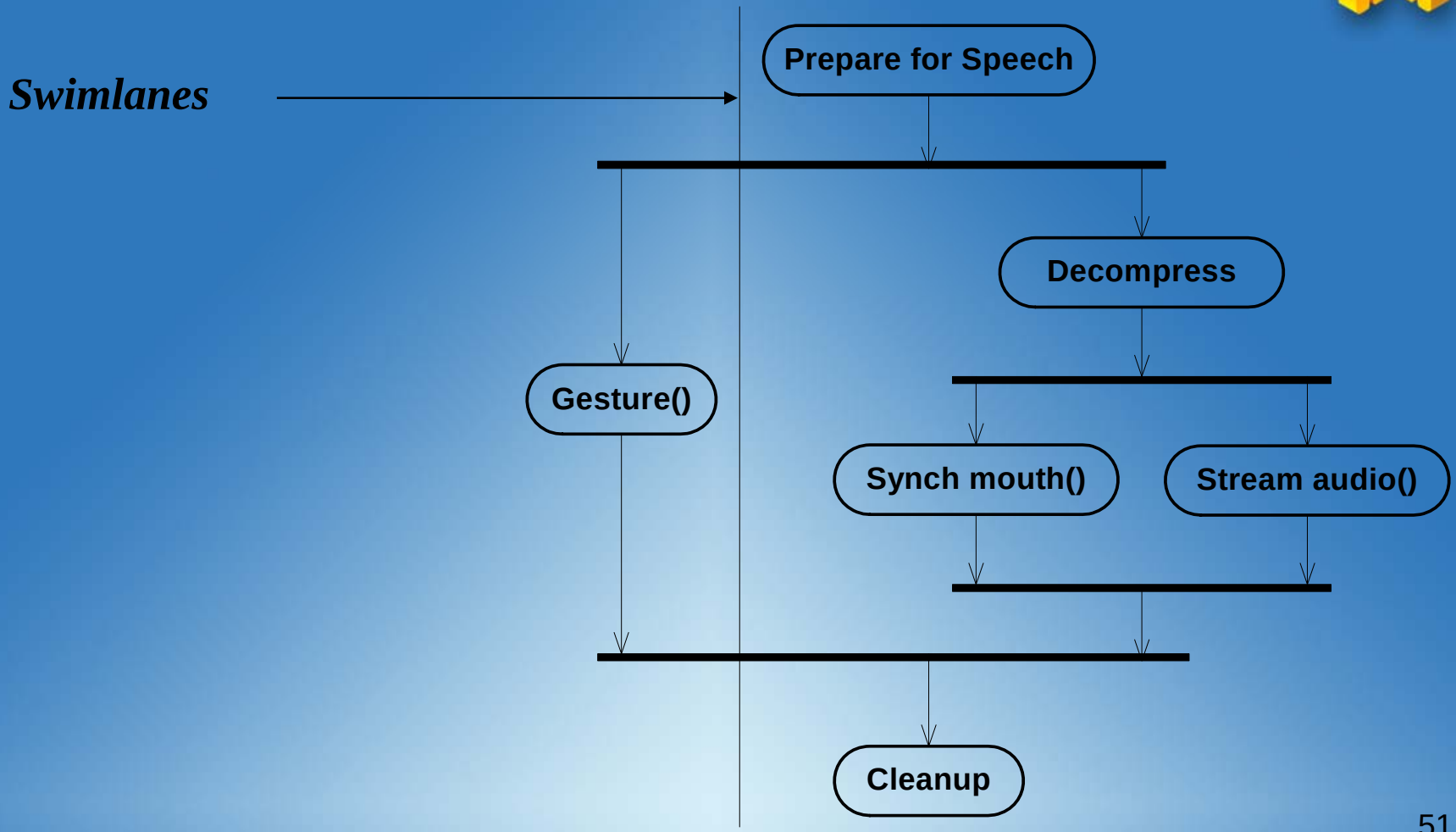
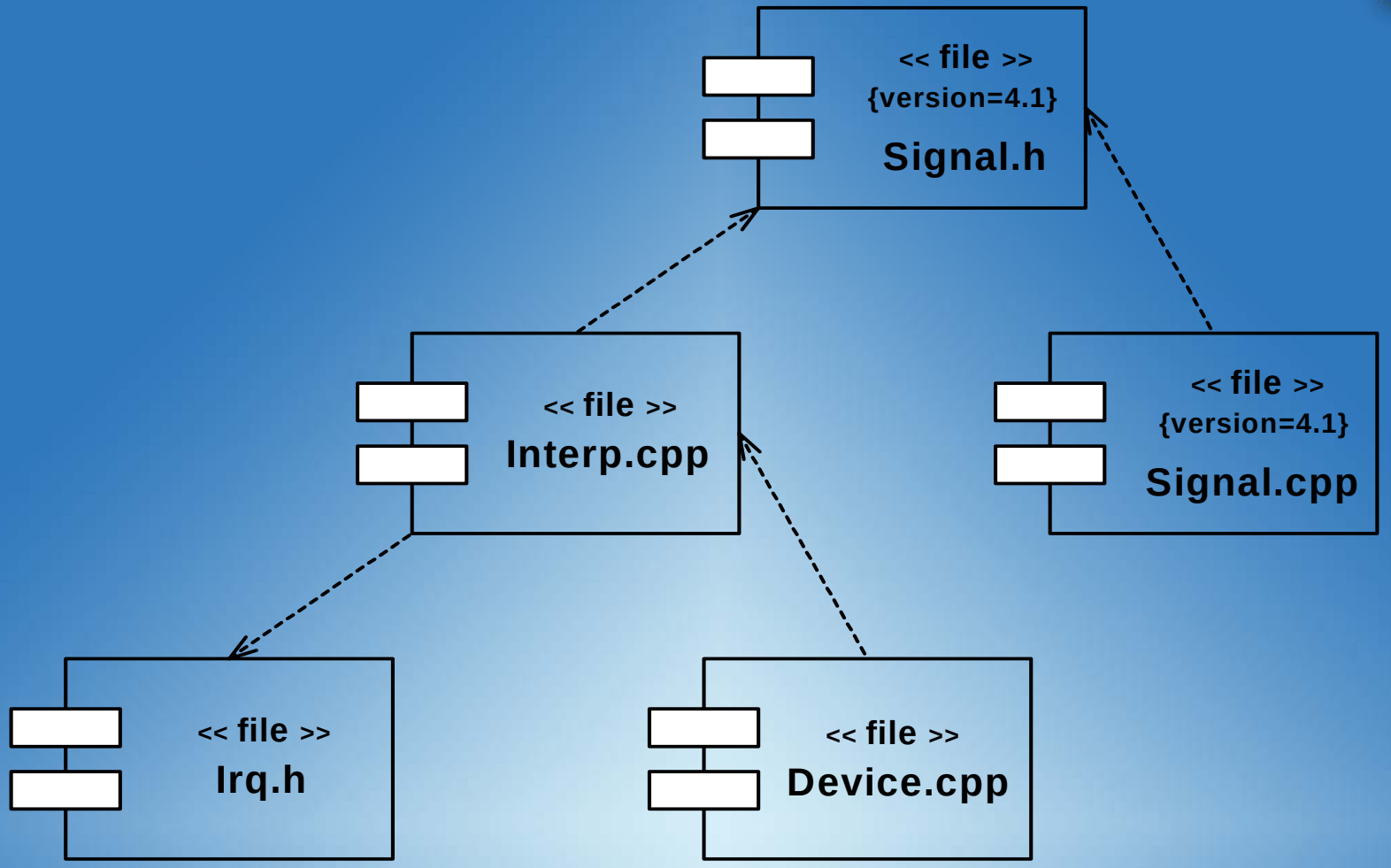


Diagram Komponen

- 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. 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.

Diagram Komponen

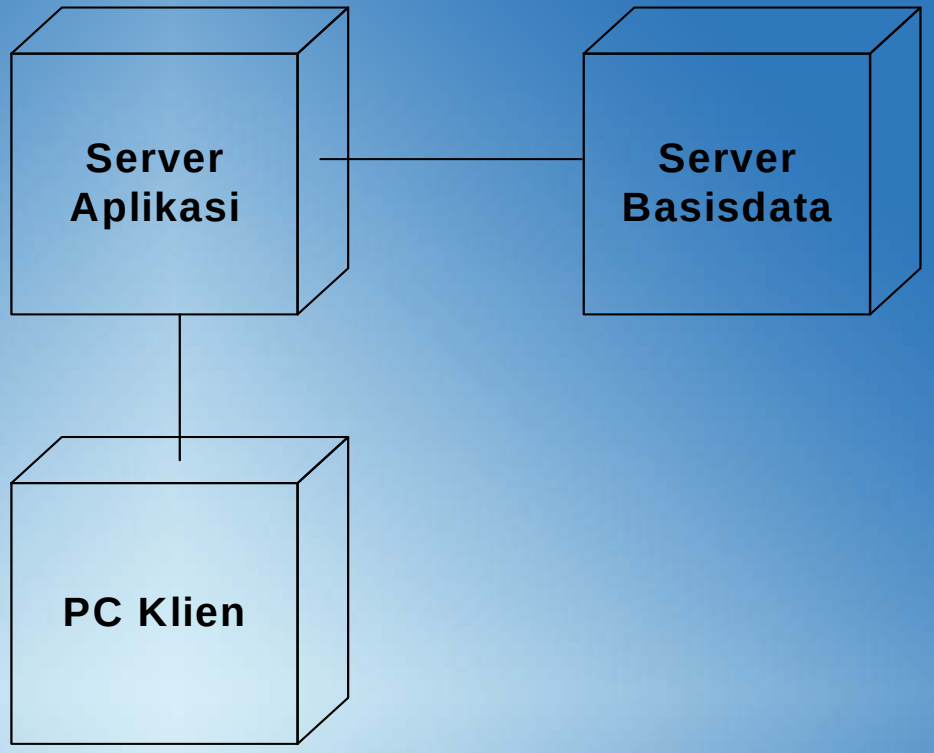


Deployment Diagram

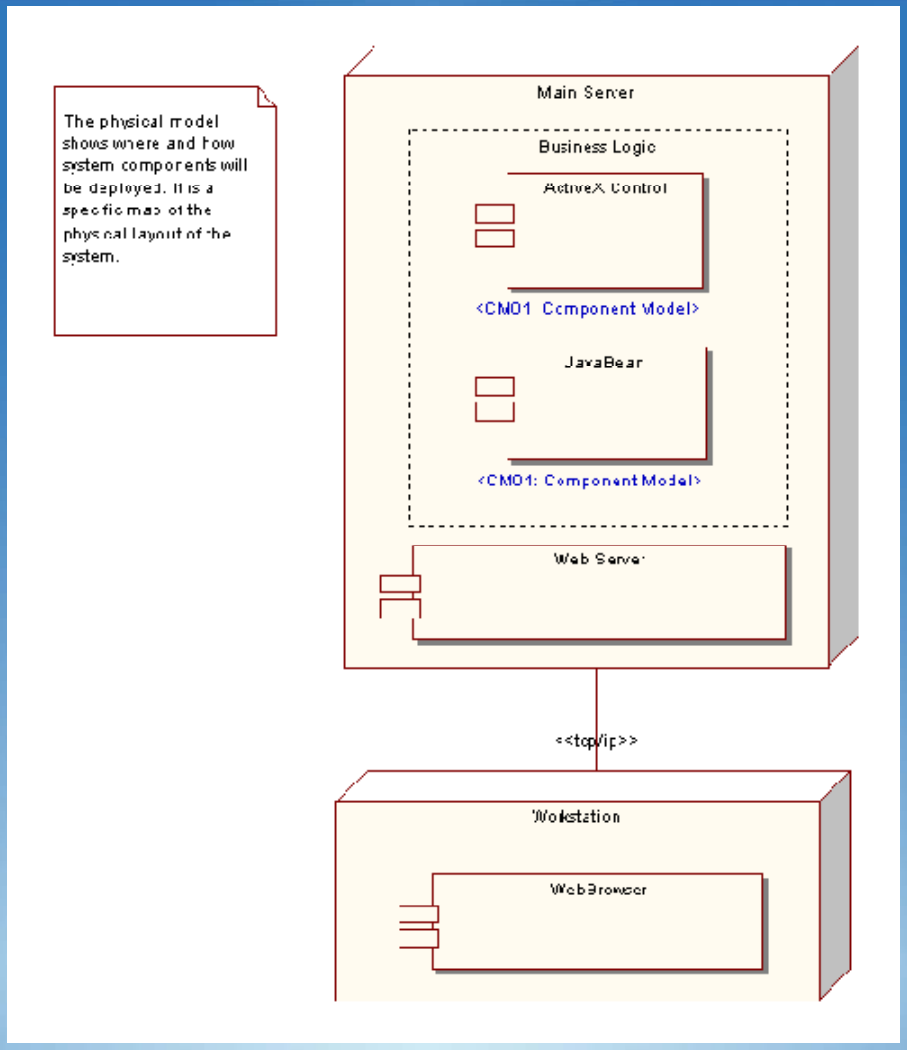
- Deployment/physical diagram menggambarkan detail bagaimana komponen di-deploy dalam infrastruktur sistem, di mana komponen akan terletak (pada mesin, server atau piranti keras apa), bagaimana kemampuan jaringan pada lokasi tersebut, spesifikasi server, dan hal-hal lain yang bersifat fisikal
- Sebuah 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.

Deployment Diagram

- Deployment diagram menunjukkan konfigurasi perangkat keras dan komponen-komponen PL yang ada di dalamnya



Deployment Diagram



Rules

Rules diperlukan untuk mendefinisikan well-formed model UML memiliki aturan-aturan untuk:

- Penamaaan: bagaimana memberi nama pada class, relasi, dan diagram
- Scope: Konteks yang dapat memberikan makna yang lebih spesifik bagi *thing {akan dibahas pada diagram kelas}*
- Visibility: berkaitan dengan akses sebuah elemen oleh elemen lain
- Integrity: bagaimana *things* berelasi secara konsisten
- Eksekusi: Bagaimana transformasi diagram

Common Mechanism

- Spesifikasi: Untuk setiap model perlu didefinisikan penjelasan tentang sintaks dan semantik dari model
- Adornment: Notasi-notasi pada UML mudah dan lengkap, sehingga model yang didefinisikan bisa dibaca dengan jelas
- Common division: selalu terdapat pasangan notasi/istilah/konsep untuk memodelkan sistem, mis.class dan objek, komponen dan instan komponen, use case dan realisasi use case, antarmuka dan implementasi, dan lain-lain
- Extensibility: memberi keleluasaan untuk memperluas bahasa secara terkontrol

Extensibility

Stereotypes digunakan untuk mengklasifikasikan dan memperluas sosiasi,relasi pewarisan, kelas, dan komponen

- Nama ditulis dalam *<<stereotype>>*
- Contoh:
  - Class stereotypes: boundary, control, entity, utility, exception
  - Inheritance stereotypes: uses and extends
  - Component stereotypes: subsystem

Tagged values — mendefinisikan informasi pada elemen

anObject:ClassA
{location=server)

Constraints — memperluas semantik dari blok pembangun UML

- {if Order.customer.credit.Rating is “poor” then Order.isPrepaid must be true}

Penggunaan Notasi UML

- Menggambarkan batasan sistem dan fungsi-fungsi utamanya dengan diagram use case
- Buat realisasi use case dengan diagram interaksi
- Gambarkan struktur statik sistem dengan diagram kelas
- Modelkan perilaku objek dengan state transition diagram
- Gambarkan arsitektur implementasi dengan diagram komponen dan deployment
- Perluas fungsionalitas dengan stereotypes