

PEMROGRAMAN BERORIENTASI OBJEK

Materi 11

Royana Afwani

Teknik Informatika Universitas Mataram



Materi

- **Unified Modelling Language (UML)**

Aplikasi yang baik dan handal = yang dikerjakan sendiri atau team ?

Bagaimana Aplikasi yang sedang dibuat dalam team bisa dimengerti bersama ?

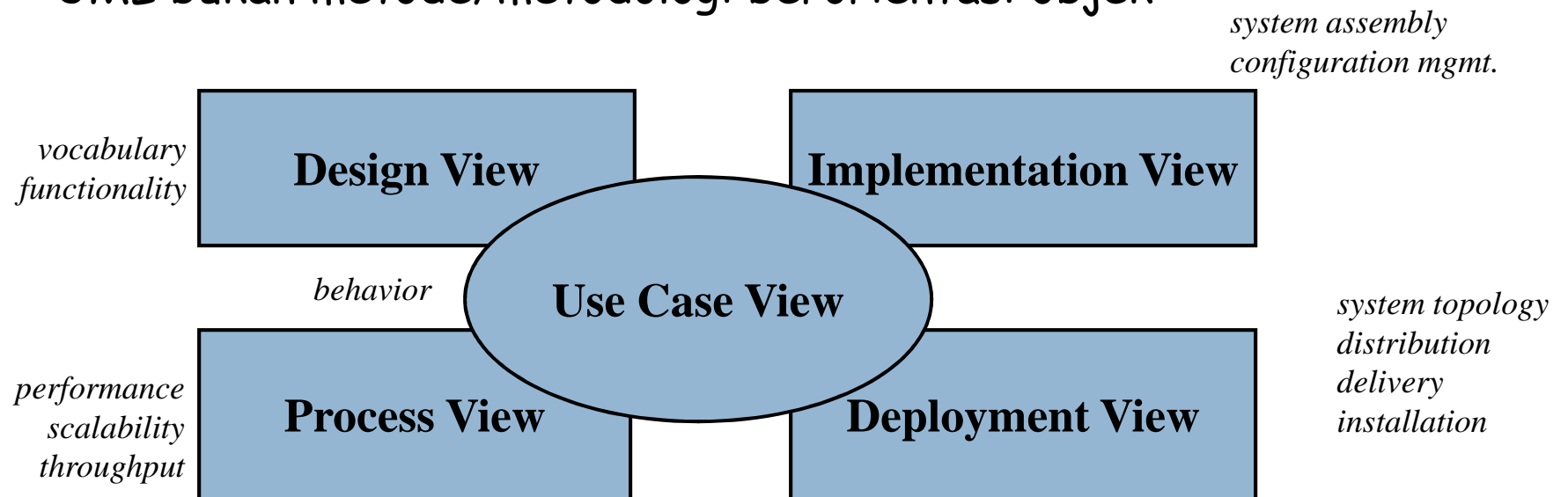
Bagaimana Aplikasi yang sudah dibuat bisa Dikembangkan/diupdate/di revisi kemudian hari ?

Bagaimana Aplikasi yang sedang dibuat bisa dimengerti oleh orang lain/orang baru untuk dikembangkan ?

RPL + APBO + APSI + PPL + dll

Arsitektur Sistem dan Sudut Pandang

- UML adalah bahasa untuk
 - Visualisasi
 - Spesifikasi
 - Konstruksi
 - Dokumentasi
- UML bukan metode/metodologi berorientasi objek



Blok Pembangun pada UML

5

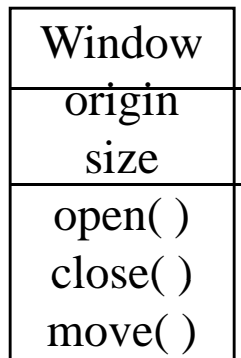
- Things
abstraksi dari apa yang akan dimodelkan
- Relationship
hubungan antar abstraksi (*things*)
- Diagrams
mengelompokkan kumpulan sejumlah abstraksi yang dihubungkan

I. Things

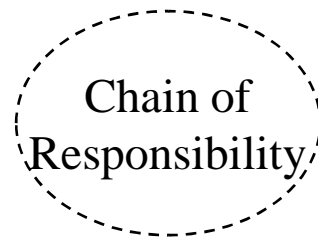
- *Structural* (berpadanan dengan kata benda) merepresentasikan aspek statis sistem
- *Behavioural* (berpadanan dengan kata kerja) merepresentasikan aspek dinamis sistem
- *Grouping*
menyatakan pengelompokkan sejumlah abstraksi dengan organisasi tertentu

Structural Things

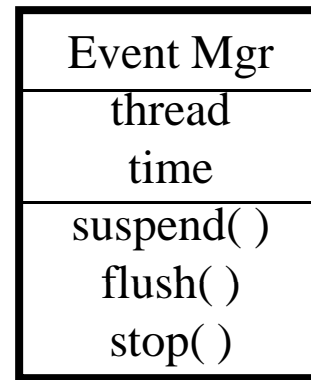
Class



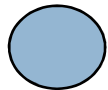
Collaboration



Active Class



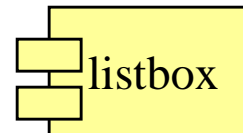
Node



IWindow
Interface



Use Case



Component

Structural Things

8

- **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 kelakuan kooperatif agregat
- **Use case**
deskripsi dari himpunan langkah aksi yang dilakukan sistem yang menghasilkan luaran kepada aktor tertentu
- **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

9

Bagian dinamik dari model UML

Biasanya terhubung dengan model struktural. Ditulis dalam kata kerja.

Ada 2 macam:

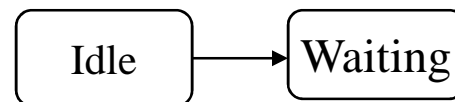
Interaksi

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



State Machine

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

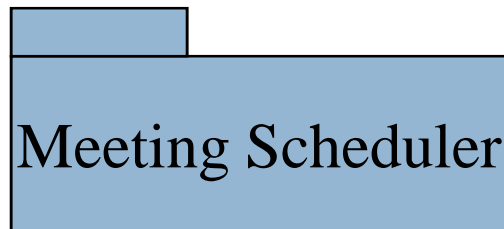


Grouping & Annotational Things

10

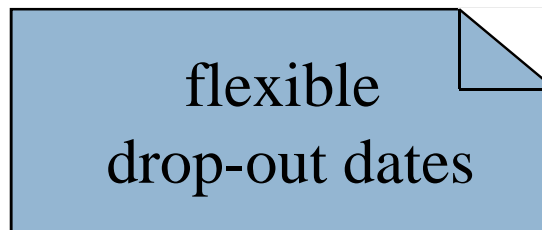
Packages

- 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



II. Relationships

11

4 jenis

- Dependensi
- Asosiasi
- Generalisasi

Relationships

12

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

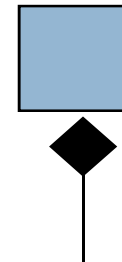
employer

employee

0..1

Aggregasi

jenis khusus dari asosiasi (menyatakan whole part)

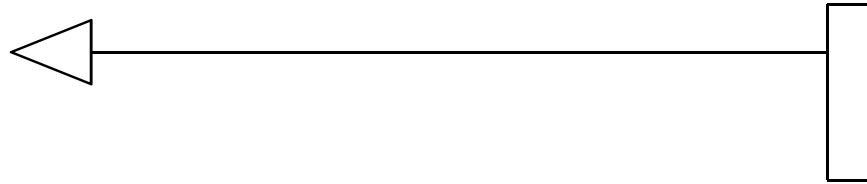


Relasi

13

Generalisasi

Relasi antar objek yang memiliki hubungan general-spesial



III. Diagrams

14

- 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 Modelling



- Menentukan berbagai hasil yang akan di komputasi oleh produk perangkat lunak dengan mengabaikan urutan pembuatannya. (*clasical object oriented analisys*)
- Use case diagram dihubungkan dengan skenario yang dibuat pada tahap requirement analisys

Contoh: Sistem registrasi Universitas

16

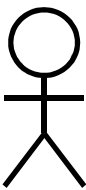
The UTD wants to computerize its registration system

- ▣ 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

17


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




Registrar



Professor



Student

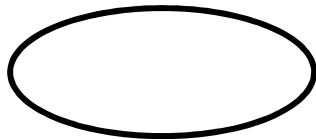


Billing System

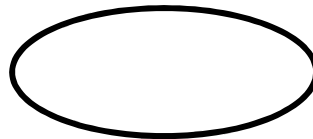
Contoh: Sistem registrasi Universitas

18

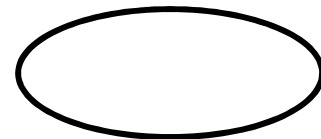
- Sebuah use case menunjukkan perilaku sistem
- Aktor:
 - Registrar -- mengelola kurikulum
 - Professor - menentukan MK yang akan ditawarkan dan meminta daftar MK
 - Student - mengelola jadwal
 - Billing System - menerima informasi tagihan



Maintain Curriculum



Request Course Roster

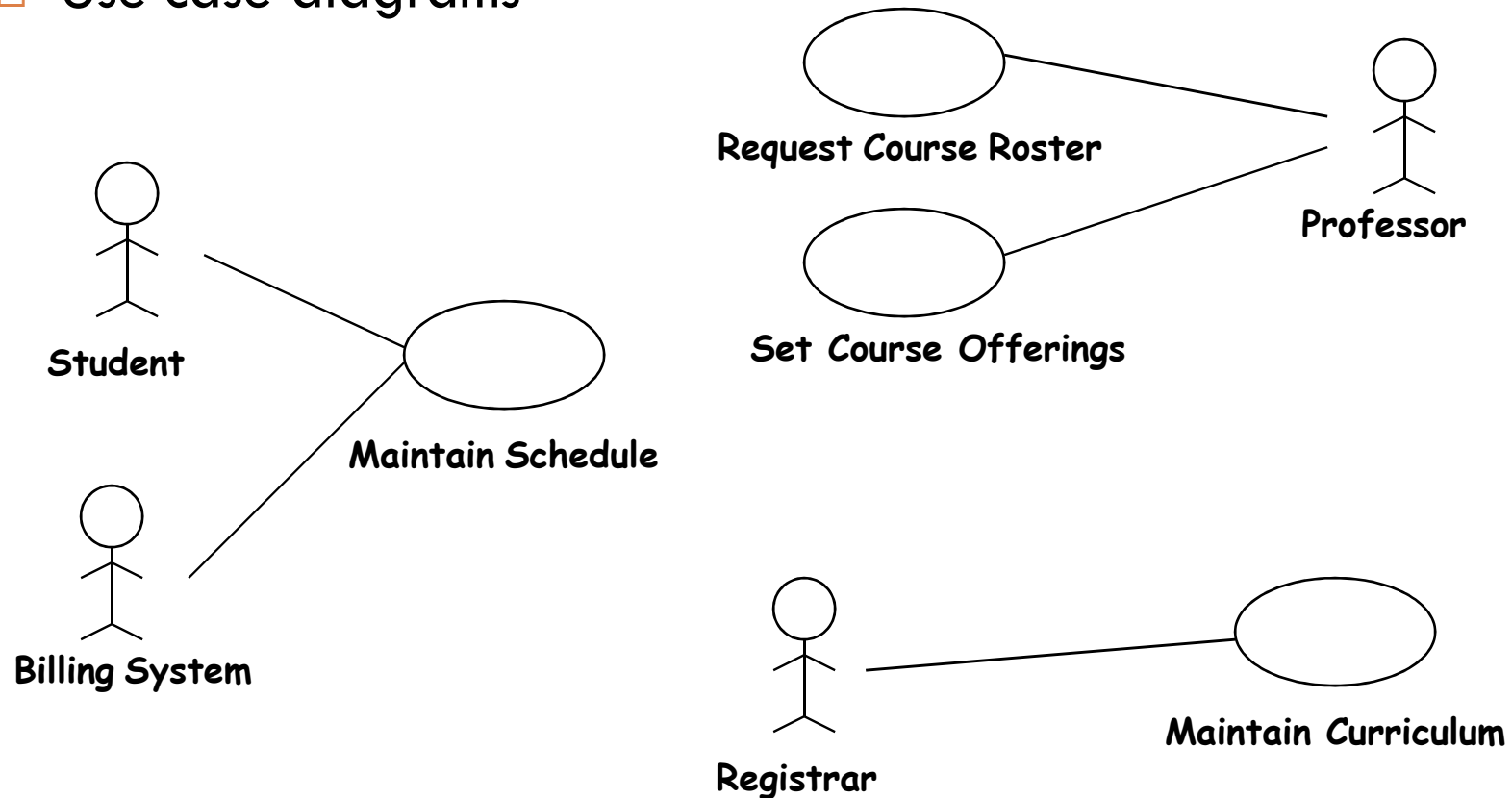


Maintain Schedule


Contoh: Sistem registrasi Universitas




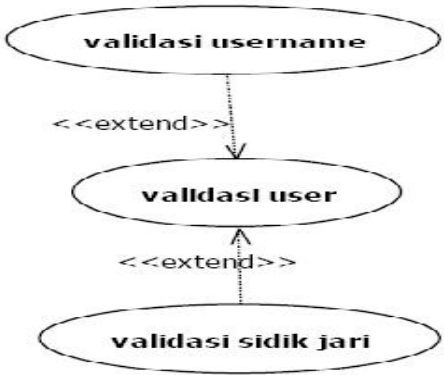
19


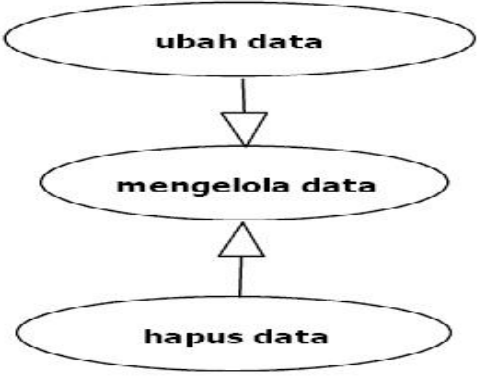


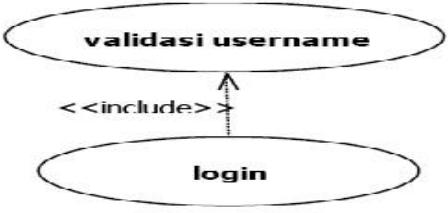
□ Use case diagrams

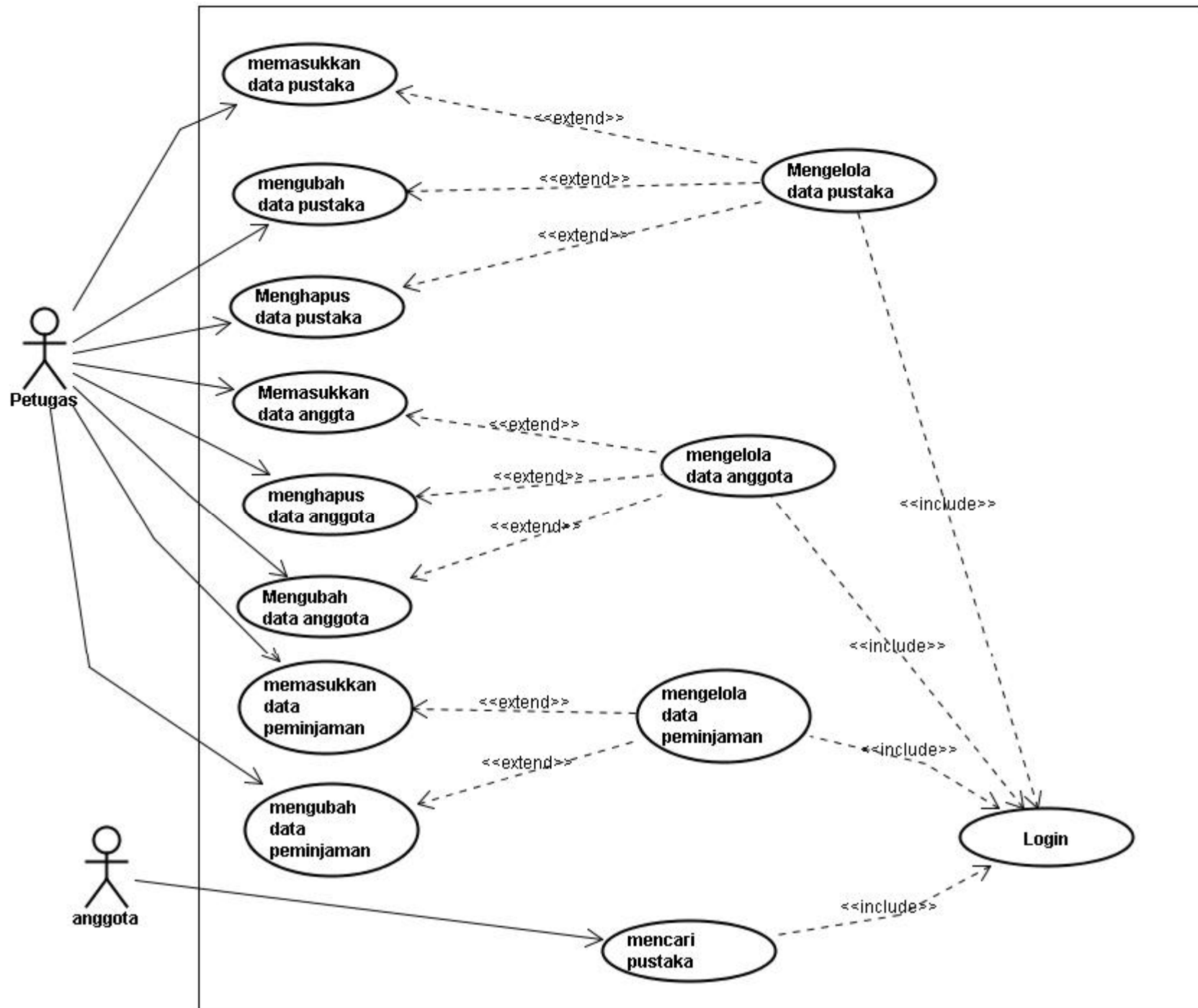


Simbol-Simbol Use Case

Simbol	Deskripsi
<i>Use case</i> 	fungsi yang disediakan sistem sebagai unit-unit yang saling bertukar pesan antar unit atau aktor; biasanya dinyatakan dengan menggunakan kata kerja di awal frase nama <i>use case</i>
<i>Aktor / actor</i>	orang, proses, atau sistem lain yang berinteraksi dengan sistem informasi yang akan dibuat di luar sistem informasi yang akan dibuat itu sendiri, jadi walaupun simbol dari aktor adalah gambar orang, tapi aktor belum tentu merupakan orang;

Simbol	Deskripsi
 <p>nama aktor</p>	<p>biasanya dinyatakan menggunakan kata benda di awal frase nama aktor</p>
<p>Asosiasi / <i>association</i></p> 	<p>komunikasi antara aktor dan <i>use case</i> yang berpartisipasi pada <i>use case</i> atau <i>use case</i> memiliki interaksi dengan aktor</p>
<p>Ekstensi / <i>extend</i></p> 	<p>relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan dapat berdiri sendiri walau tanpa <i>use case</i> tambahan itu; mirip dengan prinsip <i>inheritance</i> pada pemrograman berorientasi objek; biasanya <i>use case</i> tambahan memiliki nama depan yang sama dengan <i>use case</i> yang ditambahkan, misal</p>  <p>arah panah mengarah pada <i>use case</i> yang ditambahkan</p>

Simbol	Deskripsi
	<p>dimana fungsi yang satu adalah fungsi yang lebih umum dari lainnya, misalnya:</p>  <p>arah panah mengarah pada <i>use case</i> yang menjadi generalisasinya (umum)</p>
<p>Menggunakan / <i>include</i> / <i>uses</i></p> <p><<include>></p>  <p>«uses»</p> 	<p>relasi <i>use case</i> tambahan ke sebuah <i>use case</i> dimana <i>use case</i> yang ditambahkan memerlukan <i>use case</i> ini untuk menjalankan fungsinya atau sebagai syarat dijalankan <i>use case</i> ini</p> <p>ada dua sudut pandang yang cukup besar mengenai include di <i>use case</i>:</p> <ul style="list-style-type: none"> • include berarti <i>use case</i> yang ditambahkan akan selalu dipanggil saat <i>use case</i> tambahan dijalankan, misal pada kasus berikut: 



Realisasi Use Case

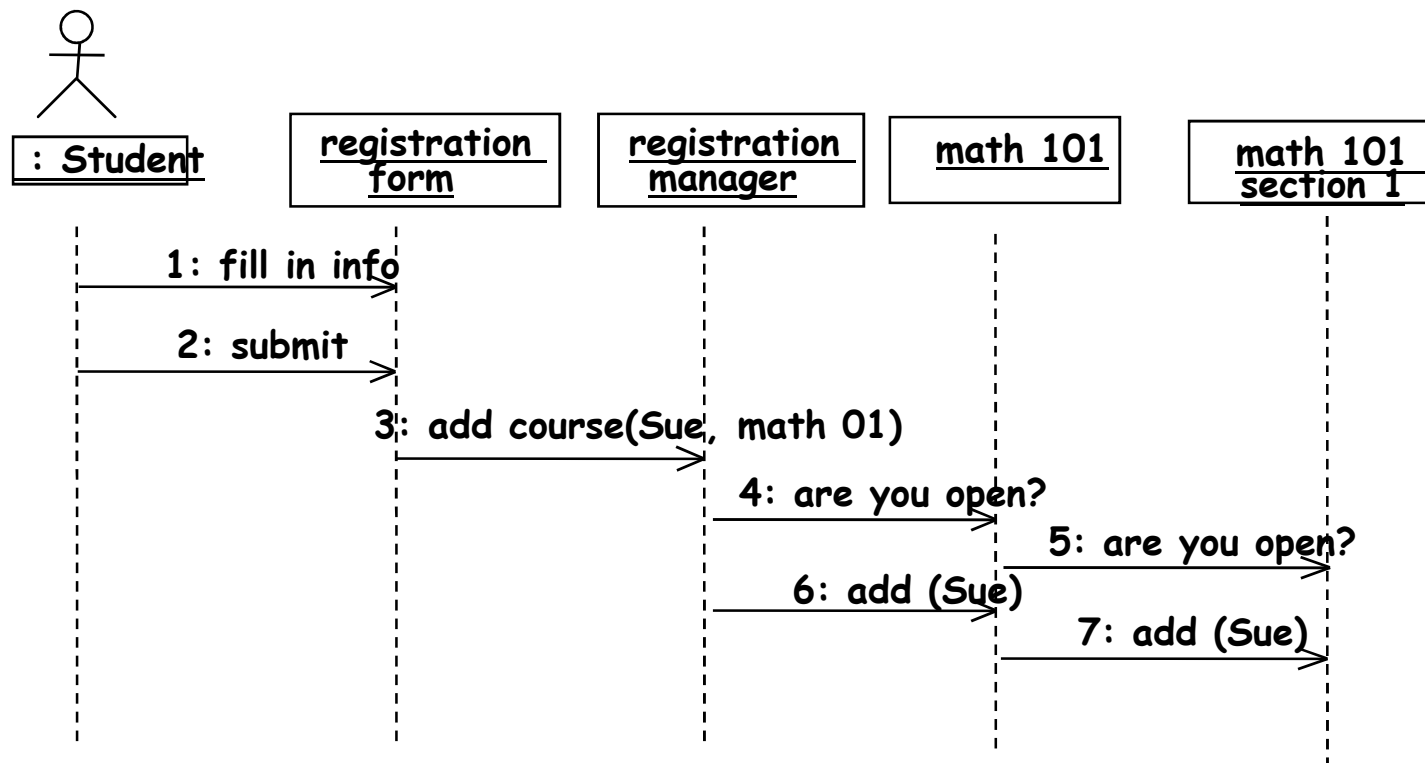
24

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 dynamic view dari sistem
- Ada 2 jenis:
 - ▣ Diagram sekuens
 - ▣ Diagram kolaborasi

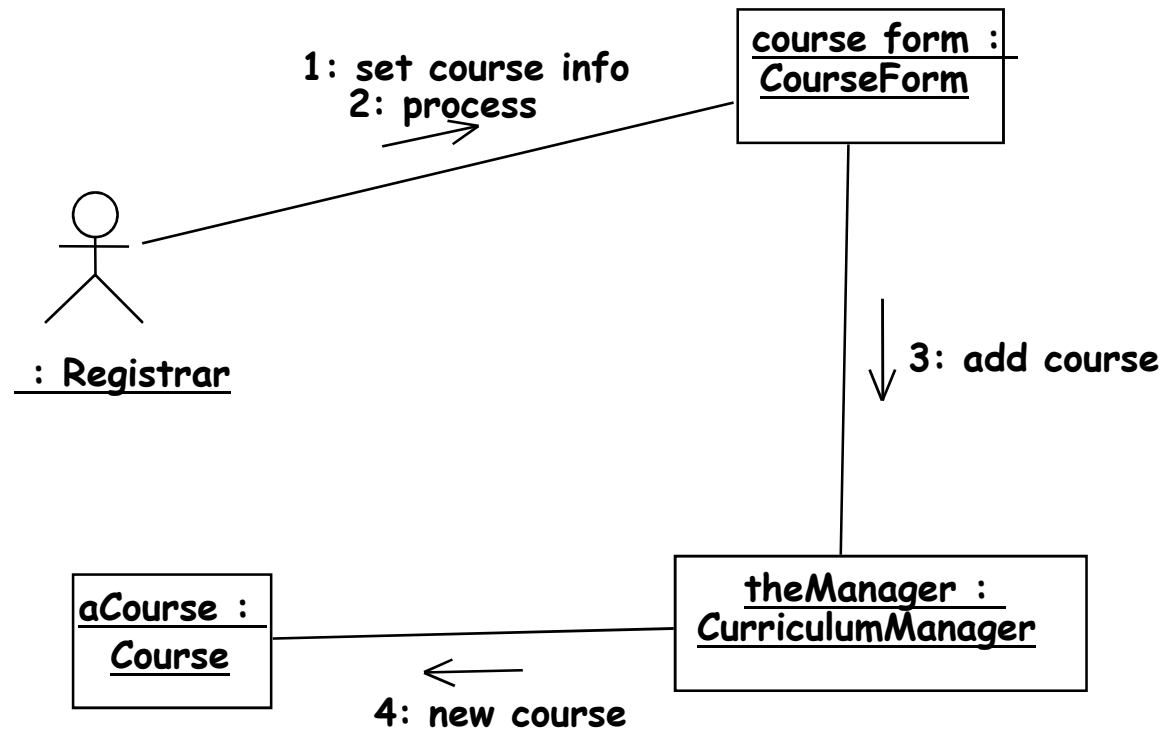
Diagram Sekuens

25

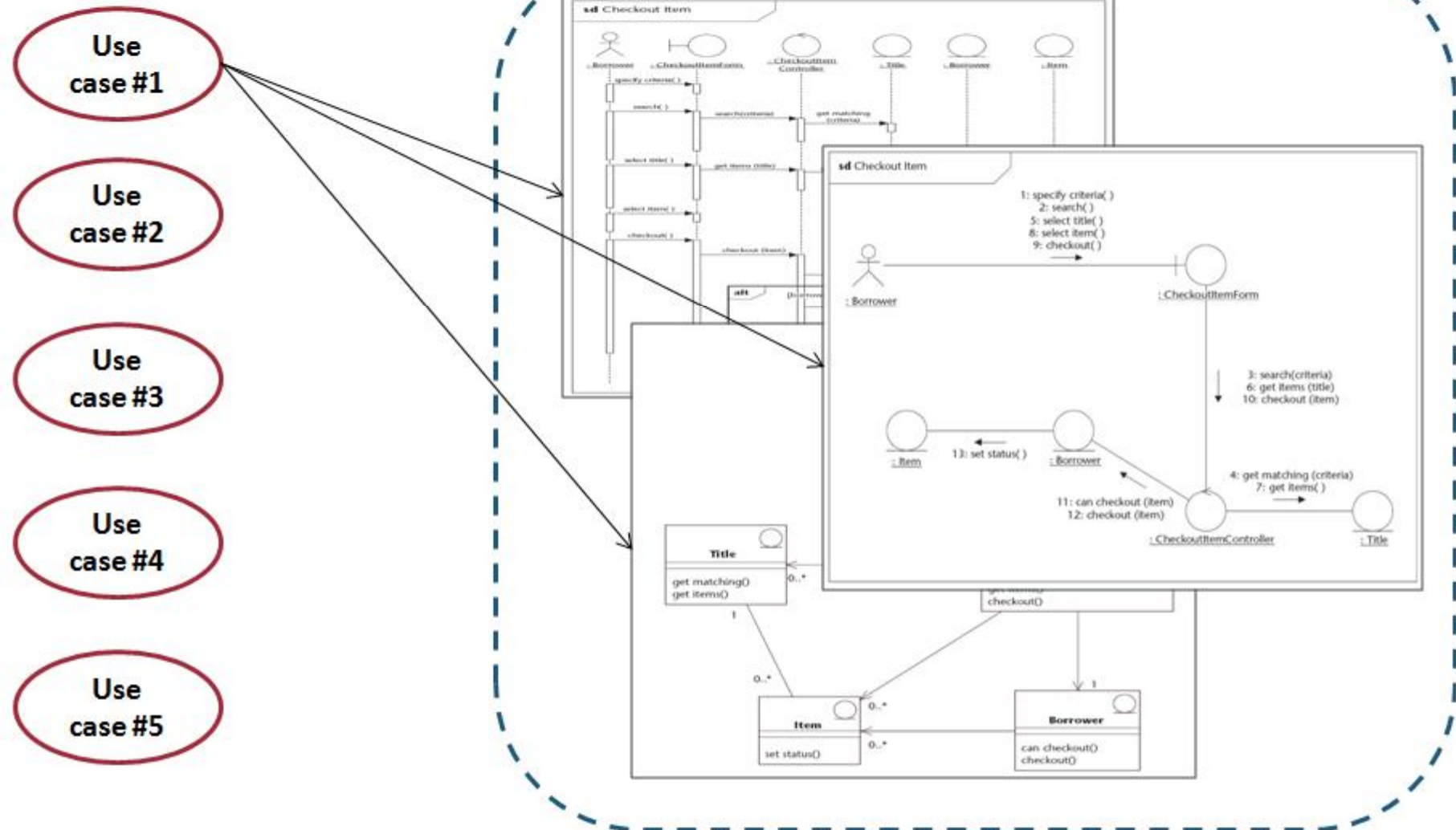


Collaboration Diagram

26



Repeat this process for all use case



Class Modelling

- Menentukan kelas, atribut dan hubungan antar kelas.
- Pada tahap ini belum menentukan method method (fungsi) karena akan dilakukan pada tahap desain – *prakteknya kadang bisa dilanggar*
- Salah satu cara penentuan kelas adalah mencari kandidat kelas yang berasal dari use case

Diagram Kelas

29

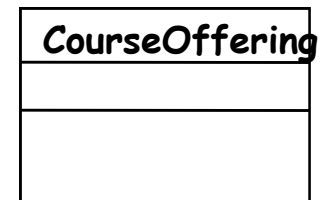
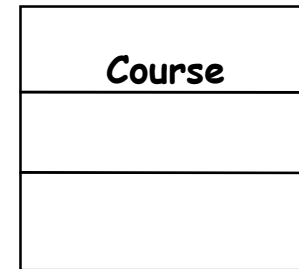
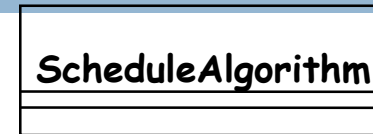
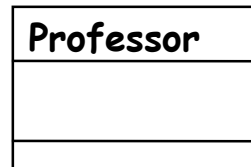
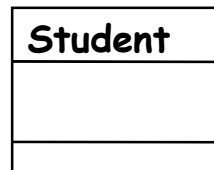
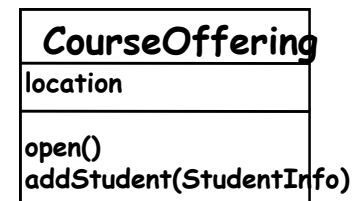
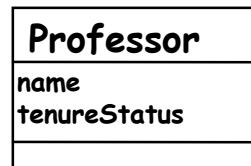
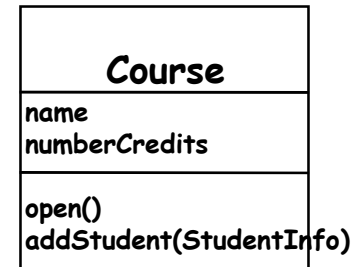
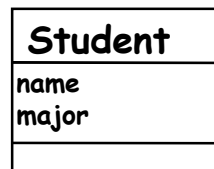
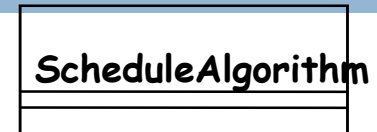
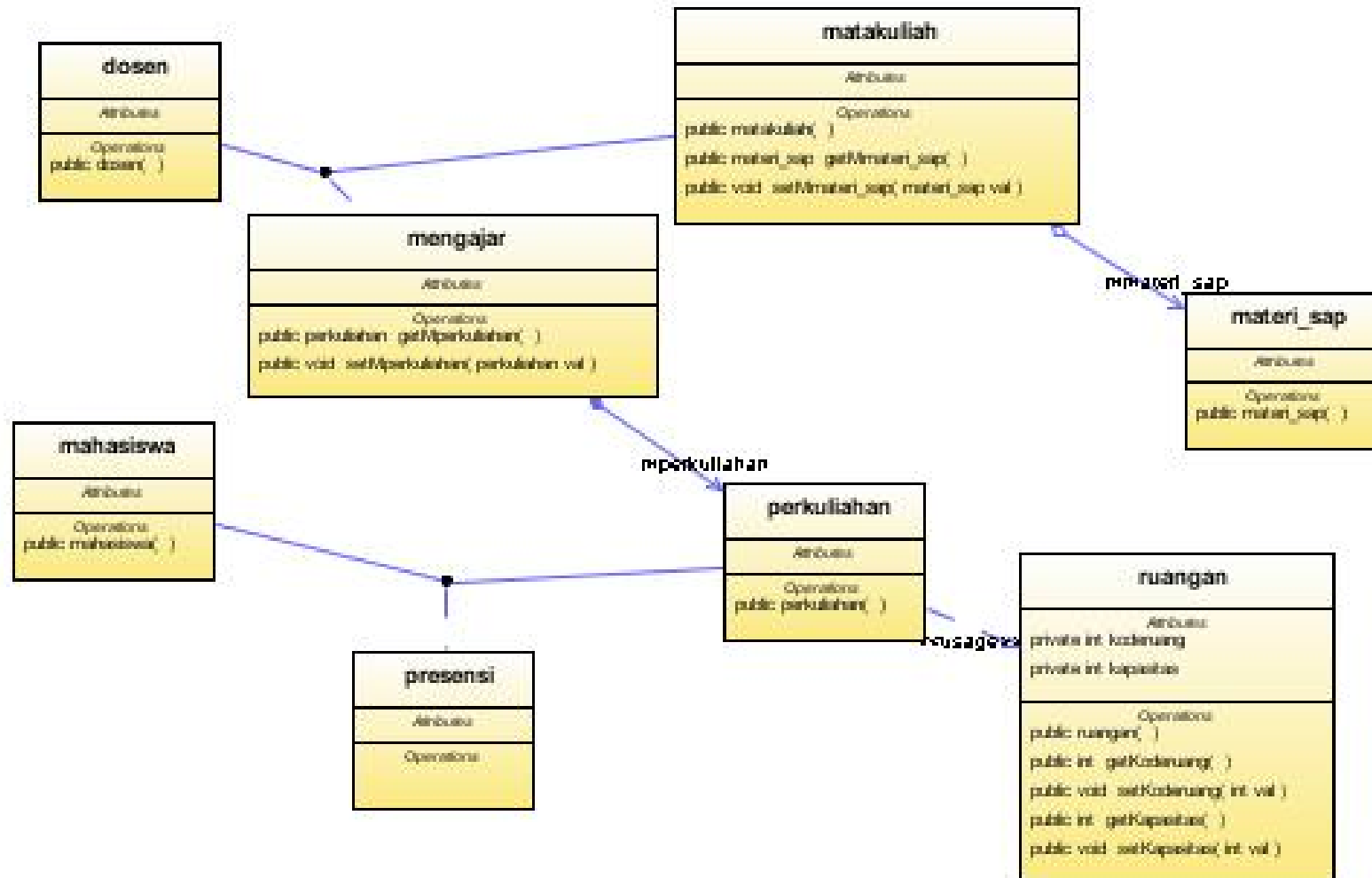


Diagram Kelas

30



© 2013 Pearson Education, Inc. or its affiliate(s). All rights reserved.

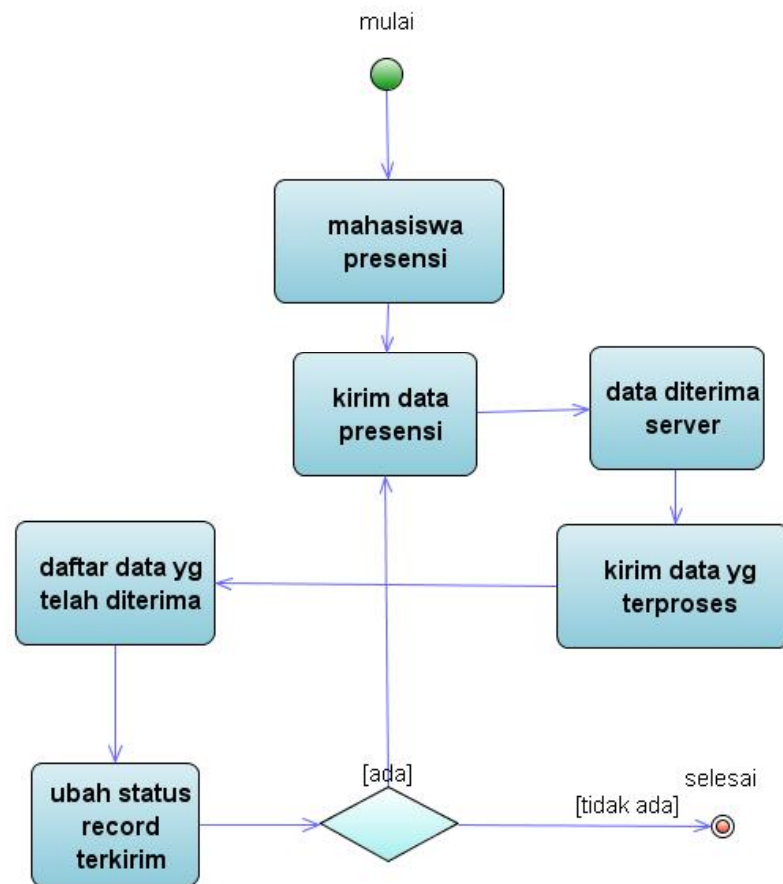


Dinamic Modelling



- Menentukan aksi yang di lakukan oleh masing masing kelas dan subkelas
- Menggunakan state diagram.
- Langkah yang ada pada method main (biasanya)

State diagram presensi



Design Phase (Object Oriented Design)

- Membuat diagram interaksi untuk setiap skenario
→ sequence diagram, collaboration diagram
- Melengkapi diagram kelas secara detail termasuk method methodnya, class diagram
- Merancang produk
 - ▣ Component diagram
 - ▣ Deployment Diagram

Sequence Diagram

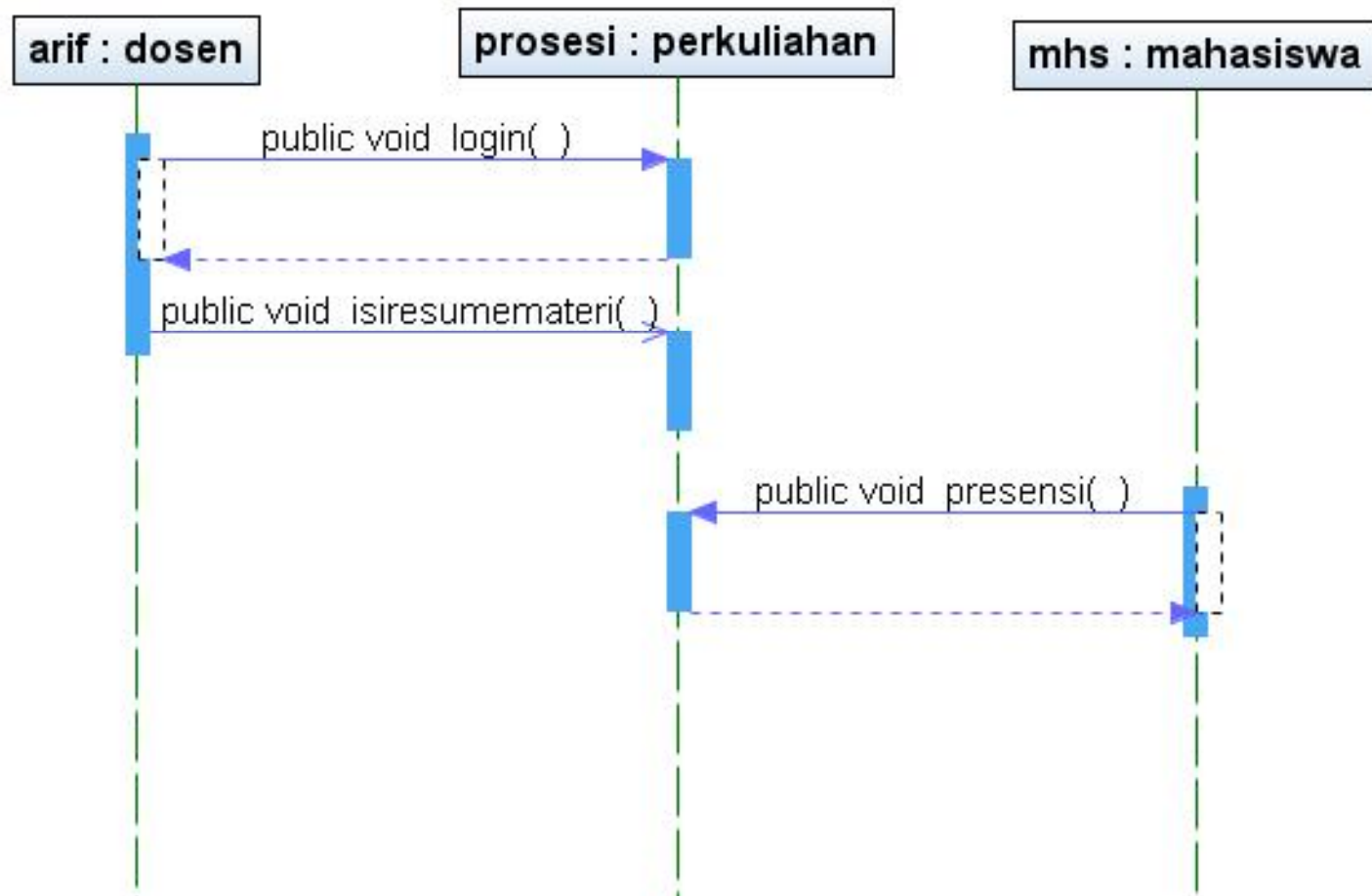
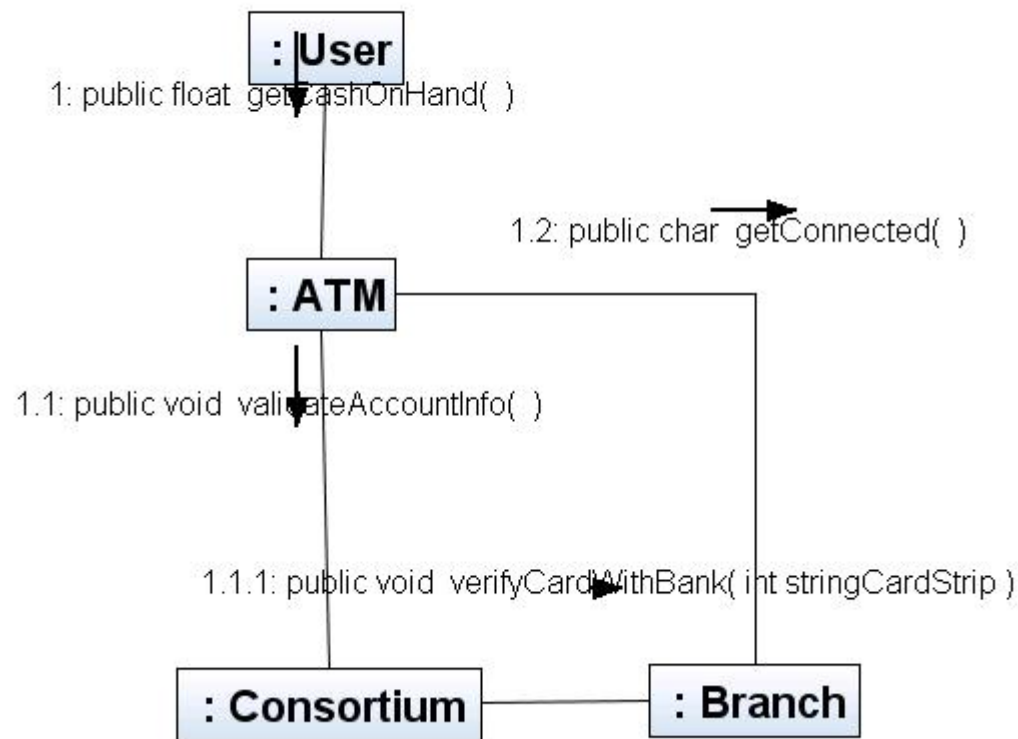


Diagram Kolaborasi



Deployment Diagram

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

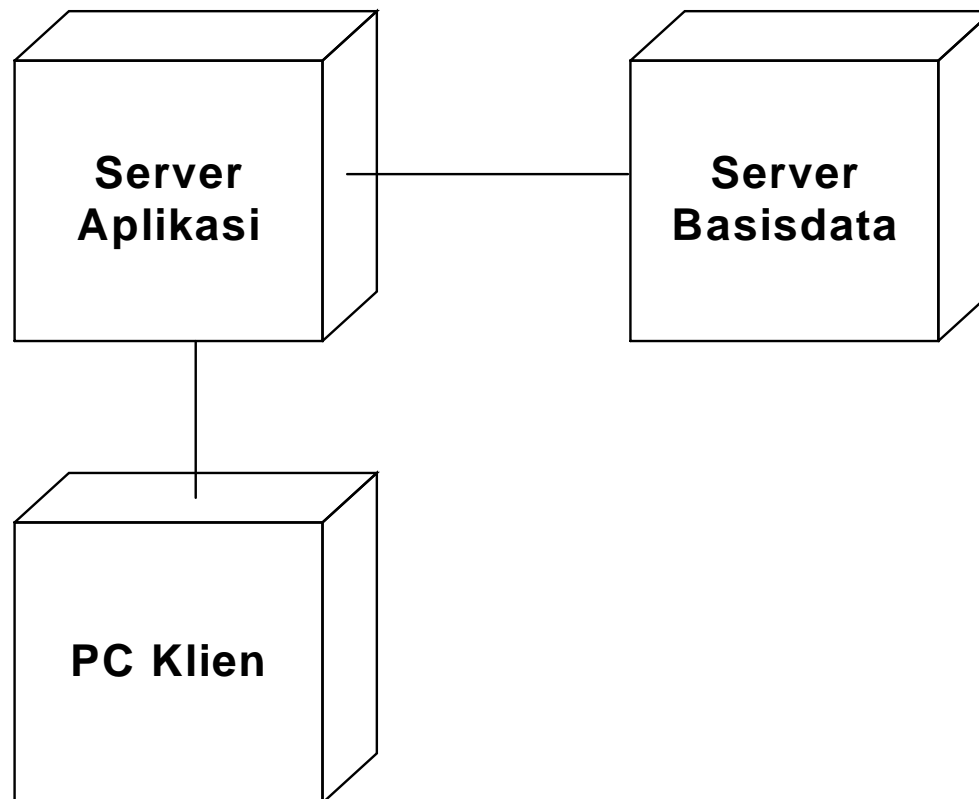
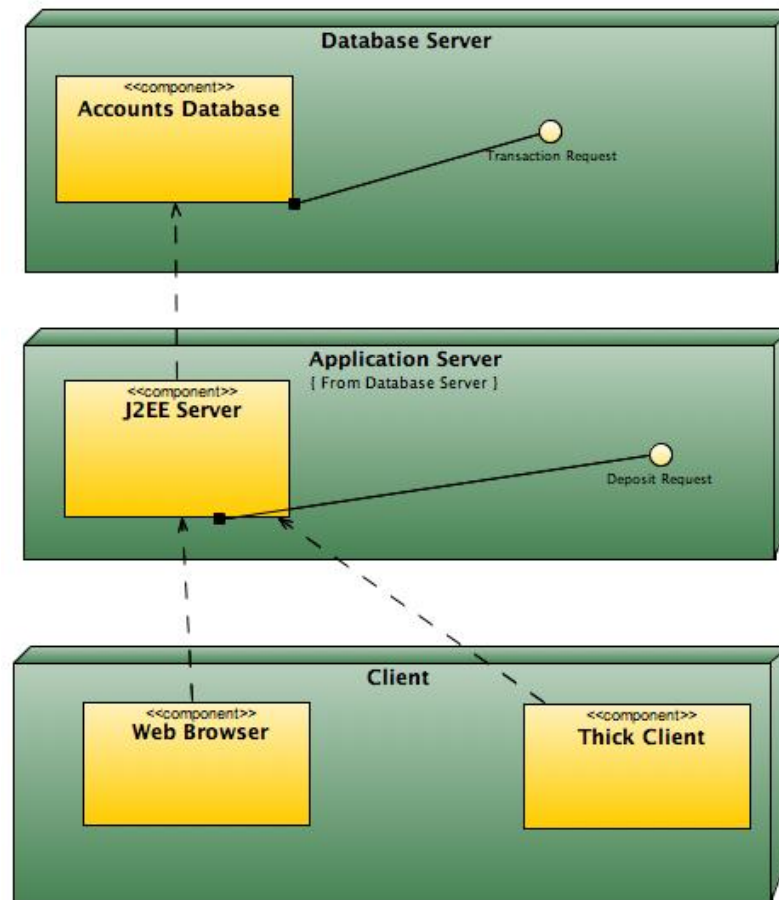


Diagram Deployment



KONSENTRASI KE KELAS



- Kelas merupakan dokumen yang paling dekat dengan program yang akan menjadi produk dari pengembangan aplikasi berbasis obyek
- Dokumen lain dalam tahap pengembangan perangkat lunak berorientasi obyek sebenarnya harus di buat, bahkan jika diperlukan boleh menambahkan dokumen lain seperti ERD, DFD atau dokumen lain yang dianggap perlu untuk memperjelas rancangan.

Penggunaan Notasi UML

40

- ❑ 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 // lebih lengkap pada mata kuliah analisa PBO

References for Coding

For coding, the specifications are collected from the following diagrams in the design model:

- **Class diagrams.** The class diagrams in which the class is present, showing its static structure and relationship to other classes.
- **State machine diagram.** A state machine diagram for the class, showing the possible states and the transitions that need to be handled (along with the operations that trigger the transitions).
- **Dynamic diagrams (sequence, communication, and activity)** in which objects of the class are involved. Diagrams showing the implementation of a specific method in the class or how other objects are using objects of the class.
- **Use-case diagrams and specifications.** Diagrams that show the result of the system give the developer more information on how the system is to be used when he or she might be getting lost in details—losing sight of the overall context.





TERIMA KASIH