

ANALISA & DESAIN SISTEM INFORMASI

02. Perencanaan Sistem

03. Analisa Sistem

04. Analisa Kebutuhan Informasi

05. Analisa dan Perancangan Proses Bisnis

07. Perancangan Model Data

Doni Abdul Fatah
github.com/doniaft
Universitas Trunojoyo Madura

Pokok Bahasan

01. Pengenalan ADSI

02. Perencanaan Sistem

03. Analisa Sistem

04. Analisa Kebutuhan Informasi

05. Analisa dan Perancangan Proses Bisnis

06. Desain & Rekayasa Ulang Proses Bisnis

07. Perancangan Model Data

08. Perancangan Berorientasi Objek

09. Perancangan Antarmuka

10. Manajemen Proyek Sistem Informasi

11. Pembuatan Prototype Sistem Informasi

12. UAS

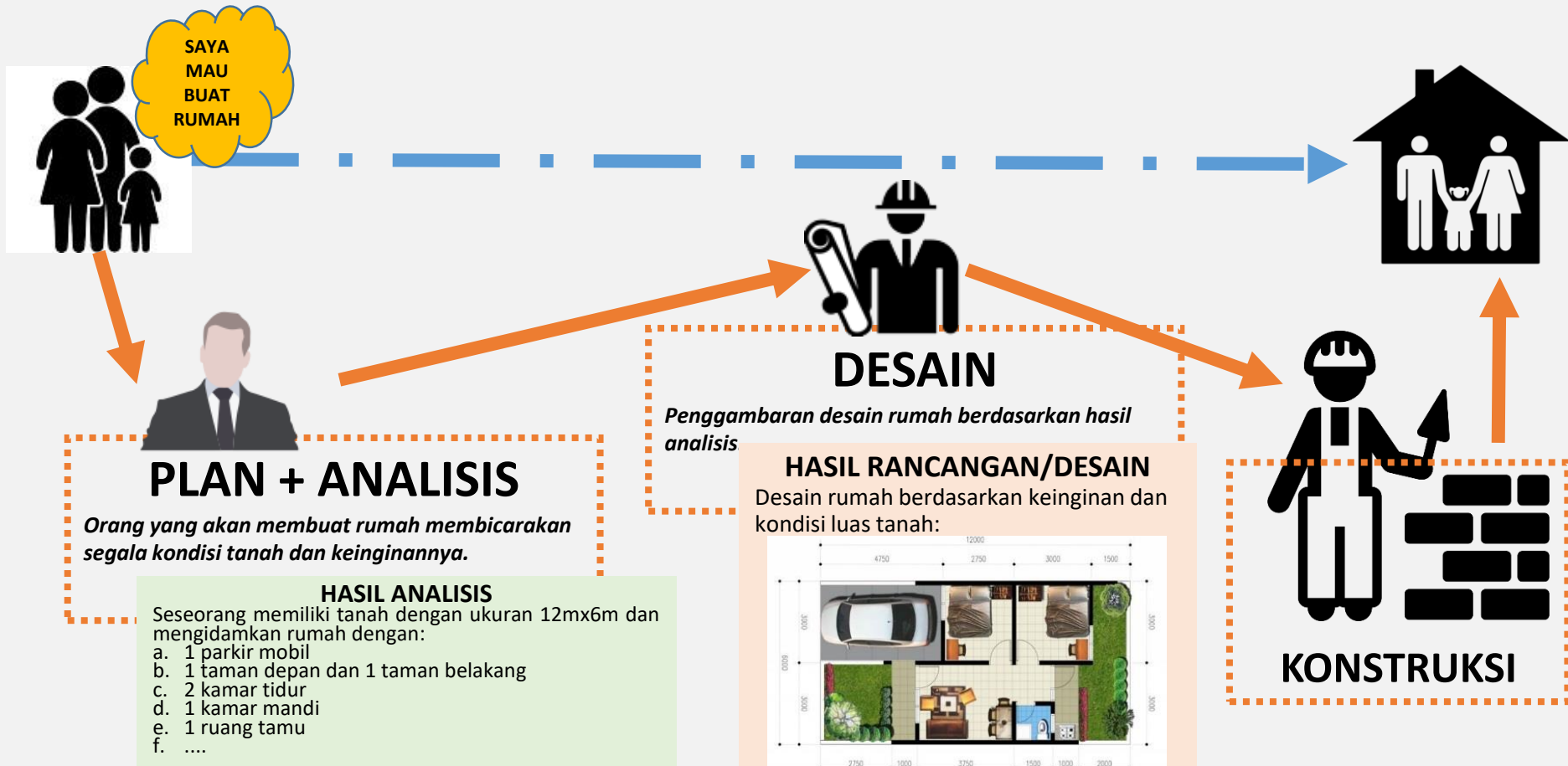
01. Analisa Desain Sistem Informasi

- 1) Analisa Sistem
- 2) Analisa Kebutuhan Informasi
- 3) Analisa dan Perancangan Proses Bisnis
- 4) Kontrak Perkuliahan
- 5) Contact
- 6) Referensi

Analisis & Teknik Pengumpulan Requirement

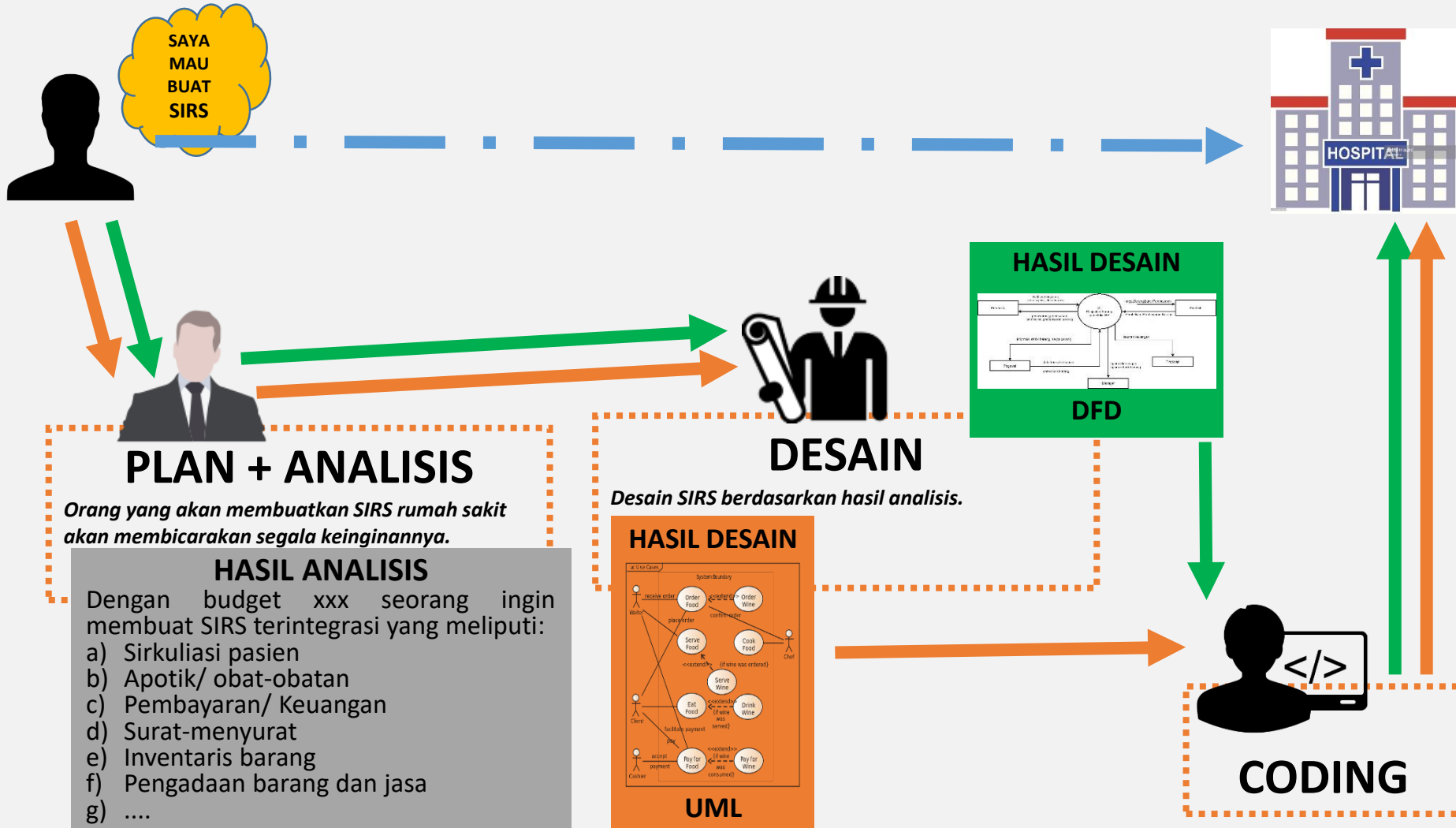
Analisis (dalam dunia konstruksi bangunan)

adalah kajian yang dilaksanakan terhadap sebuah bahasa guna meneliti struktur bahasa tersebut secara mendalam. (*wikipedia*)



Analisis (dalam dunia pembuatan software)

- ❑ Biasa disebut dengan **analisis kebutuhan**.
- ❑ Kegiatan analisis kebutuhan **dimulai dari bertemu client sampai menerjemahkannya ke desainer**.

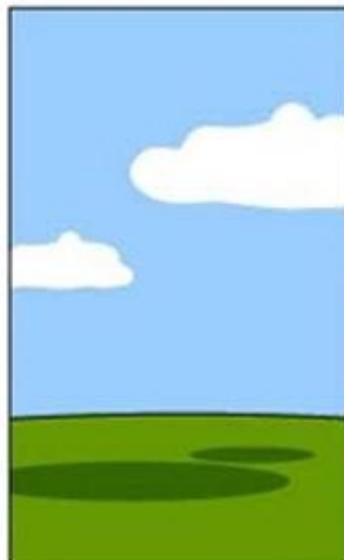




The customer's initial vision.



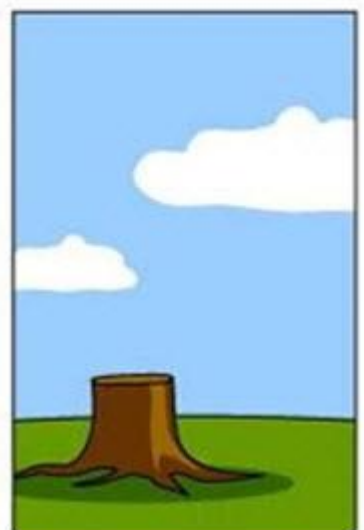
The team generates energy around the vision.



How the team started.



How the first build in the iteration looked.



What the customer reviewed after the 1st iteration.



How the next build in the next iteration looked.



How another build in the next iteration looked.



What the customer reviewed after the 2nd iteration.



How another build in the next iteration looked.



What the customer really needed ... and accepted in the final iteration.

Pihak yang Berkontribusi dalam Analisis

- ❑ **Client/Customer** yang meminta pengembangan *software* dengan tujuan meningkatkan produktivitas karyawan.
- ❑ **User** adalah orang yang familiar dengan sistem saat ini dan akan menggunakan sistem dimasa mendatang, mereka merupakan *expert* sistem sekarang, dan menjadi subyek dari sistem improvement.
- ❑ **Pakar**, mereka yang familiar dengan permasalahan terkait dengan software yang akan diotomatisasi, misalnya konsultan keuangan.
- ❑ **Peneliti Market** yang melakukan survei terkait dengan *trend* masa depan dan potensial kebutuhan *customer*.
- ❑ **Pengacara atau auditor** yang familiar dengan tata kelola, keamanan dan requirement terkait legal.
- ❑ **Software engineer atau technology expert** yang mengerti kelayakan teknis dan ekonomi.

Cara Mendapatkan Analisis Kebutuhan Sistem (**Teknik Pengumpulan Requirement**)

- a) Wawancara
- b) Observasi / Pengamatan
- c) Kuesioner / Angket

Cara Mendapatkan Analisis Kebutuhan

a) Wawancara

Adalah komunikasi tanya jawab secara langsung antara *client* dengan *system analyst* terkait alur bisnis yang sudah atau akan berjalan yg nantinya akan menjadi acuan pembuatan *software*.



Cara Mendapatkan Analisis Kebutuhan

b) Observasi / Pengamatan

Adalah pengamatan dan juga pencatatan sistematis terkait kondisi teknis/lapangan atas alur bisnis dan seluruh data/peraturan/*form* yang ada dalam sebuah organisasi yg akan diterapkan *software*.



Cara Mendapatkan Analisis Kebutuhan

c) Kuesioner / Angket

Merupakan teknik pengumpulan data yang dilakukan dengan cara memberi seperangkat pertanyaan atau pernyataan tertulis kepada responden untuk dijawabnya.



3) Kebutuhan Fungsional dan Non-Fungsional

- a) Kebutuhan Fungsional
- b) Kebutuhan Non-Fungsional

Kebutuhan Fungsional

- ❑ Menunjukkan *what the system should do*, fasilitas yg dibutuhkan, dan aktivitas yg akan diterapkan dalam sistem baru.
- ❑ **Kebutuhan fungsional** adalah kebutuhan yang berisi proses-proses apa saja yang nantinya dilakukan oleh sistem.
- ❑ Beberapa contoh kebutuhan fungsional: **)tidak selalu/sesuai case*
 - ❑ Fungsi deskripsi kebutuhan.
 - ❑ Laporan baik *hardcopy* maupun *softcopy*.
 - ❑ *Updating* dan *query online*.
 - ❑ Penyimpanan data, pencarian kembali dan transfer data.

Kebutuhan Non-Fungsional

- ❑ **Kebutuhan tambahan** yang mana bukan bersumber/berasal dari masalah inti.
- ❑ **Kebutuhan nonfungsional** adalah kebutuhan yang **menitikberatkan pada properti perilaku yang dimiliki oleh sistem**
- ❑ Beberapa contoh kebutuhan Non Fungsional: **)tidak selalu/sesuai case*
 - ~~❑ Waktu respon.~~
 - ~~❑ Rata-rata waktu untuk kegagalan.~~
 - ~~❑ Kebutuhan keamanan.~~
 - ~~❑ Akses untuk pengguna yang tidak punya hak.~~
 - ❑ Penataan tampilan/ *layout*.
 - ❑ Menggunakan SO Windows/Linux
 - ❑ Spesifikasi komputer Minimal
 - ❑ Printer
 - ❑ Sistem aplikasi dan database dilengkapi dengan password.
 - ❑ Informasi menampilkan tata cara pendaftaran anggota baru

4) Contoh Analisis Kebutuhan

Contoh 1 : Sistem Informasi Rawat Jalan Poliklinik ABC

a) Identifikasi Masalah



1. Keterbatasan tempat dan keamanan. **Data-data kertas pasien rawat jalan, rekam medis pasien**, serta **dokter** yang menangani tiap **pasien** mengalami peningkatan yang signifikan dan **rawan hilang/terbakar**.
2. Kesulitan pembatasan akses data. Misal, **rekap medis** perlu dijaga kerahasiannya, **pasien dan tenaga medis** yang menangani yang **diperbolehkan mengetahuinya**.
3. Kesulitan akses dan olah data. **Pengaksesan dan pengolahan data** yang **relatif banyak dan tersebar** menyebabkan **waktu dan tenaga ekstra**.

Contoh 1 : Sistem Informasi Rawat Jalan Poliklinik ABC

b) Kebutuhan Fungsional

- ☐ Login untuk pasien, dokter, dan petugas
- ☐ Pengelolaan data pasien, meliputi input, update dan delete.
- ☐ Pengelolaan data dokter, meliputi input, update dan delete.
- ☐ Pengelolaan data petugas, meliputi input, update dan delete.
- ☐ Pemeriksaan mulai pendaftaran (baru/lama), pemberian diagnosa, resep.

Contoh 1 : Sistem Informasi Rawat Jalan Poliklinik ABC

c) Kebutuhan Non-Fungsional

- ❑ Tampilan disusun

logo	Judul sistem	
menu	breadcumb	
	content	
footer		

- ❑ Dikembangkan berbasis *web* dengan *framework*.

Contoh 2 : Sistem Informasi Pemilihan Presentator

a) Identifikasi Masalah

Dalam **dunia pendidikan** khususnya dalam **kegiatan presentasi** ada beberapa masalah:

☐ Nilai *Inaccessible*

Nilai sementara hanya diakses oleh dosen, dan mestinya bisa diakses oleh mahasiswa.

☐ Waktu Melebihi Alokasi

Waktu presentasi melewati/melebihi alokasi waktu yang telah direncanakan.

☐ Subjektifitas

Anggapan bahwa mahasiswa memiliki kemampuan yang beragam perlu dihilangkan karena seluruh mahasiswa telah mendapatkan waktu pengerjakan, bimbingan, dan kepercayaan yang sama.

SI Pemilihan Presentator

konsep

- ❑ SI Pemilihan Presentator dikembangkan **berbasis web**.
- ❑ **Penerapan hak akses** terhadap penggunaan **fitur aplikasi**.
- ❑ **Memuat fitur-fitur fungsional** dalam **pelaksanaan pemilihan secara random terkomputerisasi** untuk **mahasiswa** yang akan **maju presentasi**.
- ❑ **Menerapkan ketertiban & kedisiplinan proses penilaian** serta **menghilangkan subjektifitas** dalam pemilihan urutan presentator.

Fitur

SI Pemilihan Presentator

fitur - keseluruhan

- ❑ Adalah **sebuah fasilitas fungsional / fasilitas** tulang punggung dalam penerapan **Pemilihan Presentator** di matakuliah ADSI yang diampu Doni Abdul Fatah.
- ❑ Diantara **fiturnya:**
 - 1) Login
 - 2) Halaman Utama
 - 3) Mahasiswa
 - 4) Kelas
 - 5) Tugas Besar
 - 6) Pengundian dan Penilaian Presentasi
 - 7) Logout

SI Pemilihan Presentator

fitur – 1. Login

- ☐ Berisi halaman untuk **login** untuk masuk ke SI Pemilihan Presentator.
- ☐ Ketentuan autentifikasi yang benar adalah kesesuaian antara **id dan password**.
- ☐ Id dan password yang sesuai akan **dibentuk session** sebagai **pengunci login**.
- ☐ Yang **boleh login** adalah **admin dan mahasiswa**.

SI Pemilihan Presentator

fitur – 2. Halaman Utama

- ☐ Berisi **halaman sapaan** dan **landing page** dari **login**.
- ☐ Di pengembangan selanjutnya, halaman ini memungkinkan **berwujud dashboard**.

SI Pemilihan Presentator

fitur – 3. Mahasiswa

- ☐ **Melihat (Read/R)** data mahasiswa yang berada dalam sistem.
- ☐ **Membuat (Create/C)** data mahasiswa baru.
- ☐ **Mengedit (Update/U)** data mahasiswa saat ada pembaharuan atau kesalahan data.
- ☐ **Menonaktifkan (Soft Delete/SD)** data mahasiswa yang telah tidak digunakan.

SI Pemilihan Presentator

fitur – 4. Kelas

- ☐ **Melihat (Read/R)** data kelas yang berada dalam sistem.
- ☐ **Membuat (Create/C)** data kelas baru.
- ☐ **Mengedit (Update/U)** data kelas saat ada pembaharuan atau kesalahan data.
- ☐ **Menonaktifkan (Soft Delete/SD)** data kelas yang telah tidak digunakan.
- ☐ **Memasukkan (Enter/E)** mahasiswa ke kelas.
- ☐ **Menghapus (Remove/Rmv)** mahasiswa dari kelas.

SI Pemilihan Presentator

fitur – 5. Tugas Besar

- ☐ **Melihat (Read/R)** data tugas besar yang berada dalam sistem.
- ☐ **Membuat (Create/C)** data tugas besar baru.
- ☐ **Mengedit (Update/U)** data tugas besar saat ada pembaharuan atau kesalahan data.
- ☐ **Menonaktifkan (Soft Delete/SD)** data tugas besar yang telah tidak digunakan.
- ☐ **Memasukkan (Enter/E)** mahasiswa ke tugas besar.
- ☐ **Menghapus (Remove/Rmv)** mahasiswa dari tugas besar.

SI Pemilihan Presentator

fitur – 6. Pengundian dan Penilaian Presentasi

- ☐ **Melihat (Read/R)** penilaian mahasiswa yang telah melakukan presentasi.
- ☐ **Melakukan pengundian (Create/C)** mahasiswa yang belum melakukan presentasi berdasarkan kelas mahasiswa.
- ☐ **Memberikan penilaian (Enter/E)** terhadap mahasiswa yang terpilih maju presentasi.



SI Pemilihan Presentator

fitur – 7. Logout

- ❑ Menghapus session (Remove/Rmv) session.

Hak Akses

SI Pemilihan Presentator

hak akses - keseluruhan

- ❑ Adalah **jenis akun atau akses** yang digunakan dalam operasional kegiatan pemilihan presentator dengan **hak/ fungsinya** masing-masing.
- ❑ **Hak akses** yang ada dalam SI Pemilihan Presentator:
 - 1) Admin (ADM)
 - 2) Mahasiswa (MHS)

SI Pemilihan Presentator

hak akses - icon



ADMIN



MAHASISWA

SI Pemilihan Presentator

hak akses – tentang admin



- ☐ Dijabat oleh Doni Abdul Fatah selaku dosen pengampu MK Analisa & Desain Sistem Informasi .
- ☐ Memiliki **akses penuh** terhadap SI Pemilihan Presentator.



SI Pemilihan Presentator

hak akses – tentang mahasiswa



- ☐ Adalah **akses** yang menjadi **objek pengolahan data** dalam proses **pemilihan presentator** tugas besar ADSI.
- ☐ **Mahasiswa** dalam sistem ini adalah **ADSI Kelas A, B, C dan D semester 2 Sistem Informasi**.
- ☐ **Akses mahasiswa** terbatas membaca data yang ada dalam sistem.

hak akses – rincian akses admin dan mahasiswa

MENU/FITUR		HAK AKSES											
		ADMIN 					MAHASISWA 						
		DONI ABDUL FATAH					ADSI KELAS A,B, C, D Semester 2 Sistem Informasi						
		C	R	U	SD	E	Rmv	C	R	U	SD	E	Rmv
1	Login												
2.1	Halaman Utama												
2.2	Mahasiswa												
2.3	Kelas												
2.4	Tugas Besar												
2.5	Pengundian dan Penilaian Presentasi												
2.6	Logout												

Konsep SDLC PROTOTYPING

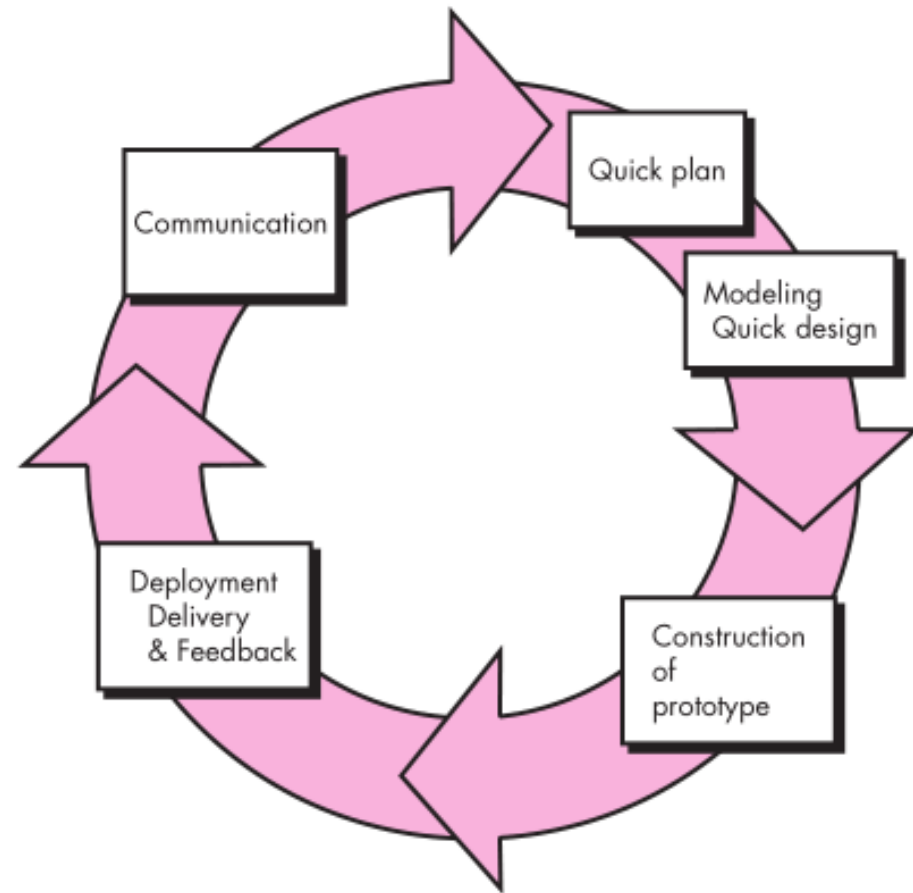
SI Pemilihan Presentator

SDLC SI Pemilihan Presentator

konsep SDLC prototyping

Alasan memilih SDLC prototipe merujuk pernyataan Roger S Pressman dalam Software Engineering A Practitioner's Approach (2010:43):

A quick design focuses on a representation of those aspects of the software that will be visible to end users (e.g., human interface layout or output display format)





SDLC SI Pemilihan Presentator

penerapan SDLC prototyping

1. Analisis
2. Desain/ gambar aplikasi jadi
3. Implementasi/ mewujudkan aplikasi

Desain/ Gambar Aplikasi

SDLC SI Pemilihan Presentator

alur – keseluruhan

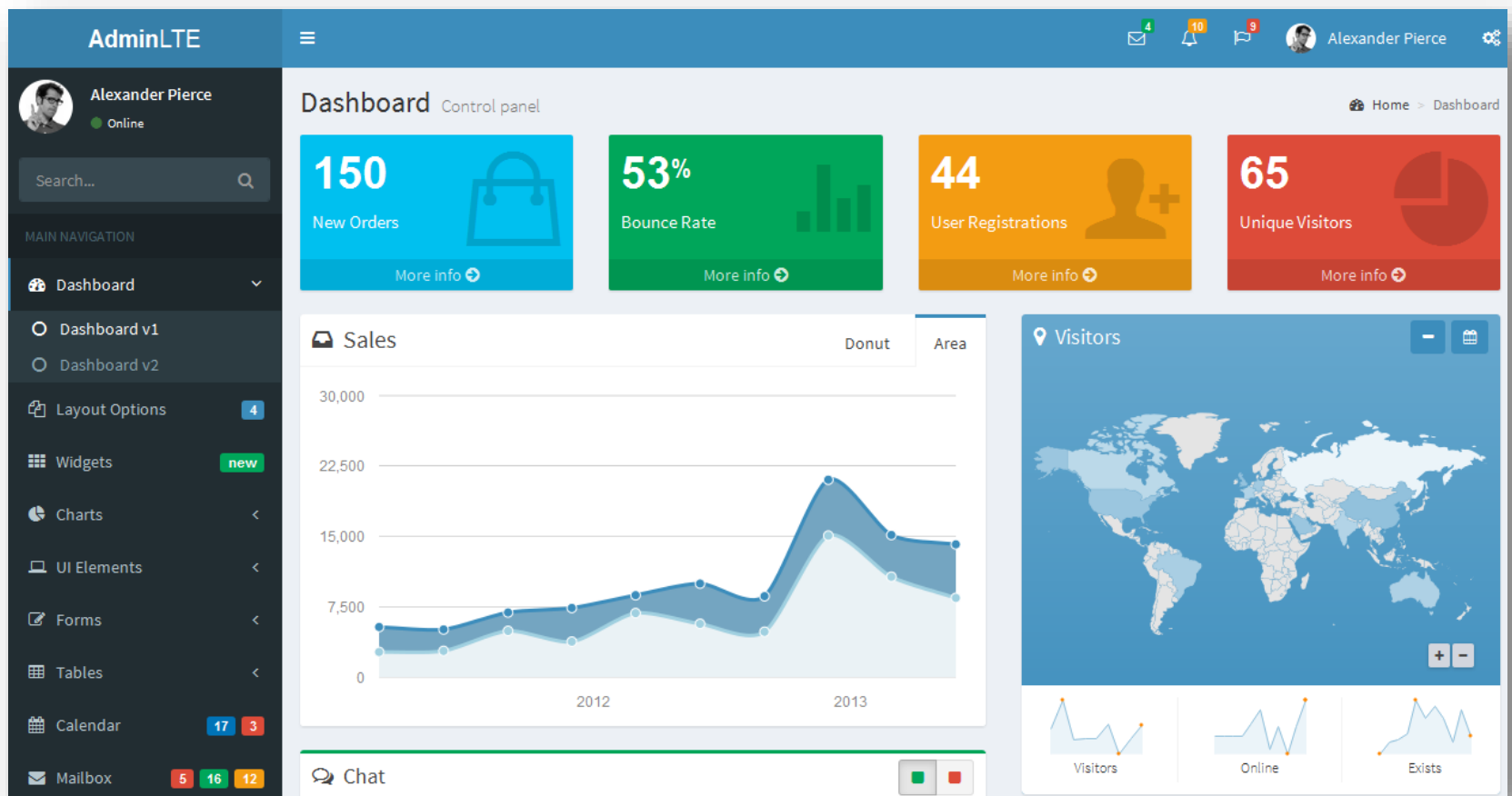
❑ Alur pada umumnya dalam mendesain adalah:

1. Pilih template.
2. Desain aplikasi berdasar pada template dan fitur yang ada (*usually method*).

SDLC SI Pemilihan Presenter

alur – 1. pilih template

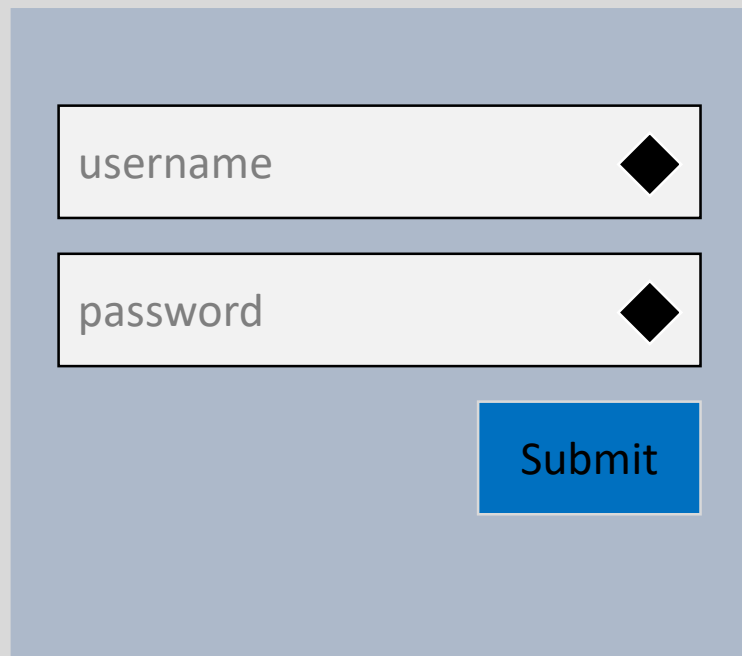
Saya memilih template adminLTE.



SDLC SI Pemilihan Presentator

alur – 2.a Desain tampilan login

Sistem Informasi Pemilihan Presentator (SIPeTor)



A login form for the SIPeTor system. It features a light blue background. At the top, there is a white rectangular box containing the text 'username' and a black diamond icon. Below this is another white rectangular box containing the text 'password' and a black diamond icon. To the right of the password box is a blue rectangular button with the text 'Submit' in white.

SDLC SI Pemilihan Presentator

alur – 2.b Desain tampilan utama

Sistem Informasi Pemilihan Presentator

AKUN



Akun

Dashboard Judul

SIPeTor > Dashboard

Dashboard

Mahasiswa

Kelas

Tugas Besar

Pengundian-Penilaian

Berisi session dan informasi rekapitulasi sistem secara keseluruhan.

SDLC SI Pemilihan Presentator

alur – 2.c Desain tampilan mahasiswa

Sistem Informasi Pemilihan Presentator

AKUN



Akun

Dashboard Judul

SIPeTor > Dashboard

Dashboard
Mahasiswa
Kelas
Tugas Besar
Pengundian-Penilaian

SDLC SI Pemilihan Presentator

alur – 2.d Desain tampilan kelas

Sistem Informasi Pemilihan Presentator

AKUN



Akun

Dashboard
Mahasiswa
Kelas
Tugas Besar
Pengundian-Penilaian

Dashboard Judul

SIPeTor > Dashboard

Berisi data kelas dan data lengkapnya, termasuk mengelola mahasiswa dalam kelas tersebut.

SDLC SI Pemilihan Presentator

alur – 2.e Desain tampilan tugas besar

Sistem Informasi Pemilihan Presentator

AKUN



Akun

Dashboard
Mahasiswa
Kelas
Tugas Besar
Pengundian-Penilaian

Dashboard Judul

SIPeTor > Dashboard

Berisi data tugas besarnya dan data lengkapnya, termasuk mengelola mahasiswa dalam tugas besar tersebut.

SDLC SI Pemilihan Presentator

alur – 2.f Desain tampln pengundian dan penilaian

Sistem Informasi Pemilihan Presentator

AKUN



Akun

Dashboard
Mahasiswa
Kelas
Tugas Besar
Pengundian-Penilaian

Dashboard Judul

SIPeTor > Dashboard

Tempat untuk mengelola pengundian dan penilaian serta bertampilan dasar mahasiswa yang telah presentasi.

Tugas

- ☐ Buatlah Analisis Kebutuhan Fungsional dan Non-Fungsional sesuai dengan project yang akan dibangun
- ☐ Tentukan SDLC yang digunakan kemudian Desain/gambar aplikasi jadinya

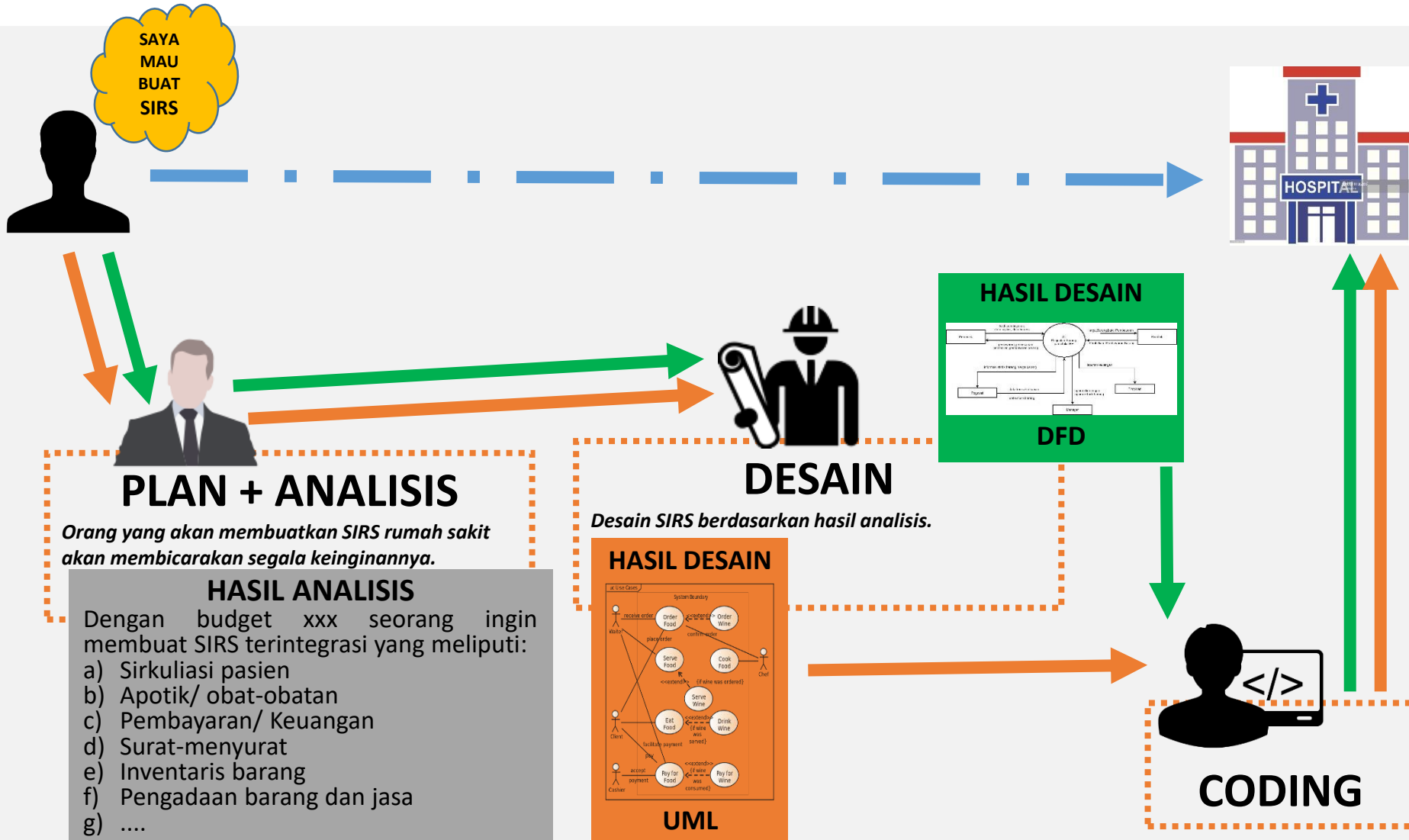
Pemodelan

Pemodelan

- 1) Pengantar
- 2) Pemodelan Proses
- 3) Pemodelan Data
- 4) Pemodelan Proses dan Data
- 5) Pemodelan Konsep Prosedural
- 6) Pemodelan Konsep OOP

1) Pengantar

Posisi Pemodelan



Pemodelan

- ❑ Proses modelling adalah cara formal untuk menggambarkan bagaimana bisnis beroperasi.
- ❑ Mengilustrasikan aktivitas-aktivitas yang dilakukan dan bagaimana data berpindah diantara aktifitas-aktifitas itu.
- ❑ Ada banyak cara untuk merepresentasikan proses model, cara yang populer adalah dengan menggunakan data flow diagram (DFD) dan UML (Unified Modeling Language)



Sasaran

- ❑ Menggambarkan apa yang dibutuhkan oleh pelanggan.
- ❑ Membangun dasar bagi pembuatan desain perangkat lunak.
- ❑ Membatasi serangkaian persyaratan yang dapat divalidasi begitu perangkat lunak dibangun.
- ❑ Membantu Estimasi sumber daya.
- ❑ Menvalidasi software requirement dari berbagai sudut pandang sehingga kesalahan dapat ditemukan segera

Tekniknya

- ❑ Pemrograman prosedural atau struktural (DFD)
 - Melibatkan data dan proses yang mengubah data. Ex: DFD
- ❑ Pemrograman berorientasi objek /OOP (UML)

Pemodelan Data - ERD

- ❑ Memungkinkan untuk identifikasi obyek data dan hubungannya dengan menggunakan notasi grafis
- ❑ Menetapkan semua data yang dimasukkan, disimpan, ditransformasikan dan diproduksi pada suatu aplikasi
- ❑ Hanya berfokus pada data



Komponen ERD

- ❑ Entitas (*entity*)
- ❑ Relasi (*relationship*)
- ❑ Atribut (*attribute*)
- ❑ Kardinalitas (*kardinality*)
- ❑ Modalitas (*modality*)



Entitas

❑ Definisi

❑ Sebuah obyek yang dapat dibedakan dari obyek lain

❑ Contoh

- Individu : pegawai, pelanggan, mahasiswa, distributor
- Tempat : kampus, kantor, lapangan
- Obyek : buku, motor, paket software
- Peristiwa : pendaftaran, pemesanan, penagihan
- Konsep : rekening, kualifikasi

Contoh Entitas

Bangunan



Produk



Pelanggan





Relasi

Relasi

■ Definisi

- ◆ Asosiasi 2 atau lebih entitas

■ Berupa kata kerja

Mahasiswa

Mengambil

Mata Kuliah

Atribut

Address

■ Definisi

- ◆ Properti yang dimiliki setiap entitas yang akan disimpan datanya.

■ Contoh

◆ Atribut Pelanggan

- ◆ No KTP/SIM
- ◆ Nama
- ◆ Alamat



Kardinalitas Relasi

■ Definisi

- ◆ Angka yang menunjukkan banyaknya kemunculan suatu obyek terkait dengan kemunculan obyek lain pada suatu relasi
- ◆ Kombinasi yang mungkin : (1:1, 1:N, M:N)

Contoh



- 1 Departemen mungkin mempekerjakan 1 atau lebih pegawai
- 1 Pegawai hanya bekerja pada sebuah departemen

Modalitas Relasi

■ Definisi

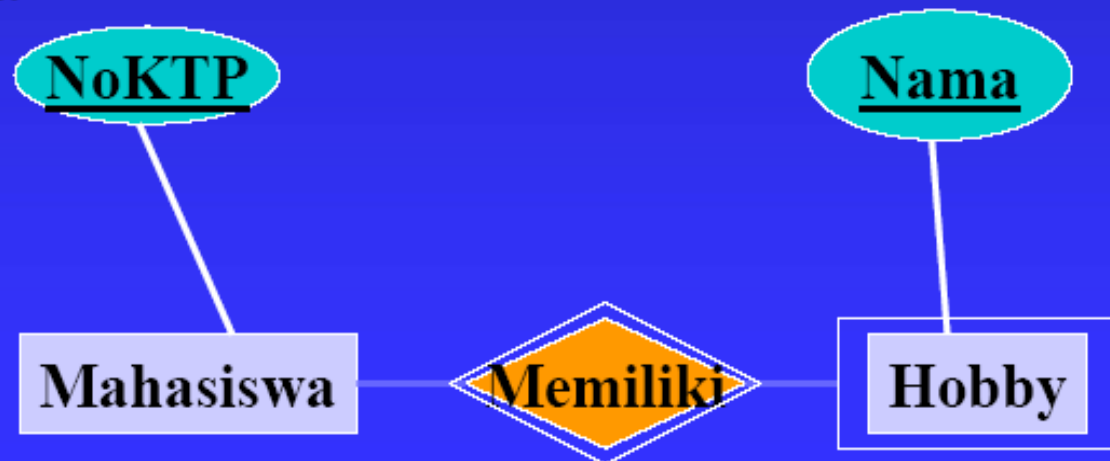
- ◆ Partisipasi sebuah entitas pada suatu relasi
- ◆ 0 jika partisipasi bersifat “optional”/parsial
- ◆ 1 jika partisipasi bersifat “wajib”/total

■ Contoh

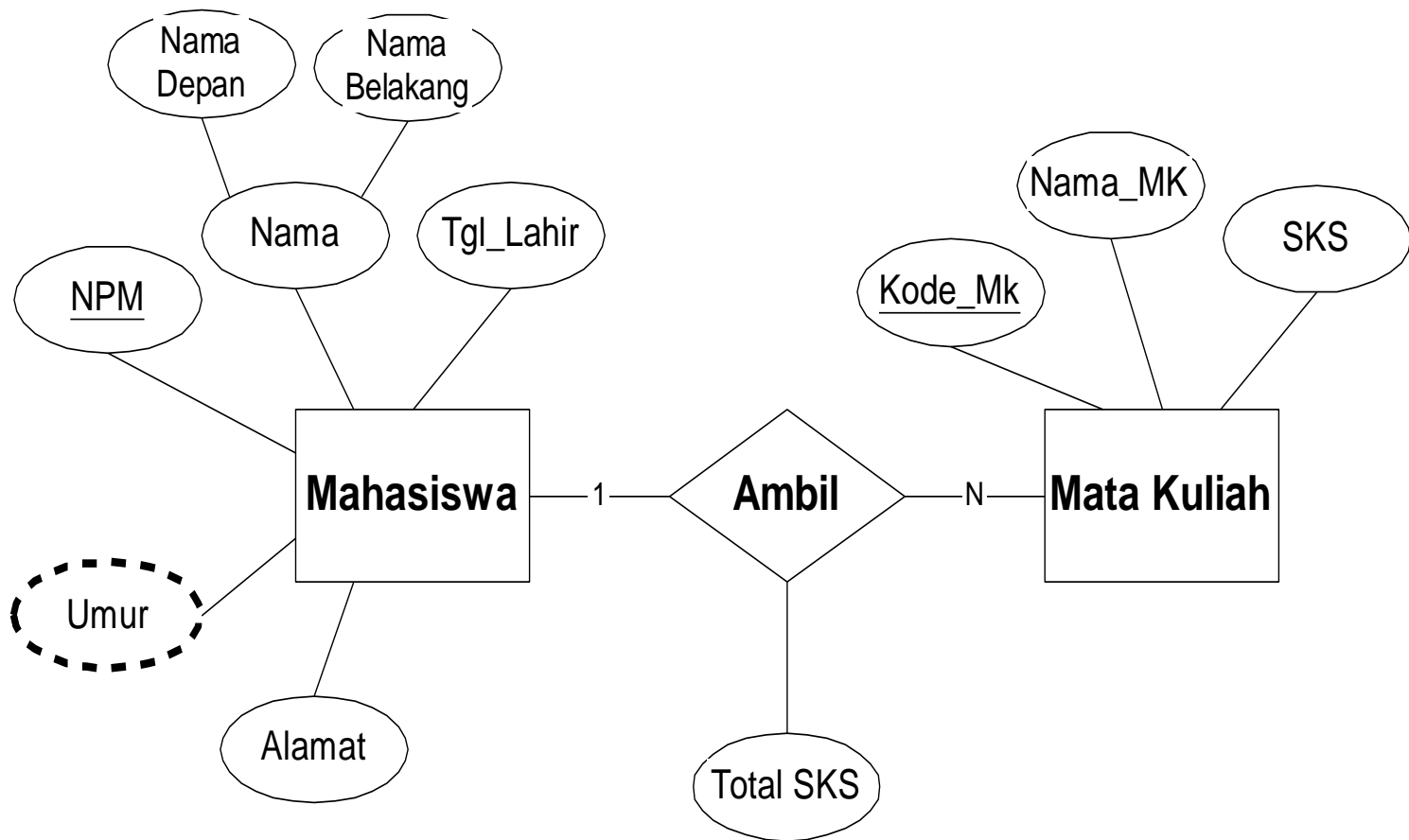
- ◆ Partisipasi total
 - ◆ Setiap anak memiliki ibu
- ◆ Partisipasi parsial
 - ◆ Tidak setiap perempuan memiliki anak

Entitas Lemah/Kuat

- Entitas Kuat : Entitas yang memiliki atribut kunci (*Key*)
- Entitas Lemah : Entitas yang biasanya berasal dari atribut multivalue pada entitas lain.



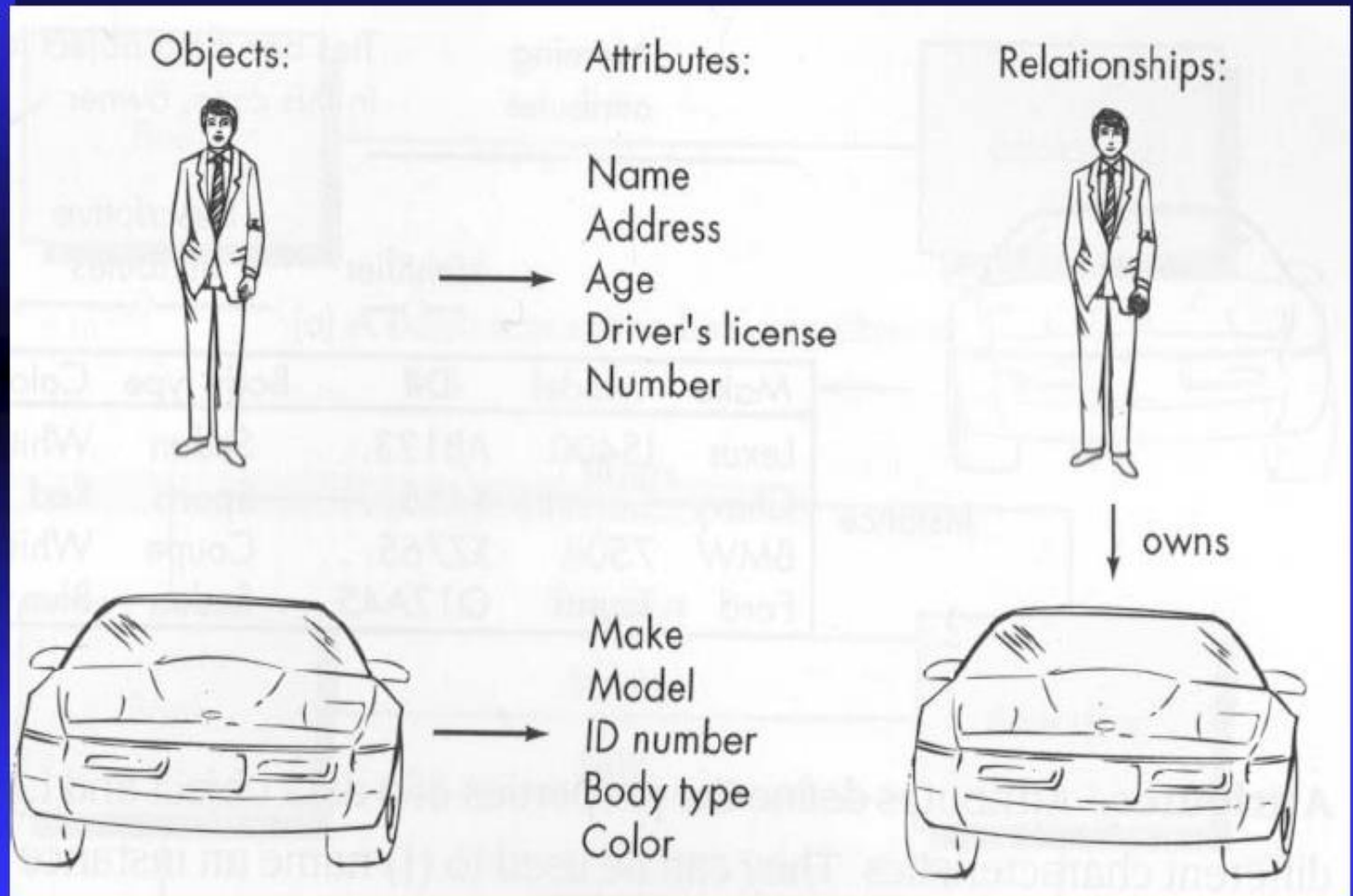
Contoh ERD



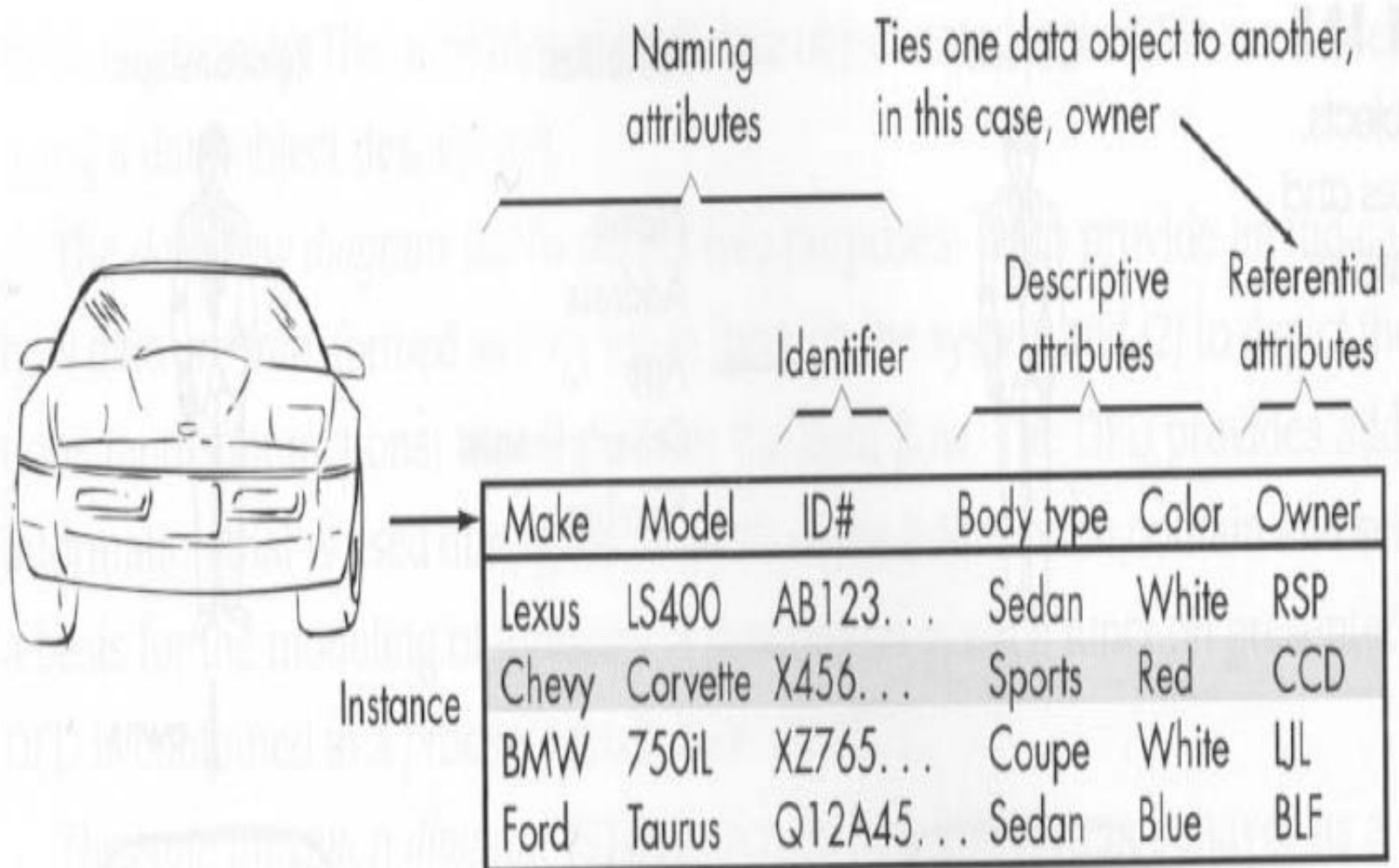
Pemodelan Data yang Baik

- Sederhana
- Tidak ada duplikasi data
(*redundant*)
- Fleksibel dan mudah beradaptasi
dengan perkembangan

Contoh Pemodelan Data

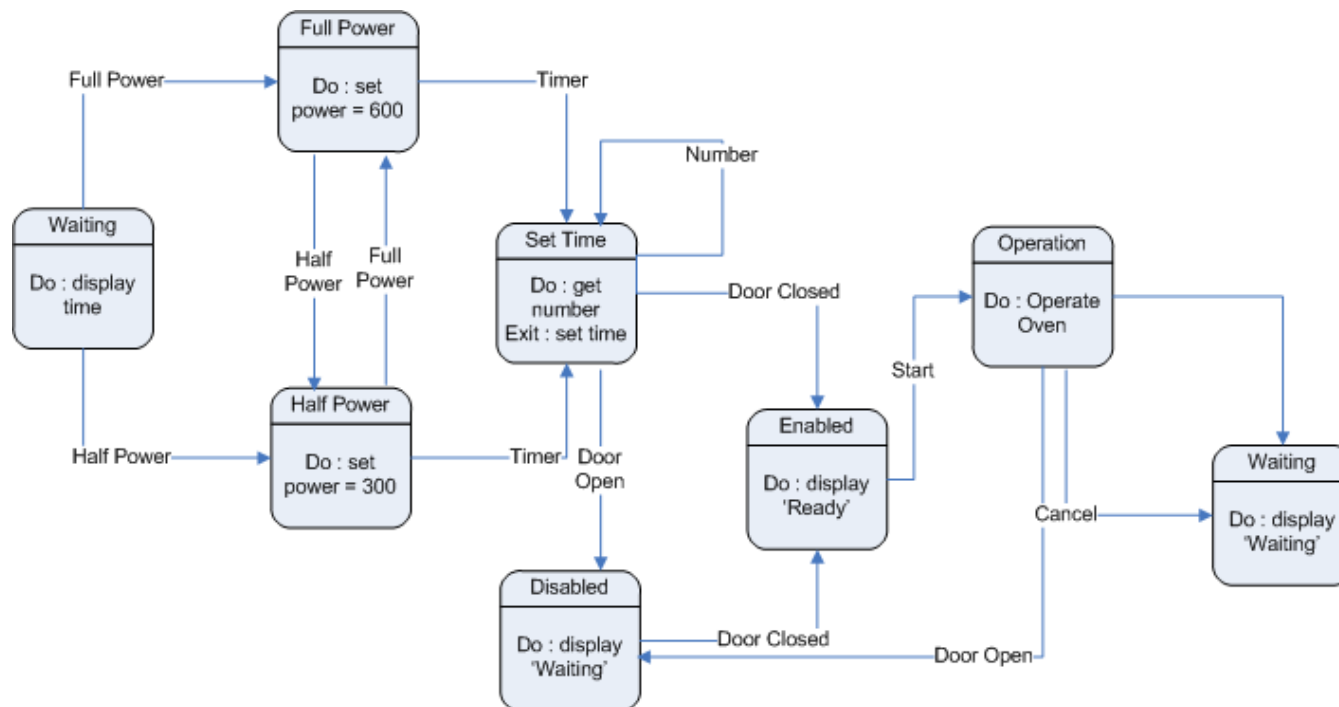


Representasi Tabular



Pemodelan Berbasis Perilaku

- ❑ Pemodelan ini lebih mengarah pada perilaku dari sistem atau produk.
- ❑ Menggambarkan bagaimana sistem atau perangkat lunak akan merespon jika ada event dari luar.



Pemodelan Konsep Prosedural

DFD - Data Flow Diagram

- ❑ Sebuah teknik grafis yang menggambarkan aliran informasi dan transformasi yang diaplikasikan pada saat data bergerak dari input menjadi output
- ❑ Pemodelan ini mendefinisikan bagaimana obyek – obyek data ditransformasikan oleh fungsi proses.
- ❑ Data Flow Diagram (DFD) adalah representasi grafis sebuah sistem aplikasi berorientasi proses.
- ❑ Komponen-komponen sebuah DFD adalah: proses, aliran data, data store, dan entitas

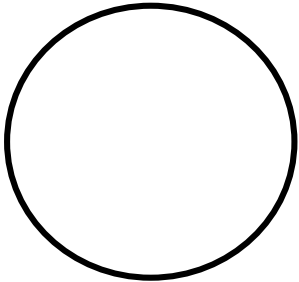
Data Flow Diagram

- ❑ Penggunaan DFD dipopulerkan oleh DeMarco – Yordan dan Gane – Sarson dengan menggunakan pendekatan Metoda Analisis Sistem Terstruktur (SSADM).

Notasi D F D



❑ Entity eksternal : prosedur atau konsumer inf. yang ada di luar bound sistem



❑ Proses : transfer informasi (fungsi) yang ada dalam bound sistem




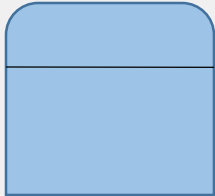


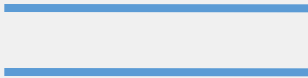
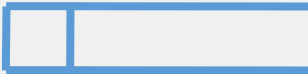


❑ Obyek data, menunjukkan aliran data



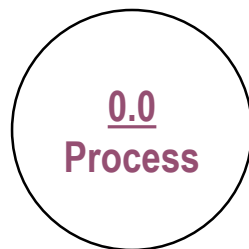
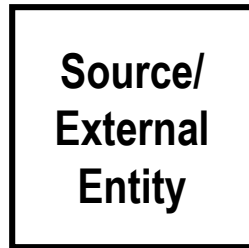
❑ Penyimpanan data

DFD Symbol

DeMarco – Yordan Symbols	Keterangan	Gane – Sarson Symbols
	External Entity (Kesatuan Luar)	
	Process (Proses)	
	Data Flow (Aliran Data)	
	Data Store (Penyimpanan Data)	

Data Flow Diagrams Symbols

DeMarco & Yourdon



Source/External Entity –

- Entitas (kesatuan) diluar sistem yang akan dimodelkan.
- Memberikan input atau menerima output dari/ke sistem.
- Berupa orang, organisasi, sumber informasi lain atau penerima akhir suatu laporan

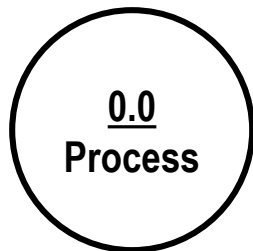
Mahasiswa

Yayasan

Data Flow Diagrams Symbols

DeMarco & Yourdon

Source/
External
Entity

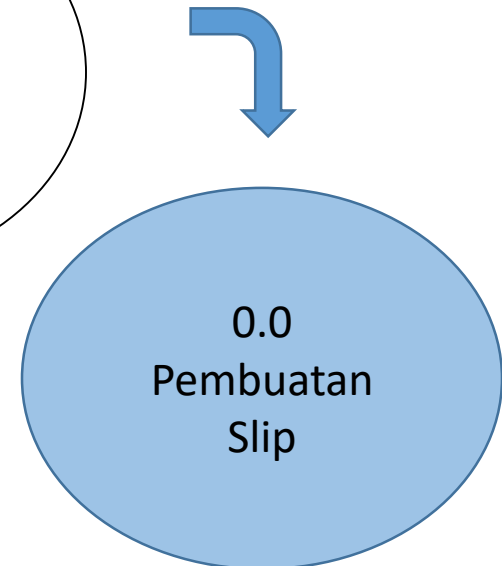
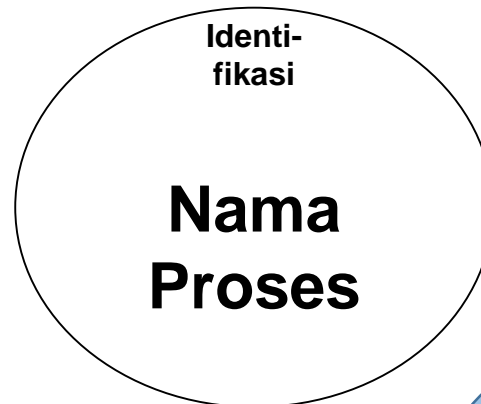


DATA STORE



Data Flow Lines

Processes – Merupakan pekerjaan atau kegiatan yang dilakukan orang atau komputer, dimana aliran data masuk, ditransformasikan ke aliran data keluar



Data Flow Diagrams Symbols

DeMarco & Yourdon

Source/
External
Entity

0.0
Process

DATA STORE

Data Flow Lines

Data Store – Dapat berupa suatu file atau suatu sistem database dari suatu komputer, suatu arsip/dokumen, suatu agenda/buku


CONTOH :

PELANGGAN

MS_BARANG

Data Flow Diagrams Symbols

DeMarco & Yourdon



Source/
External
Entity



0.0
Process



DATA STORE



Data Flow Lines

Data Flow – is the data in motion. Data can move from the outside (source) into a process. Once the inside of a system data must flow from place to place through a process, the flow lines show this movement.

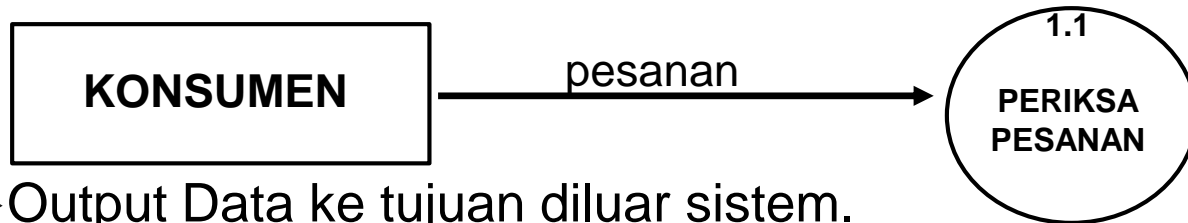
- Menggambarkan aliran data dari satu proses ke proses lain
- Menggunakan anak panah

Komponen DFD

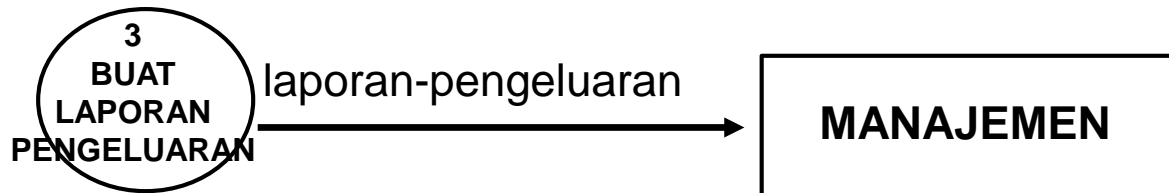
❑ ALIRAN DATA (DATA FLOW)

❖ Interface Aliran Data dengan Terminator.

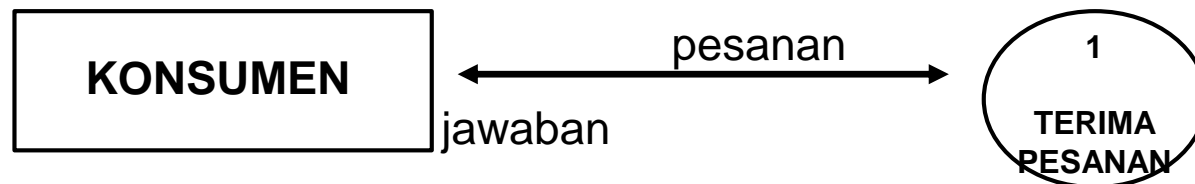
➤ Input Data ke sistem dari sumber diluar sistem.



➤ Output Data ke tujuan diluar sistem.



➤ Interaktif Dialog antara sistem sumber atau tujuan.



Komponen DFD

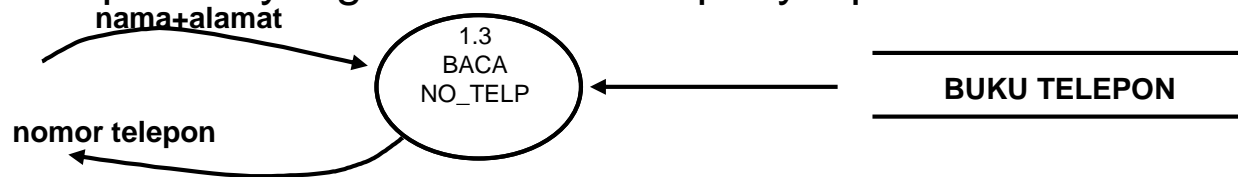
❑ ALIRAN DATA (DATA FLOW)

❖ Interface terhadap penyimpanan data.

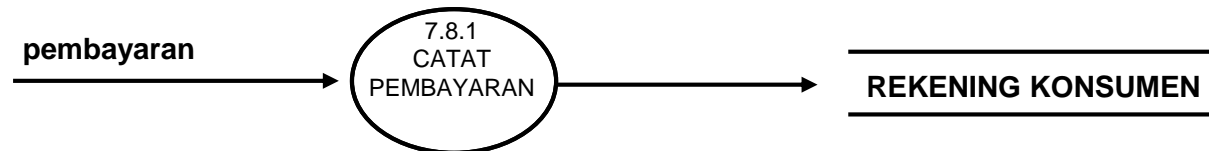
- Suatu proses yang **write data** ke penyimpanan data.



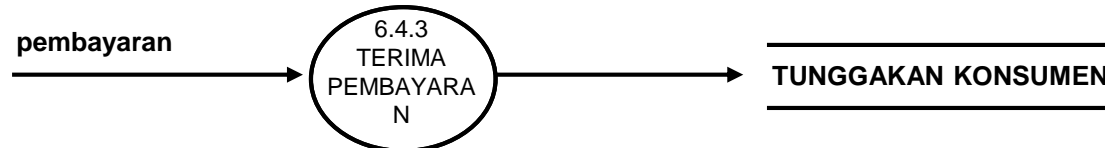
- Suatu proses yang **read data** dari penyimpanan data.



- Suatu proses yang **update data** didalam penyimpanan data.



- Suatu proses yang **delete data** didalam penyimpanan data.

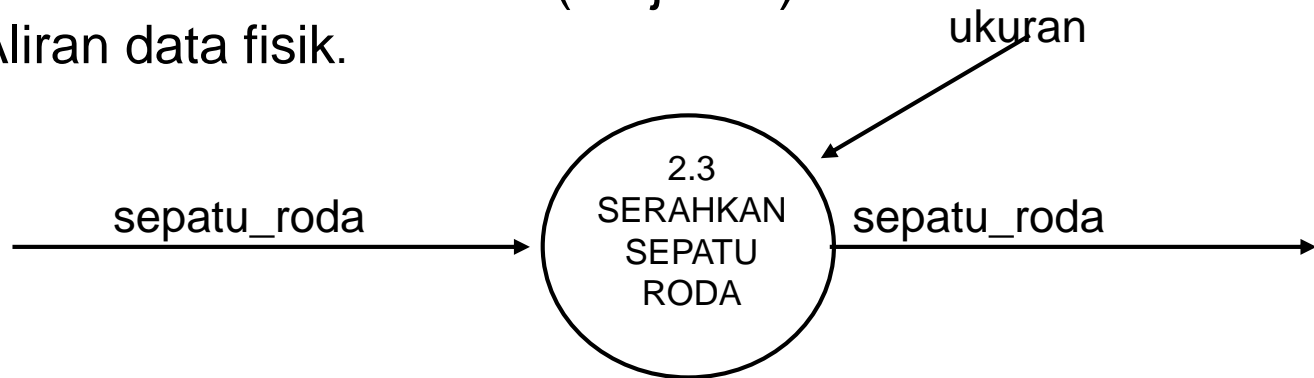


Komponen DFD

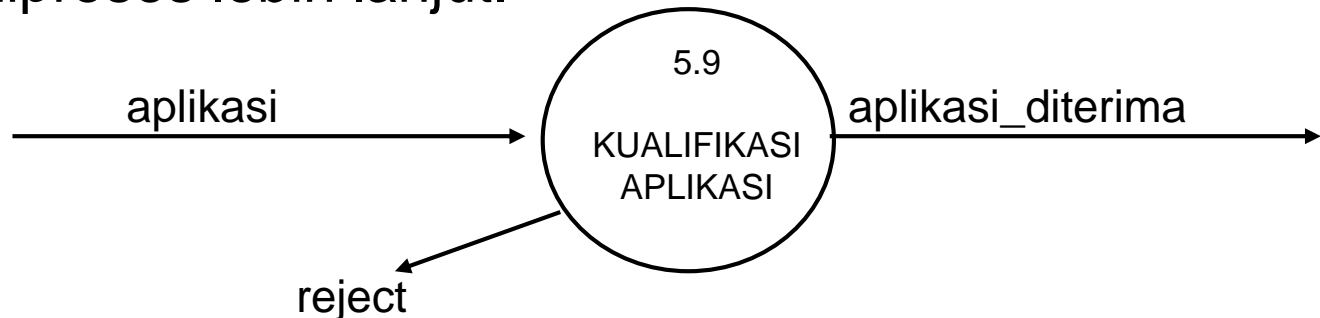
ⁿ ALIRAN DATA (DATA FLOW)

❖ Interface didalam sistem (lanjutan).

➤ Aliran data fisik.



➤ Aliran data yang ditolak (reject) oleh sistem dan tidak diproses lebih lanjut.



Design DFD

- ❑ DFD (Data Flow Diagram)
- ❑ Cara kerja adalah melakukan dekomposisi sampai level 4.

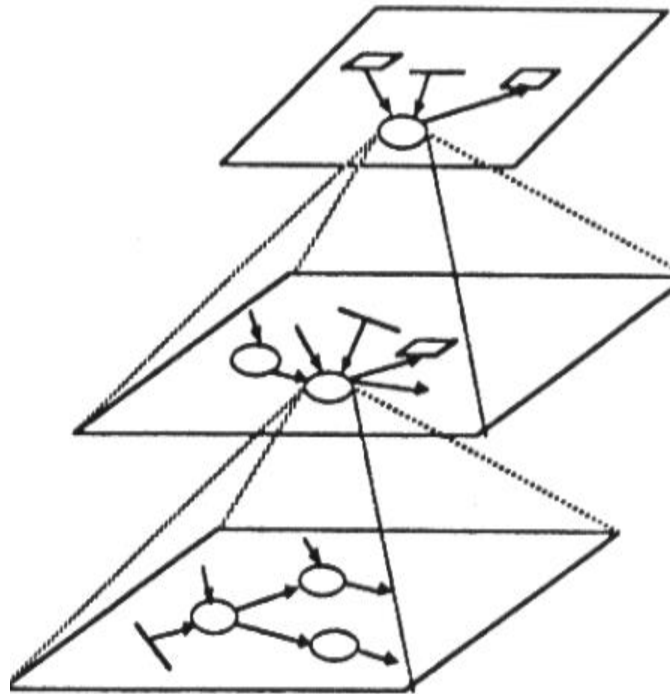
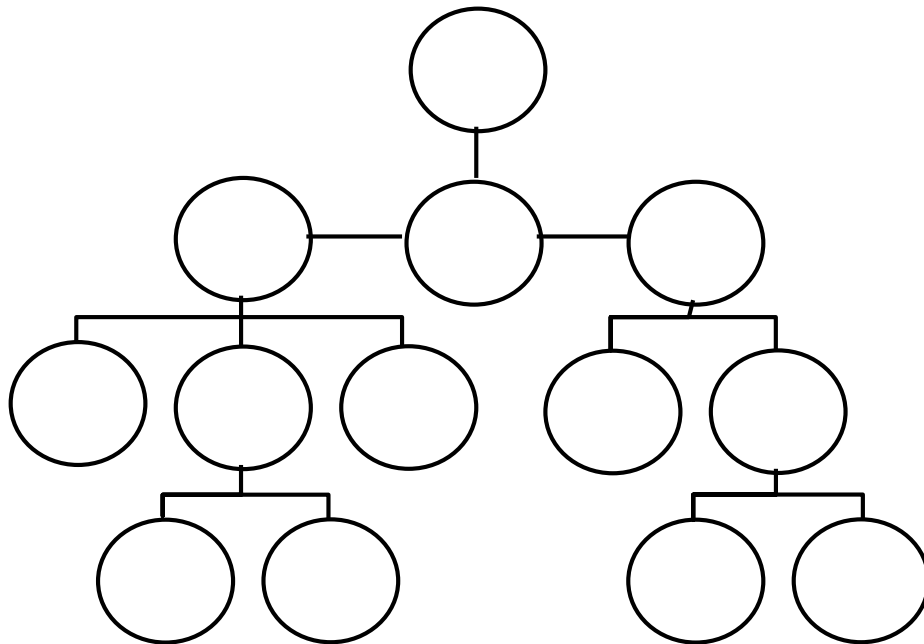


Diagram Dekomposisi

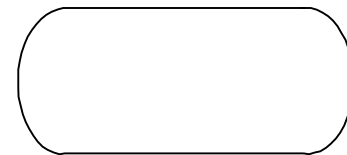
□ DEKOMPOSISI PROSES

Bentuk Diagram

Bisa menggunakan notasi lain,
seperti:



atau



atau



Diagram Dekomposisi

□ HUBUNGAN DIAGRAM DEKOMPIISI DENGAN DFD.

Diagram Dekomposisi

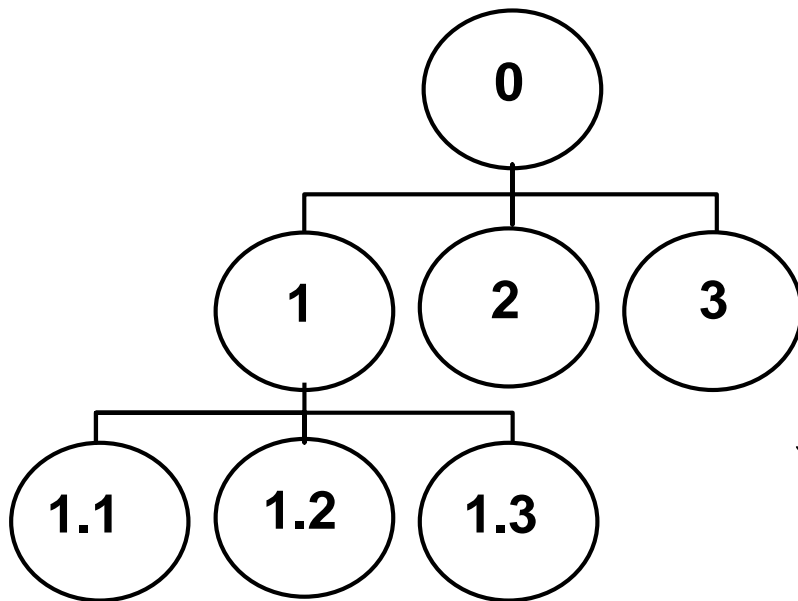


Diagram Aliran Data

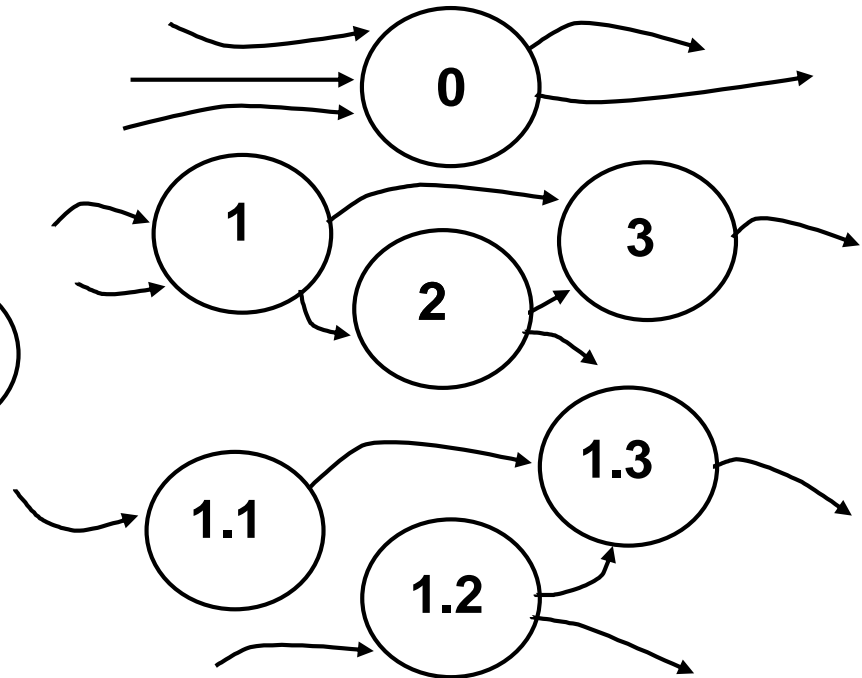
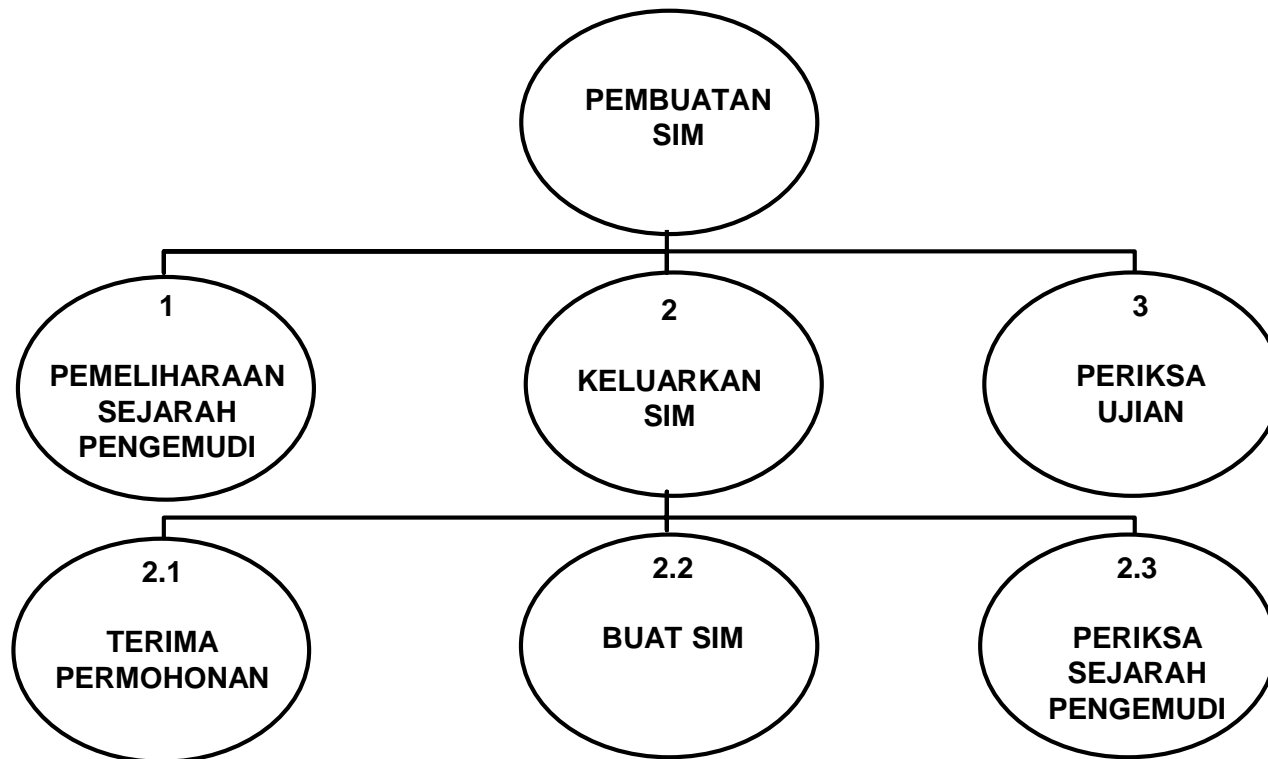


Diagram Dekomposisi

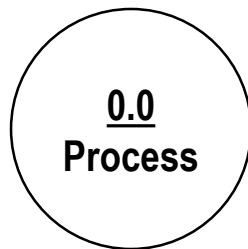
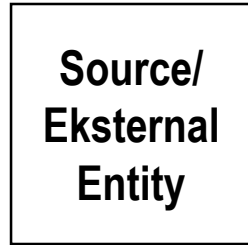
□ Contoh Diagram Dekomposisi

- Dari contoh DFD bertingkat Pembuatan SIM



Data Flow Diagrams Levels

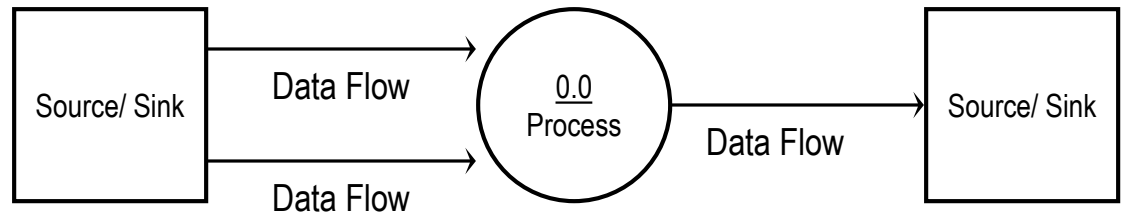
DeMarco & Yourdon



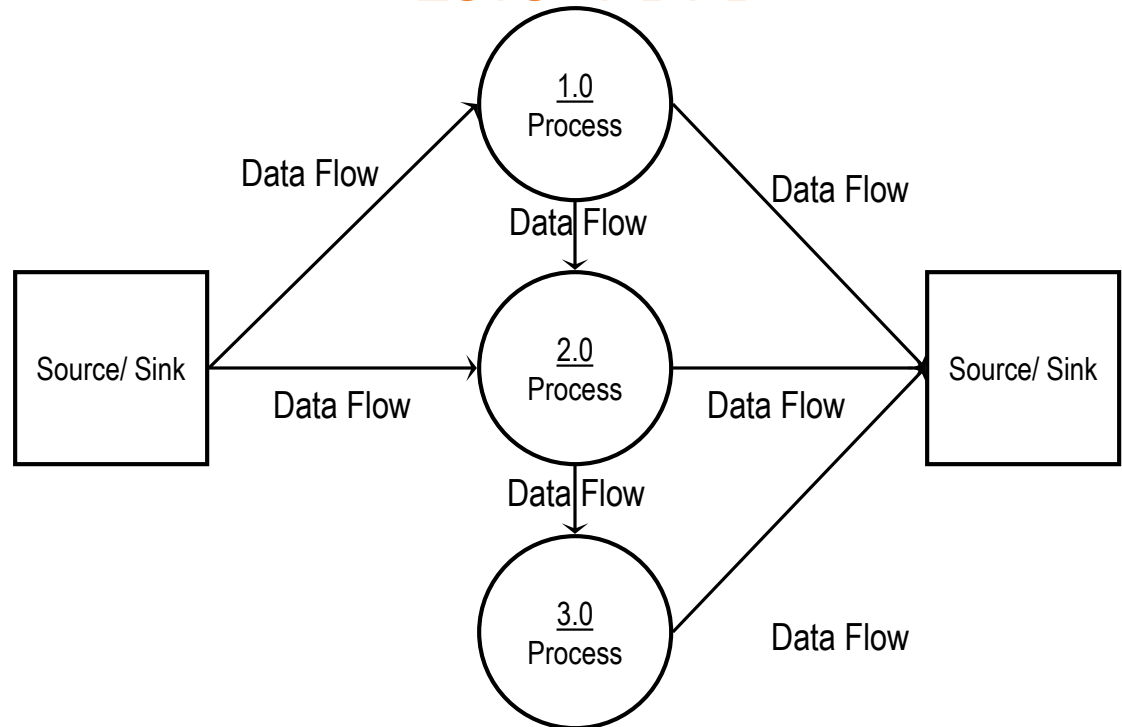
DATA STORE



Context Level DFD



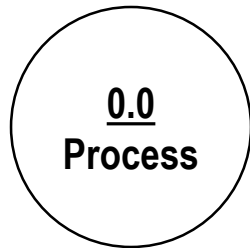
Level 1 DFD



Data Flow Diagrams Levels

DeMarco & Yourdon

**Source/
External
Entity**



DATA STORE



Level 2 DFD (and on)

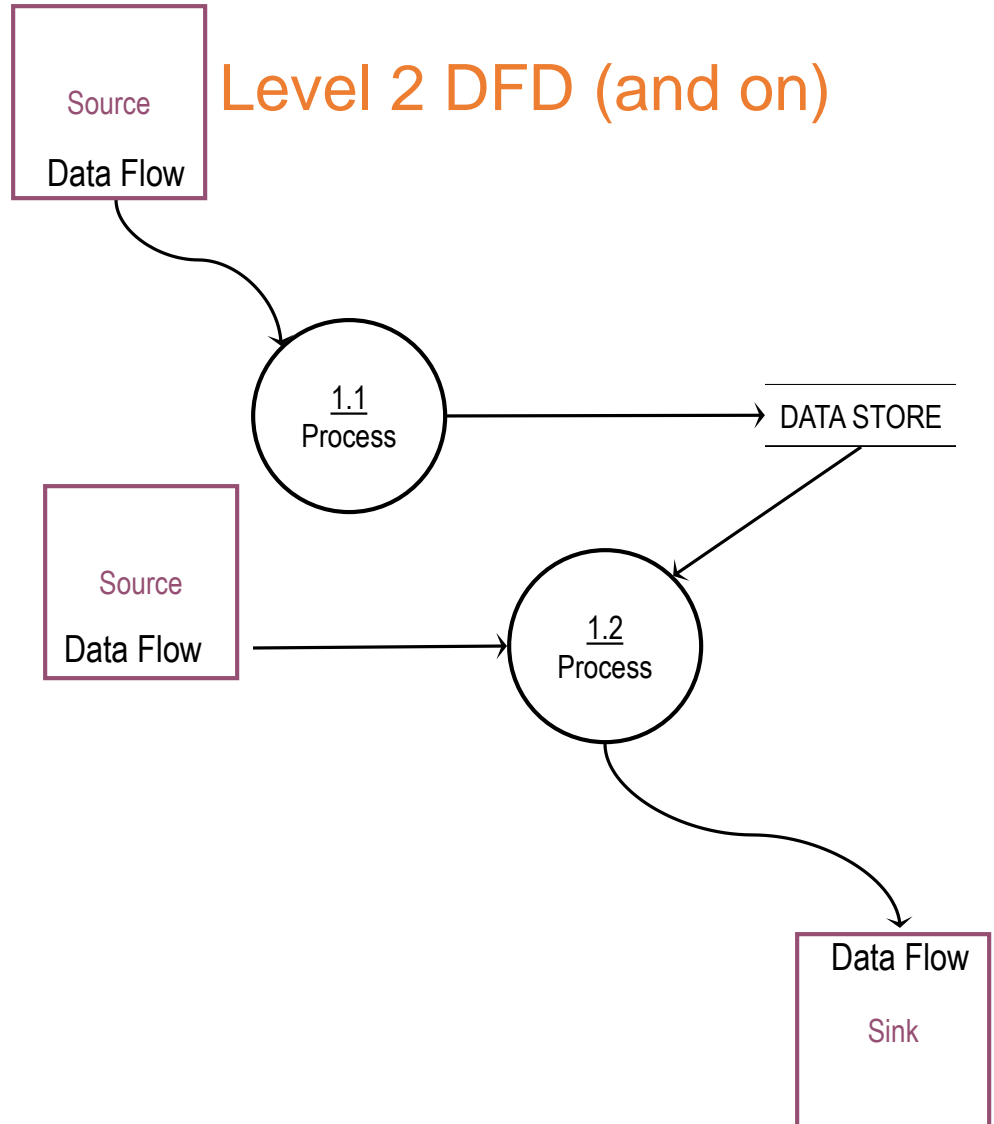


Diagram Aliran Data Bertingkat

□PENYUSUNAN DFD BERTINGKAT

❖ Penomoran

➤ Contoh diagram konteks.

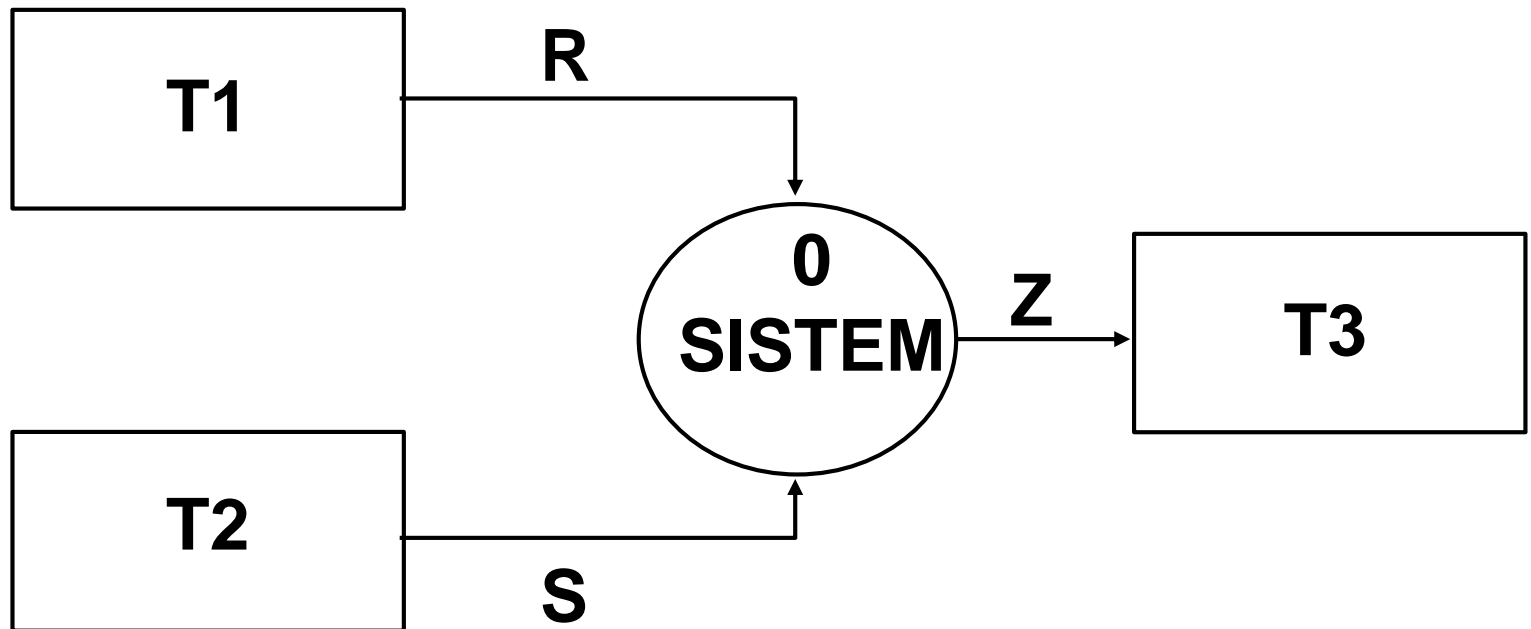


Diagram Aliran Data Bertingkat

□ PENYUSUNAN DAD BERTINGKAT

❖ Penomoran

- Nomor Diagram "**ANAK**" harus diawali dengan nomor proses pada diagram "**ORANG TUA**" yang terkait.

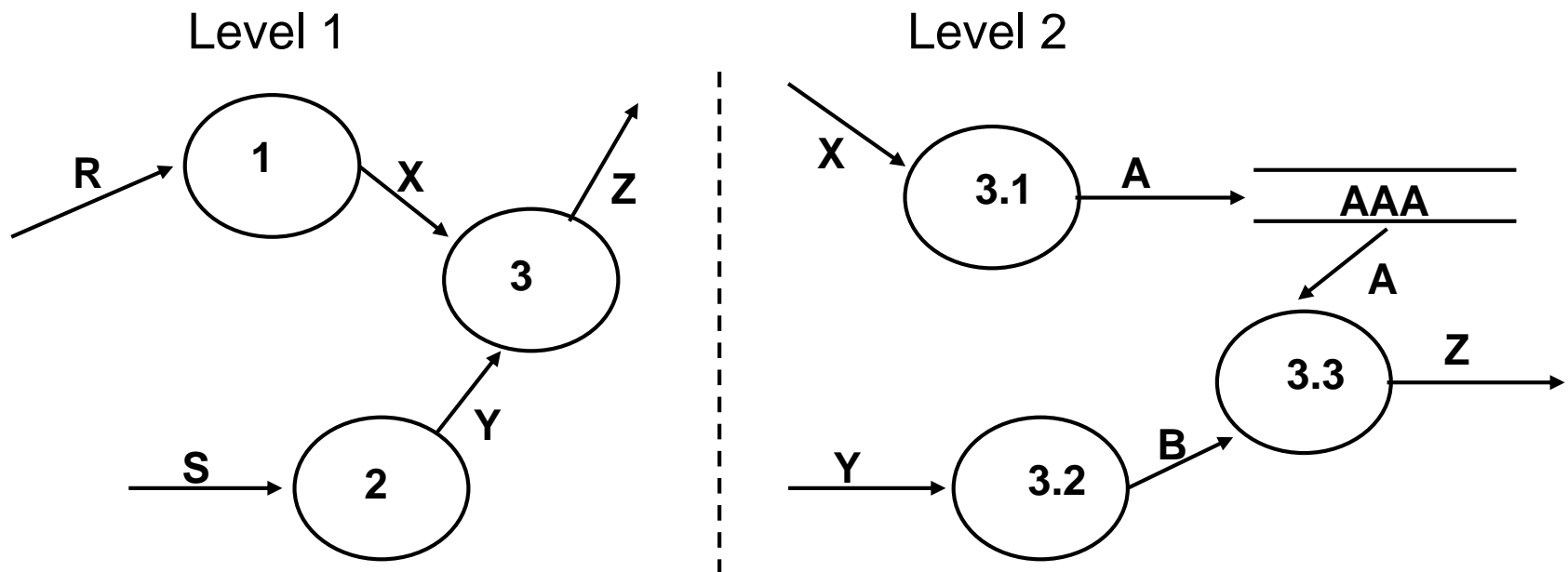


Diagram Aliran Data Bertingkat

❖PENYUSUNAN DFD BERTINGKAT

❖ Penomoran

- Dengan menyebutkan nomor diagram "**ANAK**" yang sesuai dengan nomor proses pada diagram "**ORANG TUA**" yang terkait.

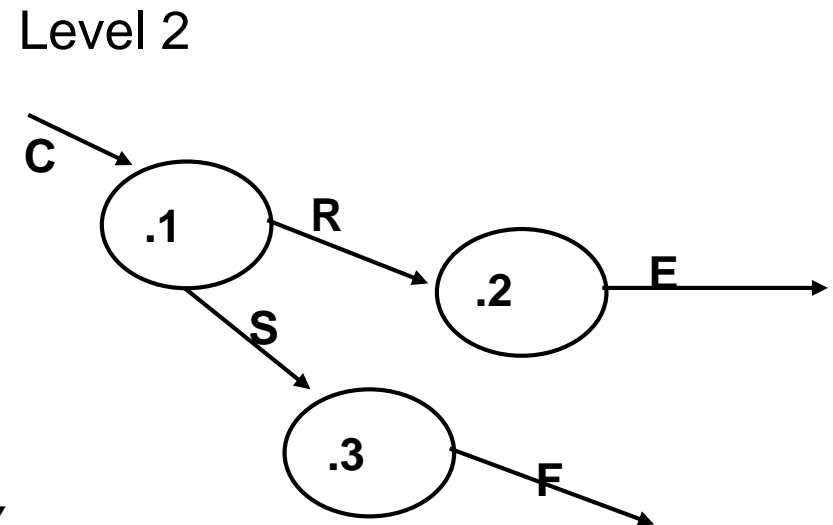
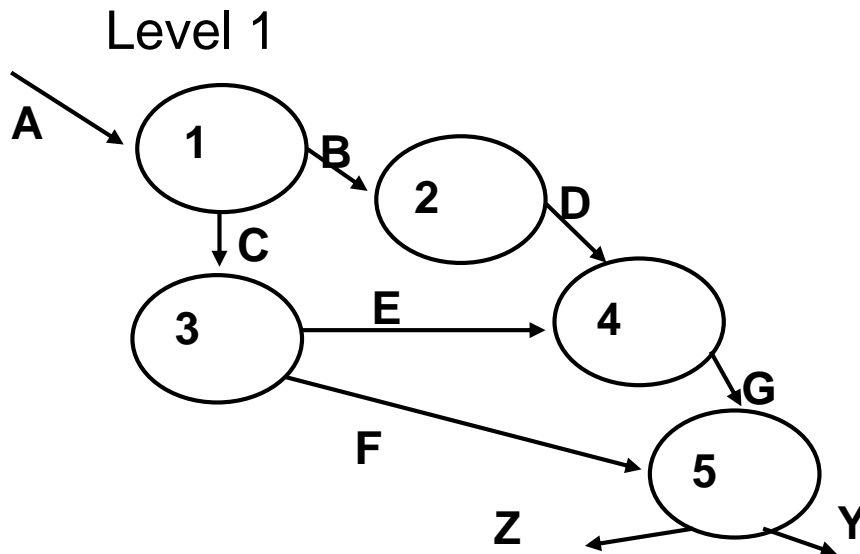


Diagram Aliran Data Bertingkat

□PENYUSUNAN DFD BERTINGKAT

❖ Aturan keseimbangan.

➤ Semua aliran data masuk dan keluar diagram "**ORANG TUA**" harus ada/sama pada diagram "**ANAK**".

Diagram "**ORANG TUA**"

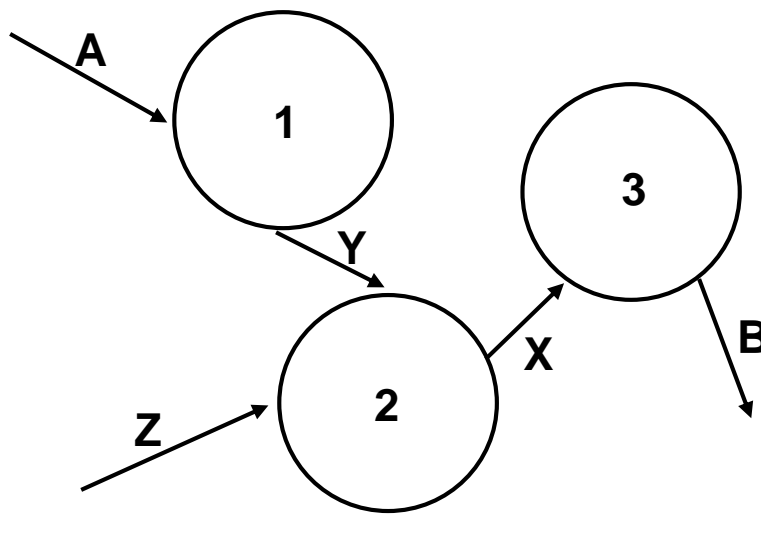
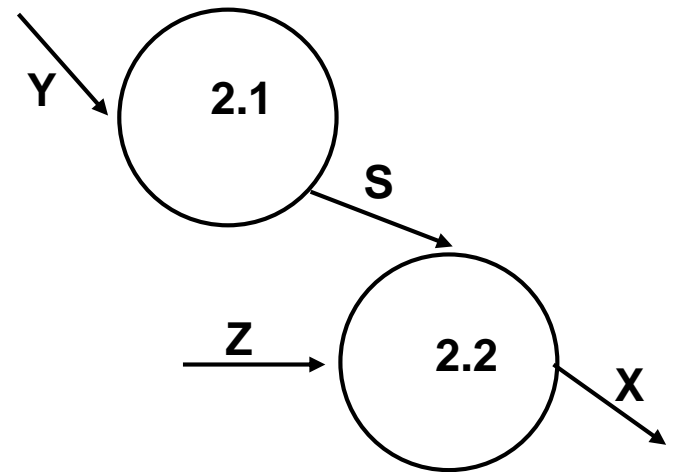


Diagram "**ANAK**"

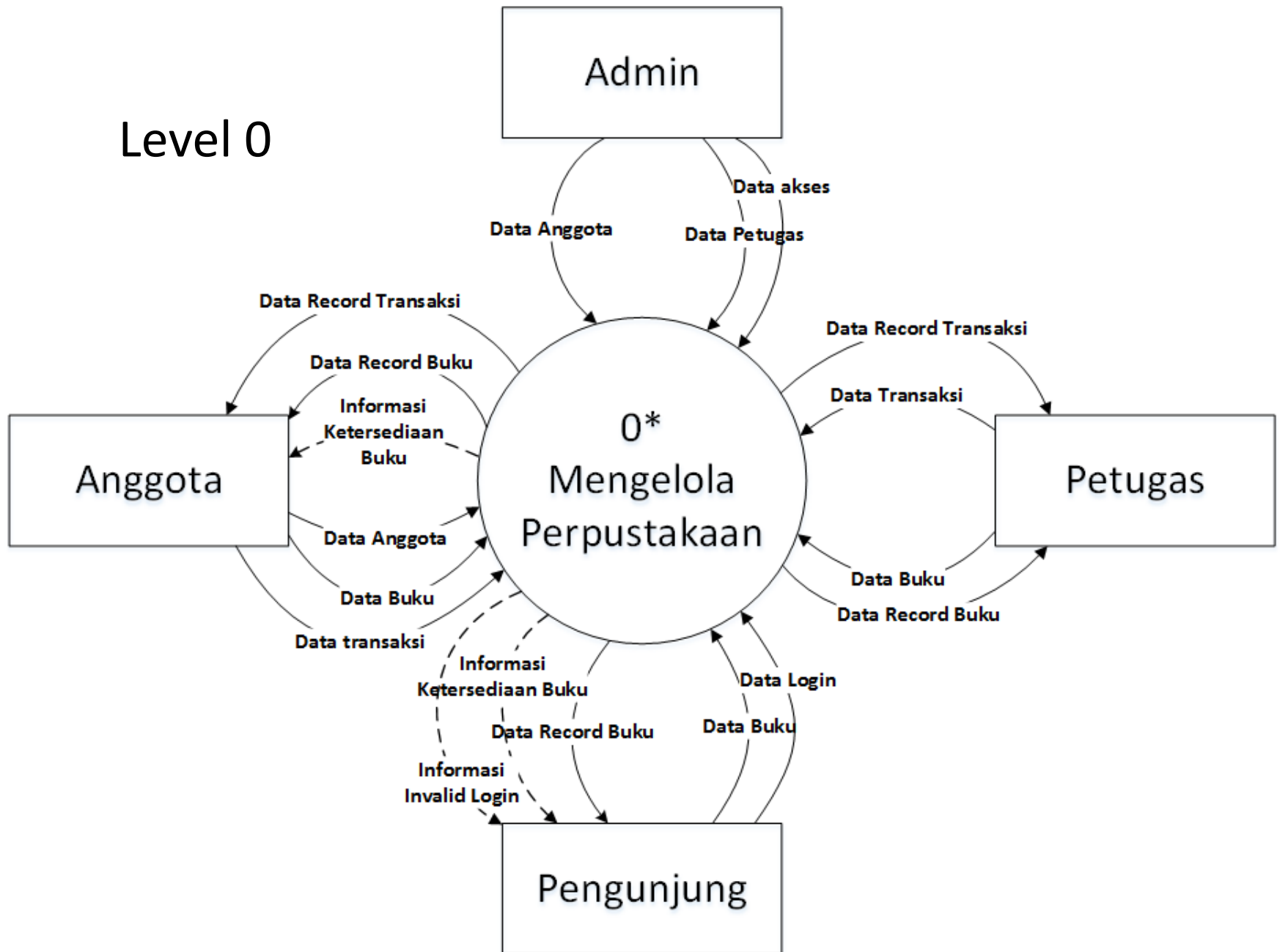




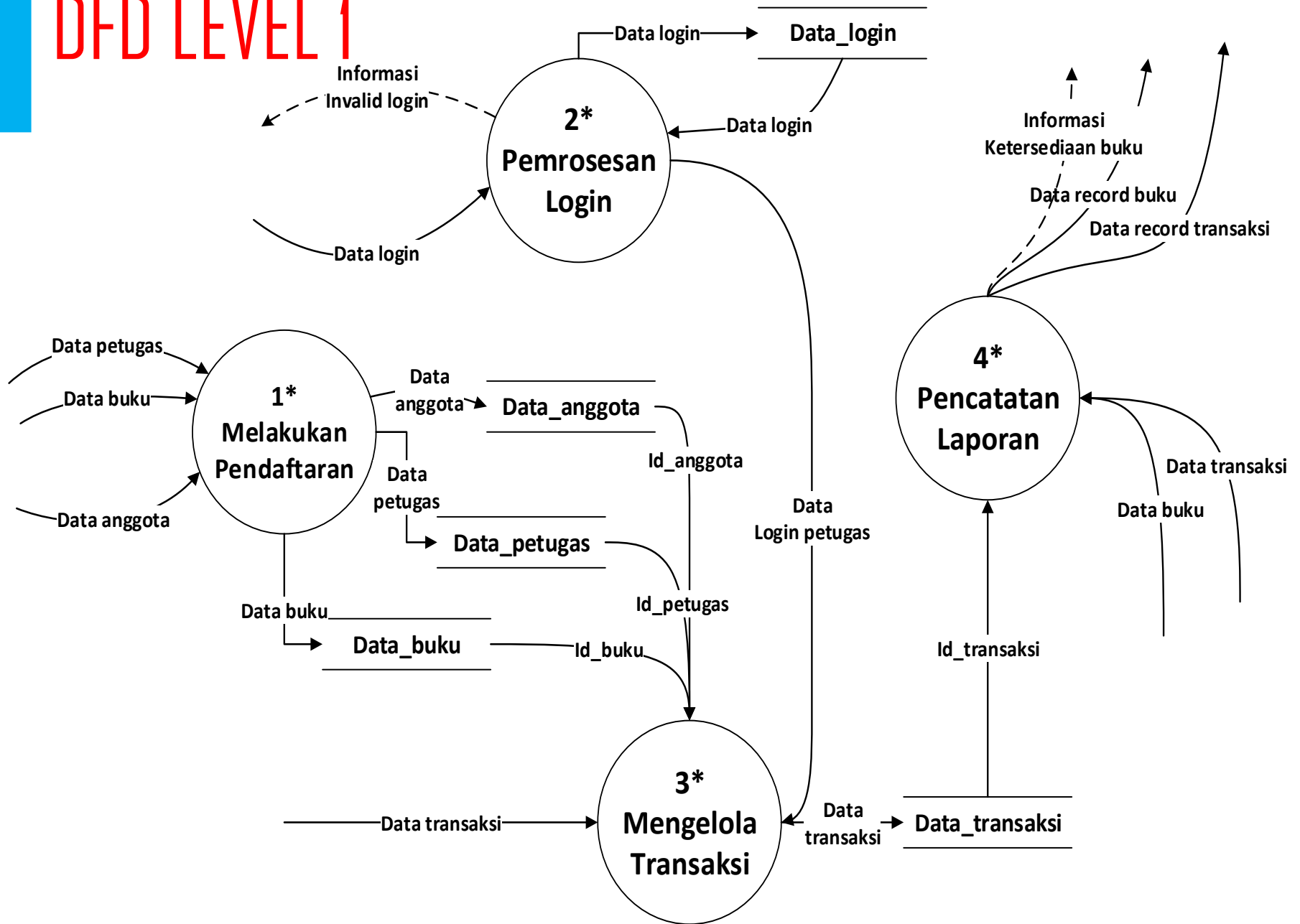
Creating Data Flow Diagrams

1. Buat daftar aktifitas
2. Bangun Context Level DFD
(Identifikasi External Entity)
3. Bangun Level 1 DFD
(mengidentifikasi sub proses yang dapat dikelola)
4. Construct Level 2- n DFD
(identifikasi arus data aktual dan penyimpanan data)

Level 0

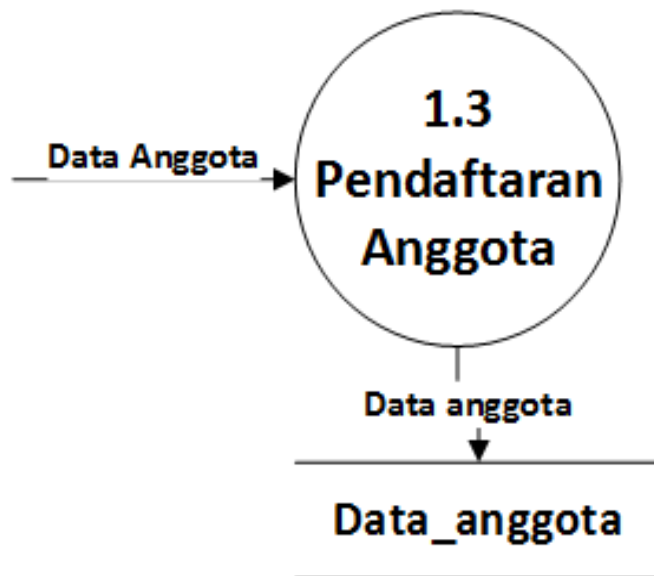
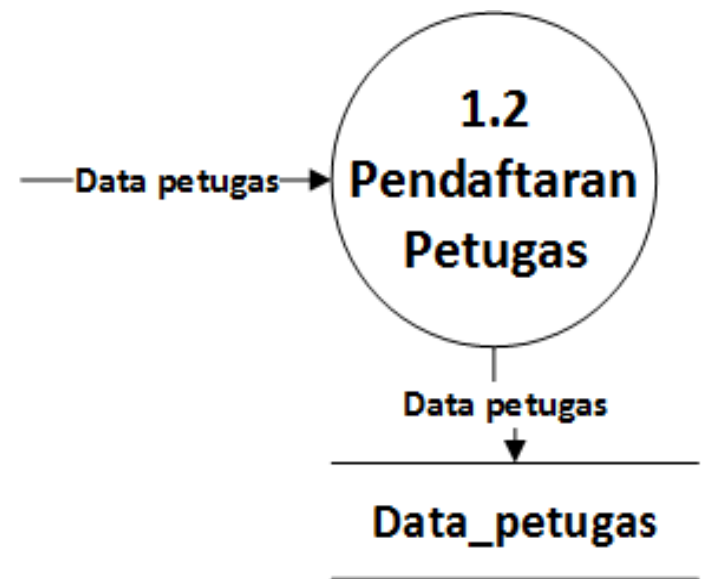
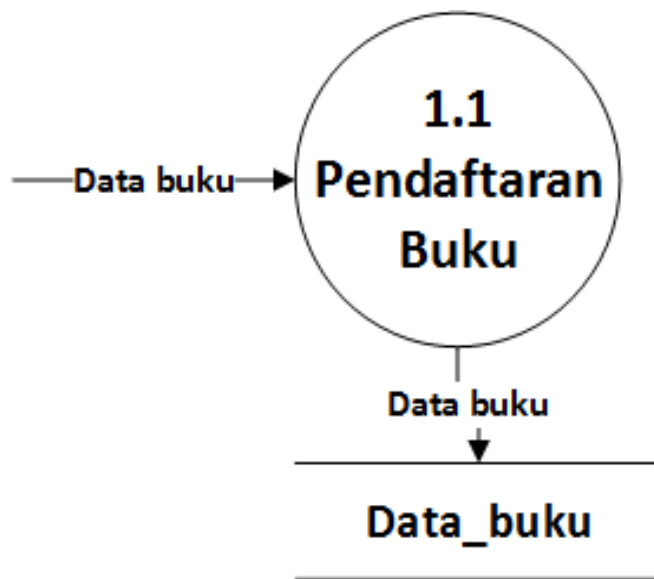


DFD LEVEL 1

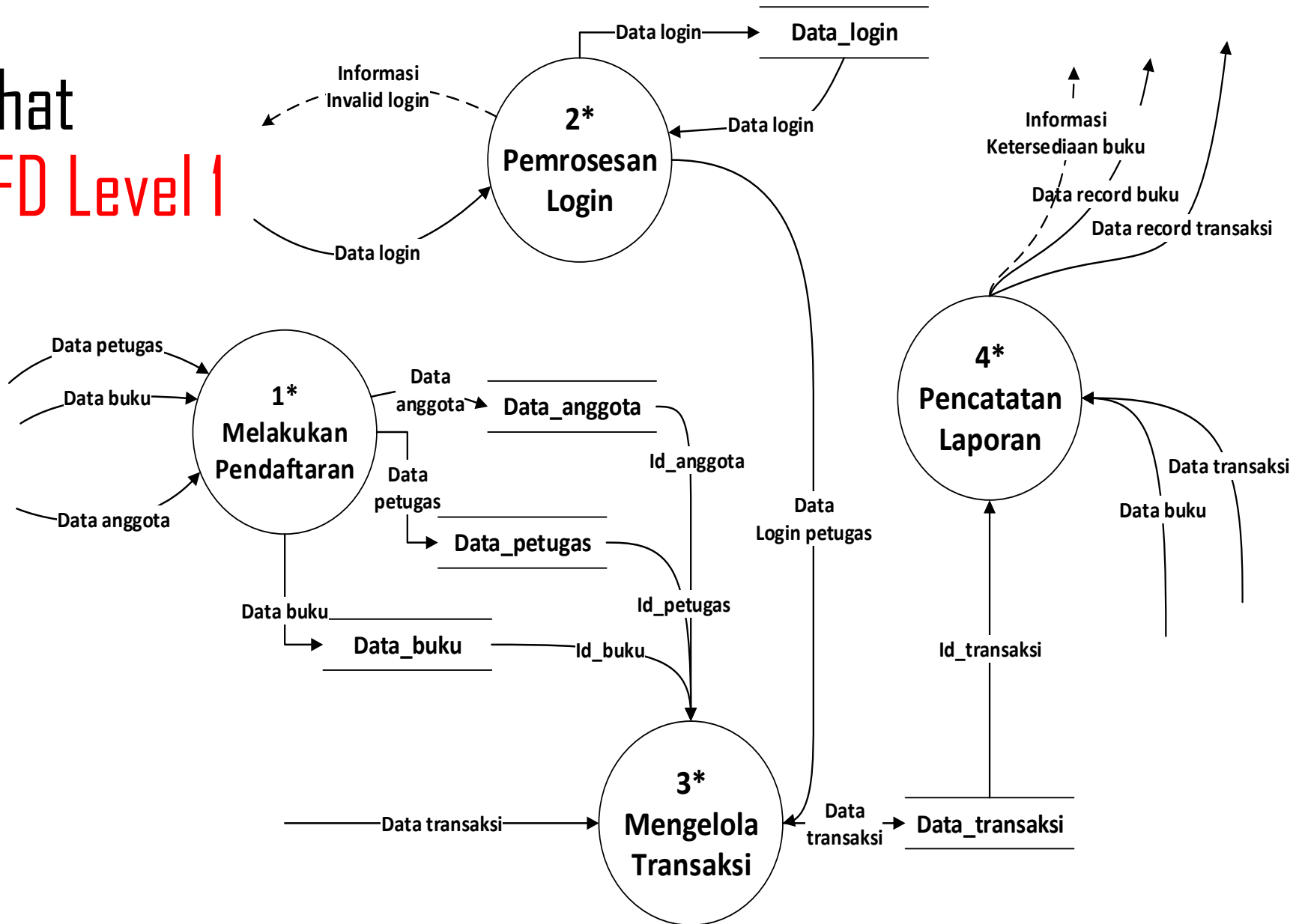


DFD

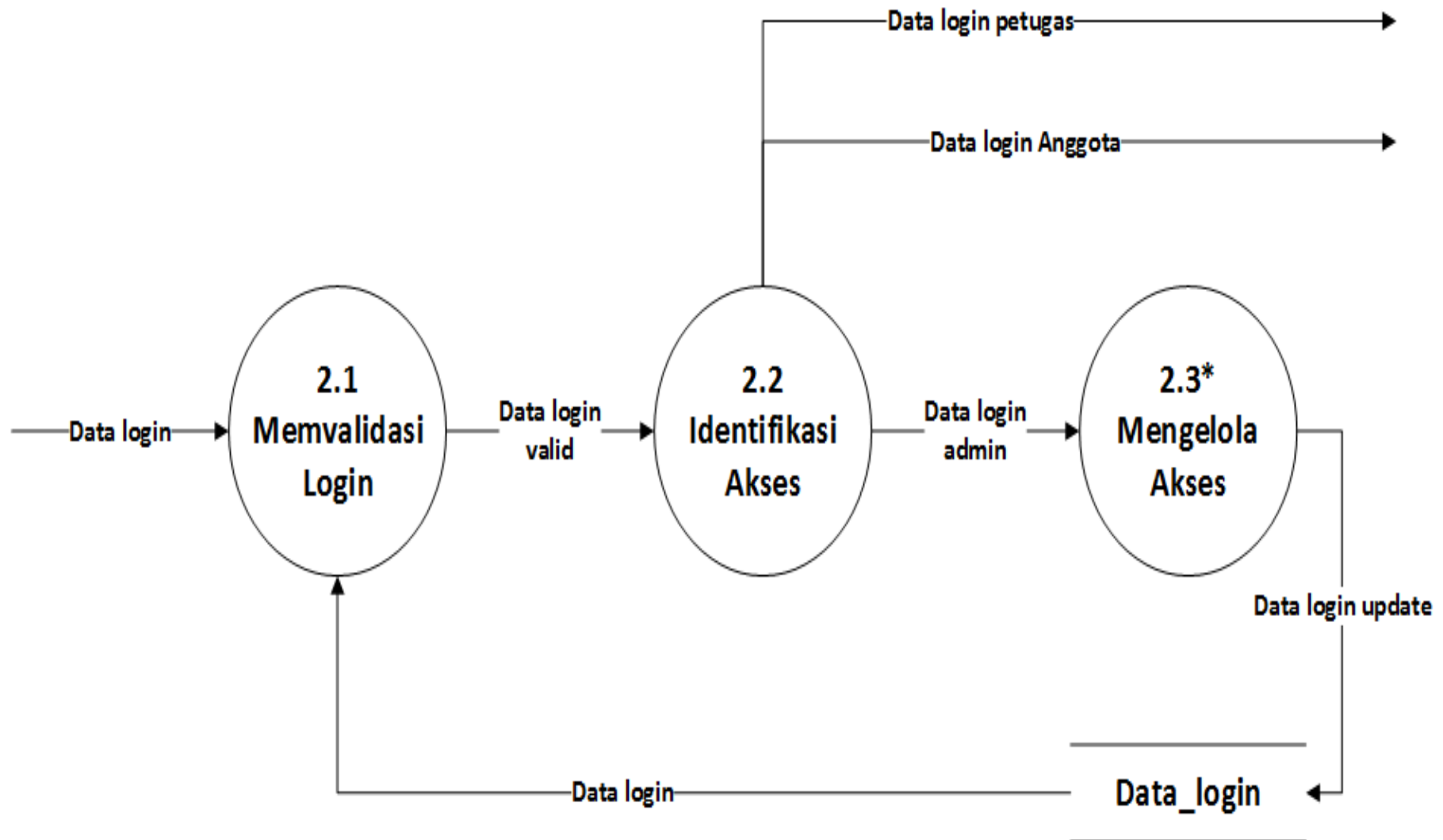
LEVEL 2



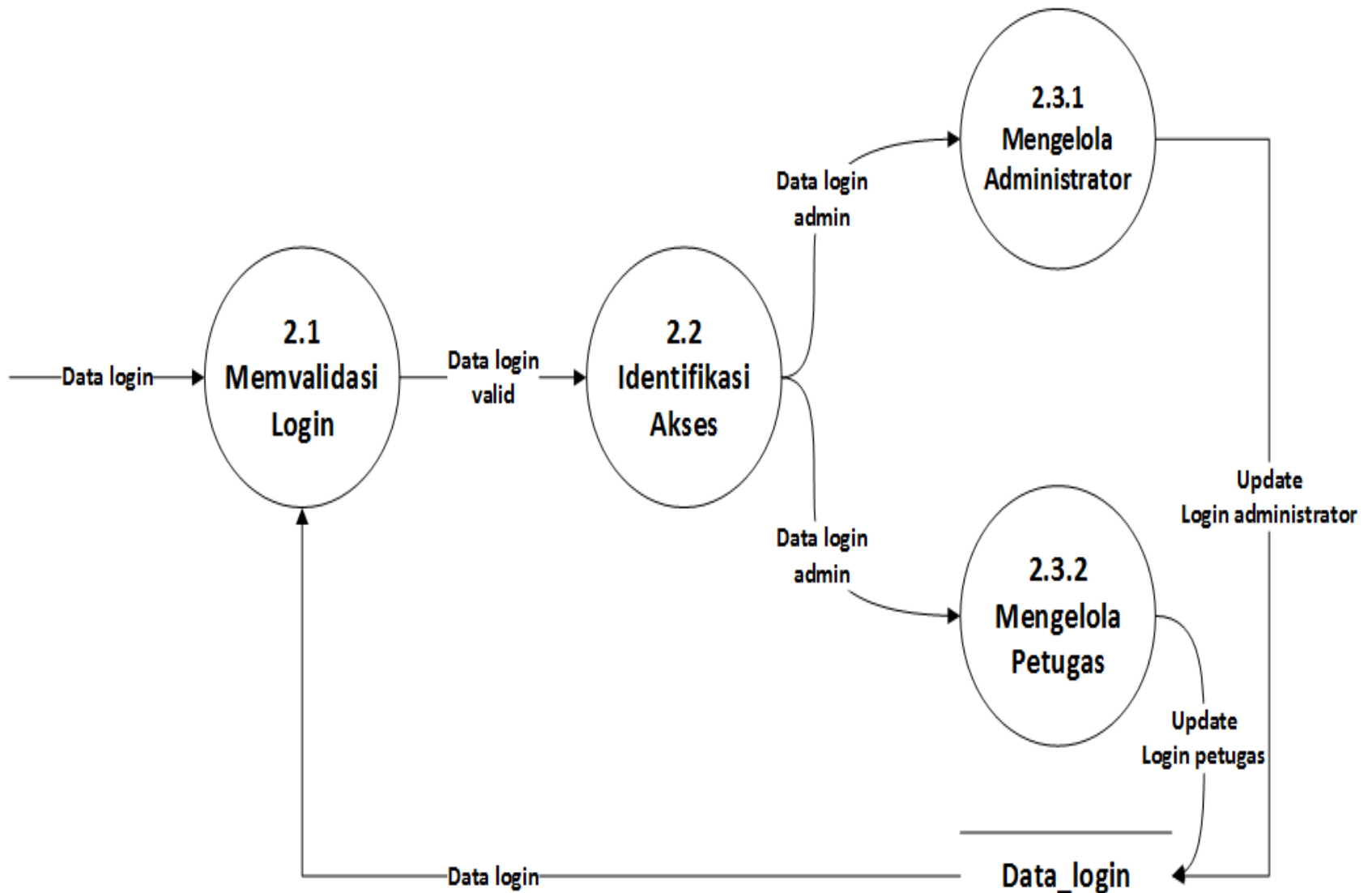
Lihat DFD Level 1



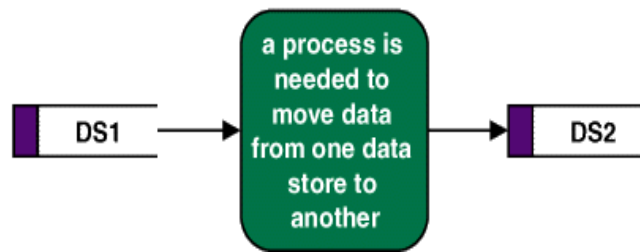
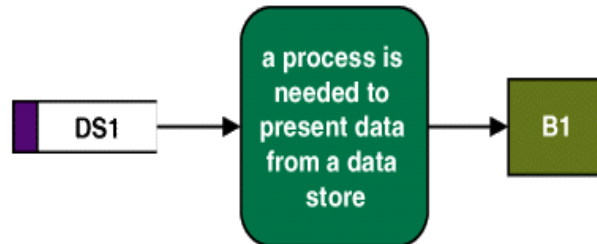
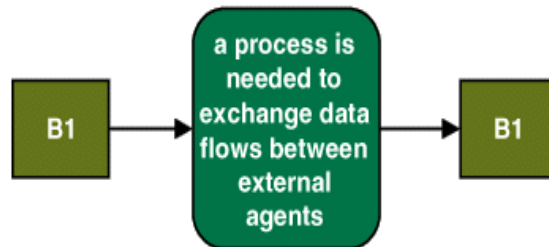
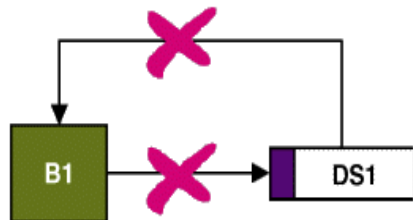
DFD LEVEL 2



DFD LEVEL 3

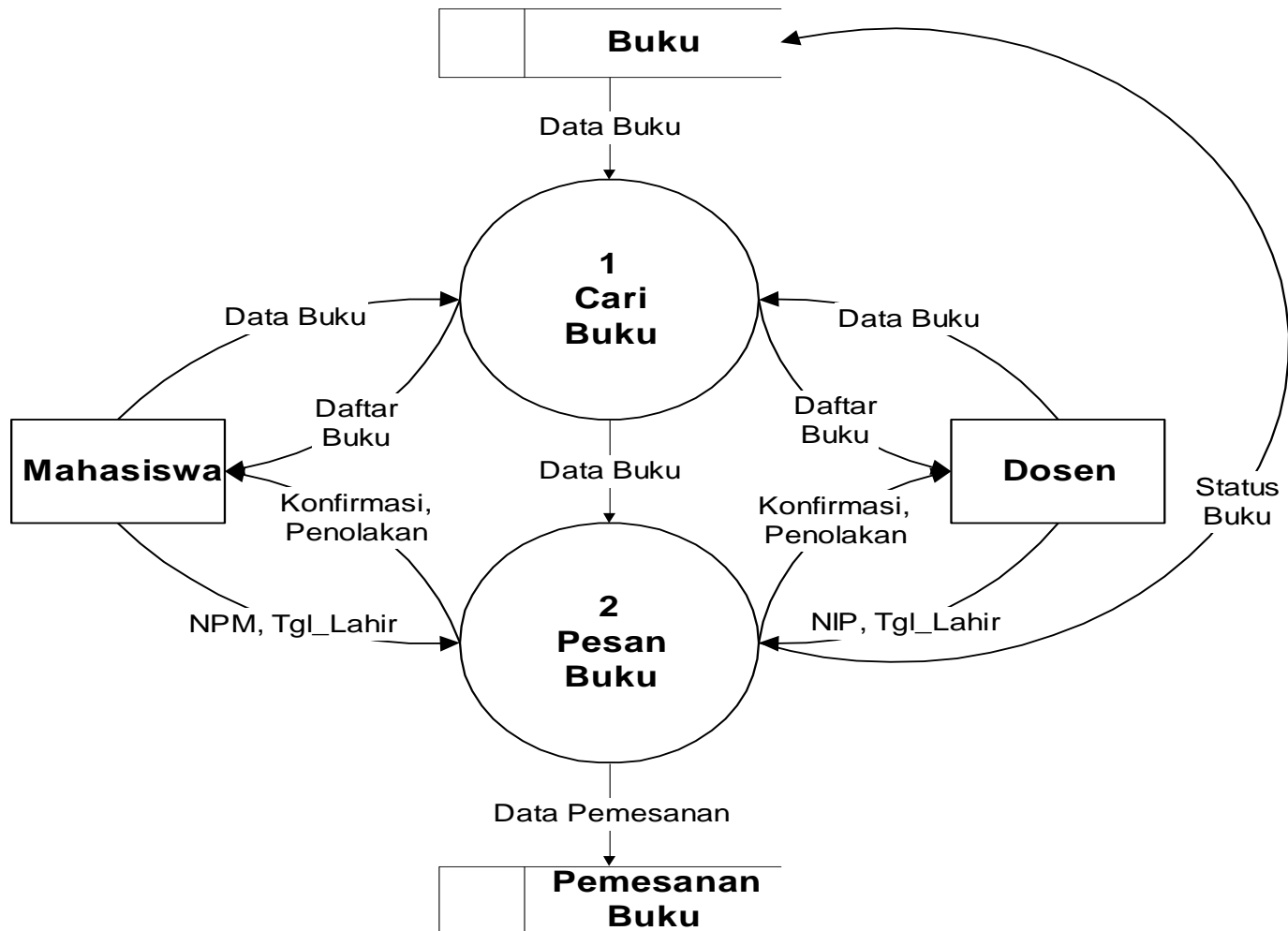


Hal-Hal "Kurang Pas" dalam DFD

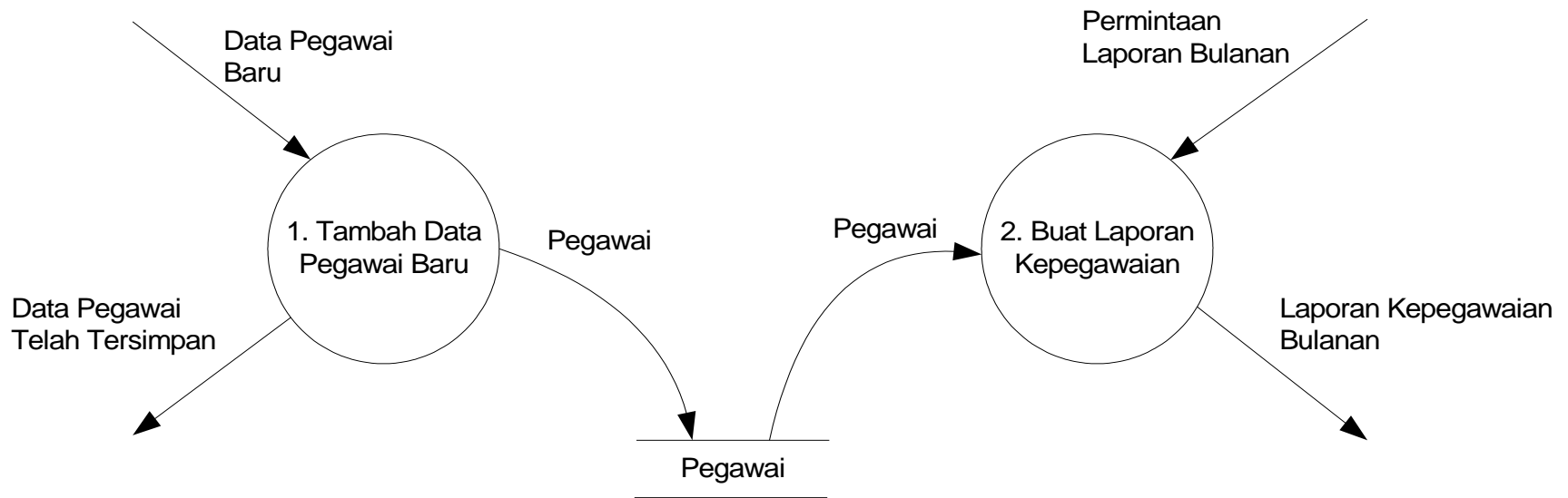


- Mencegah proses yang mempunyai masukan tetapi tidak mempunyai keluaran yang dikenal dengan **lubang hitam** (black-hole)
- Mencegah proses yang mempunyai keluaran tetapi tidak punya masukan, misalnya penghasil bilangan acak.
- Hati-hati dengan aliran dan proses yang tidak dinamakan karena dapat mengakibatkan elemen data yang saling tidak berhubungan menjadi satu.
- Hati-hati dengan penyimpanan yang punya status hanya dapat dibaca atau hanya dapat ditulis dan berkaitan dengan proses yang hanya memproses masukan atau hanya memproses keluaran.

Contoh DFD



Contoh DFD



6) Pemodelan Konsep OOP

UML (*Unified Modeling Language*)

- ❑ Sebuah bahasa yang berdasarkan grafik/gambar untuk memvisualisasi, menspesifikasikan, membangun, dan pendokumentasian dari sebuah sistem pengembangan software berbasis OO (*Object-Oriented*).
- ❑ UML sendiri juga memberikan standar penulisan sebuah sistem blue print, yang meliputi konsep bisnis proses, penulisan kelas-kelas dalam bahasa program yang spesifik, skema database, dan komponen-komponen yang diperlukan dalam sistem software.
- ❑ Masing masing diagram didesain sedemikian rupa agar pengembang dan user bisa melihat sistem perangkat lunak dari berbagai sudut pandang prespektif yang berbeda.



Tools UML

- ☐ Sparx EA
- ☐ ArgoUML (Free)
- ☐ StarUML (Free, OS)
- ☐ Rational Rhapsody (Commercial)
- ☐ Power Designer (Commercial)
- ☐ Visual Paradigm (Free, Commercial)
- ☐ dll

The Unified Modeling Language

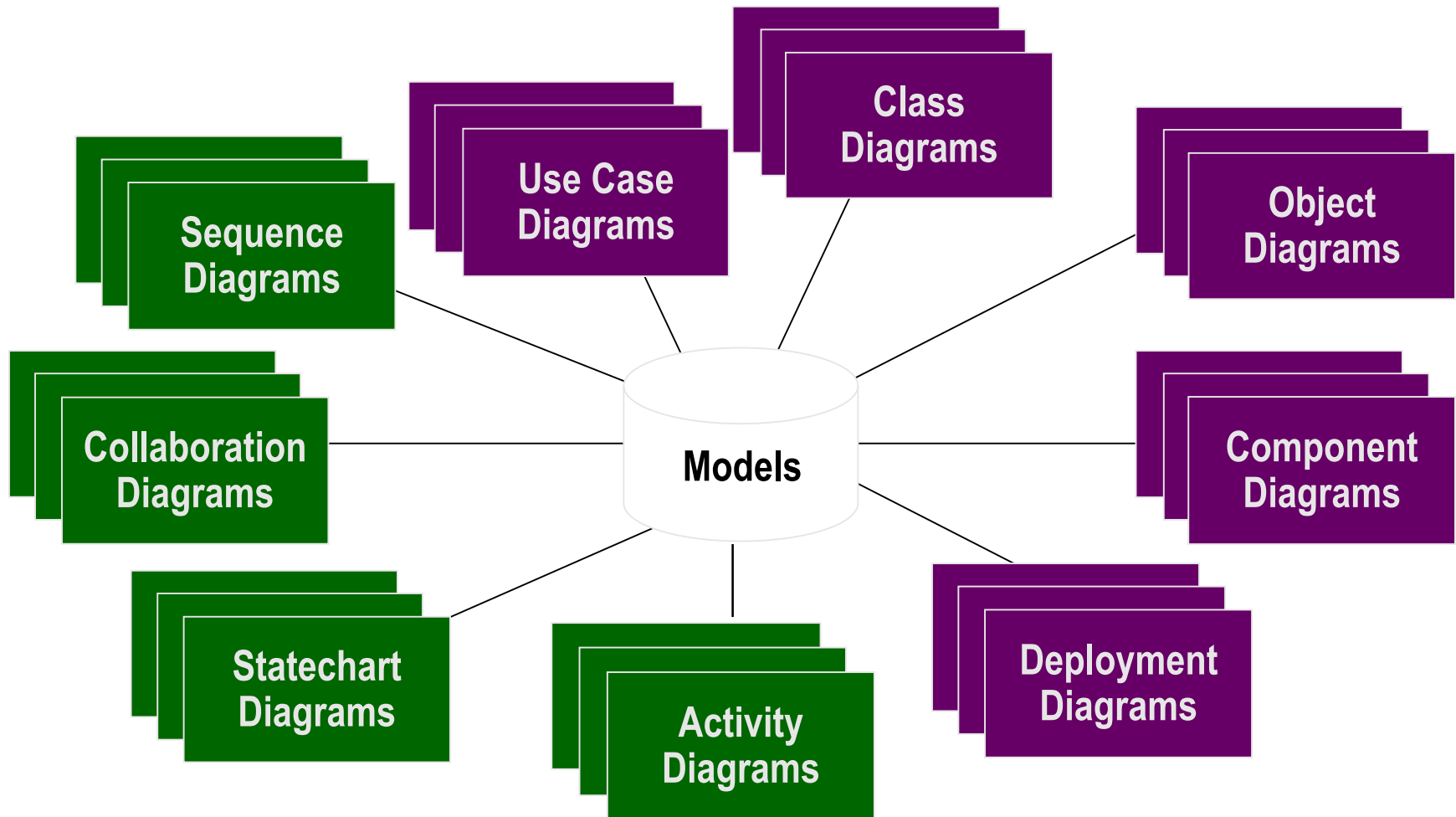
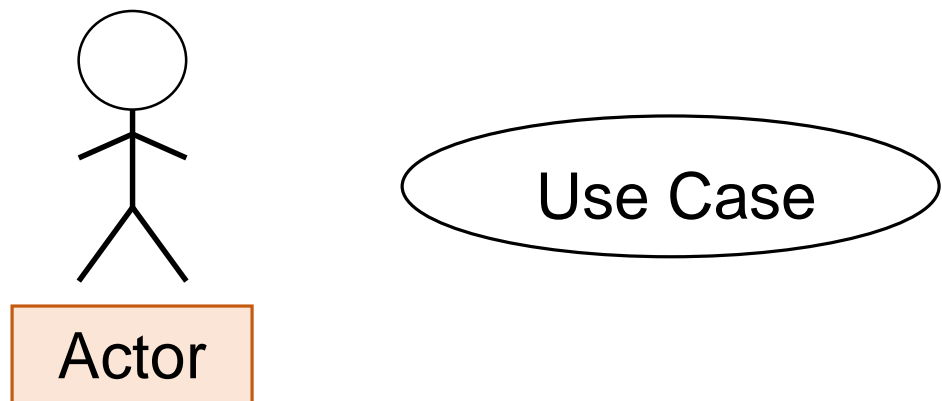


Diagram dalam UML

1. **Use Case Diagram** untuk memodelkan proses-proses bisnis
2. **Sequence Diagram** untuk memodelkan pengiriman pesan antara objek
3. **Collaboration Diagram** untuk memodelkan interaksi objek
4. **State Diagram** untuk memodelkan perilaku objek dalam sistem
5. **Activity Diagram** untuk memodelkan perilaku Use Case, objek, atau operasi
6. **Class Diagram** untuk memodelkan struktur statis Class dalam sistem
7. **Object Diagram** untuk memodelkan struktur statis Objek dalam sistem
8. **Component Diagram** untuk memodelkan komponen
9. **Deployment Diagram** untuk memodelkan distribusi sistem

Use case diagram

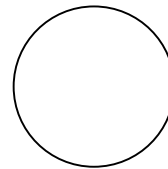
- Usecase adalah adalah sebuah set skenario yang menggambarkan interaksi antara user dengan sistem.
- Diagram use case menampilkan hubungan diantara aktor dan use cases.
- Dua buah komponen utama dari diagram use case adalah aktor dan use case



Lambang Use Case



Aktor



Usecase

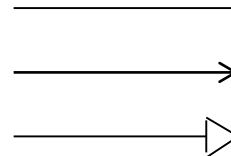


Catatan



`<<include>>`

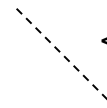
Include



Relasi Aktif

Relasi Pasif

Generalisasi



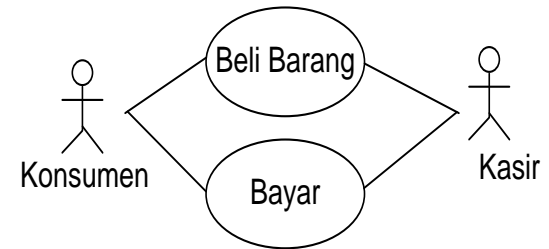
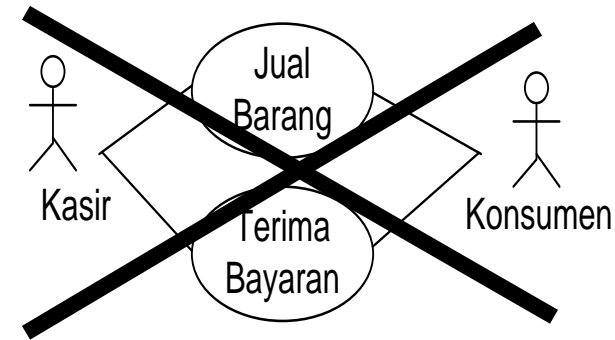
`<<extend>>`

extend

Use Case Diagram

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

- Secara umum use case adalah:
 - Pola perilaku system
 - Urutan transaksi yang berhubungan yang dilakukan oleh satu actor
- Use case diagram terdiri dari
 - Use case
 - Actors
 - Relationship
 - System boundary boxes (optional)
 - Packages (optional)



Use Case Diagram

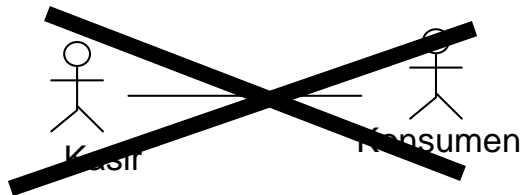
- Use case dibuat berdasar keperluan actor, merupakan “apa” yang dikerjakan system, bukan “bagaimana” system mengerjakannya
- Use case diberi nama yang menyatakan apa hal yang dicapai dari hasil interaksinya dengan actor.
- *Use case* dinotasikan dengan gambar (horizontal ellipse)
- Use case biasanya menggunakan 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
- 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

Actor-use Case Diagram

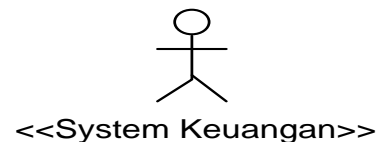
- ❑ Actor menggambarkan orang, system atau external entitas / stakeholder yang menyediakan atau menerima informasi dari system
- ❑ Actor memberi input atau menerima informasi dari system
- ❑ Actor biasanya menggunakan Noun
- ❑ Actor digambarkan dengan gambar stick figure atau dengan gambar visual



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

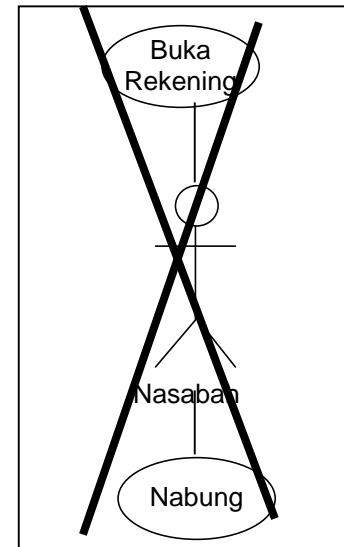
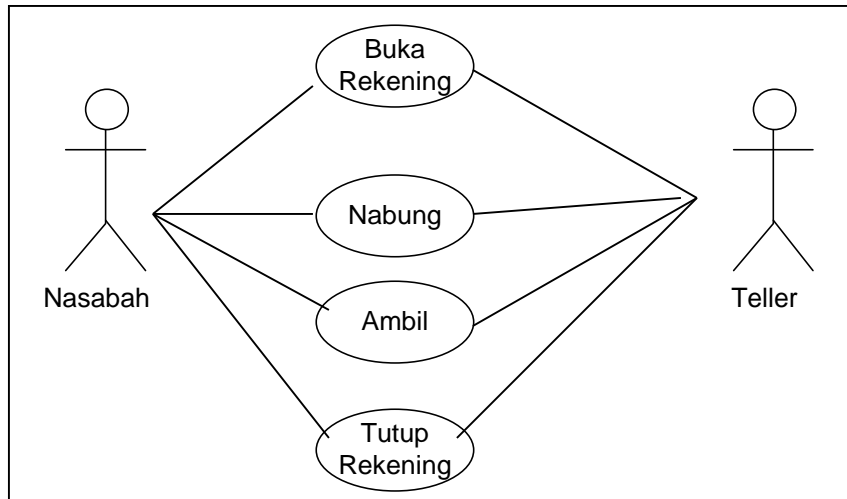


- ❑ Indikasi <<system>> untuk sebuah actor yang merupakan sebuah system



Actor-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)



- ❑ Actor jangan digambarkan ditengah-tengah use cases (actors are placed to the outside of the diagram, and not the middle of it)

Use case diagram

- **Aktor** mempresentasikan user atau sistem lain yang akan berinteraksi dengan system yang dimodelkan.
- **Use case** adalah external view dari sebuah sistem yang merepresentasikan beberapa tindakan (action) yang akan dilakukan oleh user untuk melengkapi suatu tugas.
- Untuk membuat diagram use case , dapat di awali dengan membuat daftar tindakan yang dilakukan oleh user untuk menyelesaikan sebuah action.
- Contoh dari use case adalah proses pencarian sebuah web berdasarkan keyword tertentu:
 1. Customer enters the keyword
 2. Customer clicks the search button
 3. The search is executed
 4. The results are shown

Relasi Use case diagram

ASOSIASI

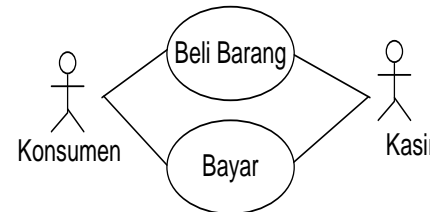
- Pada diagram use case, asosiasi digambarkan antara aktor dan usecase untuk menunjukkan bahwa aktor akan melaksanakan tugas dari usecase.
- Aktor bisa melaksanakan beberapa tugas di use case, dan sebaliknya use case bisa di eksekusi oleh beberapa aktor
- Ada 4 jenis relasi yang bisa timbul pada use case diagram
 1. Asosiasi antara actor dan use case
 2. Asosiasi antara use case
 3. Generalization/Inheritance antara use case
 4. Generalization/Inheritance antara actor
- Associations bukan menggambarkan aliran data/informasi
- Associations digunakan untuk menggambarkan bagaimana actor terlibat dalam use case

Association – Use Case Diagram

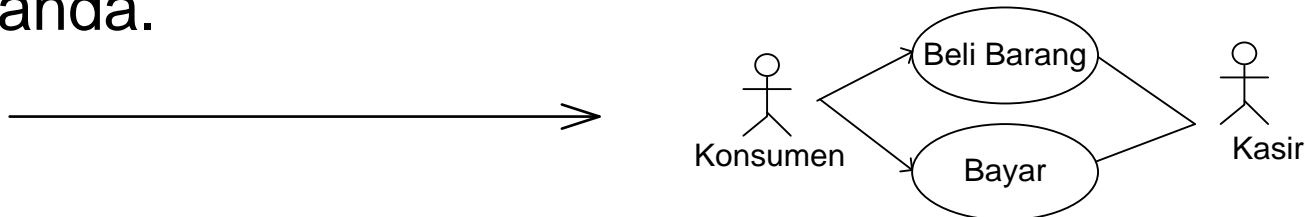
Association antara actor dan use case

- 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



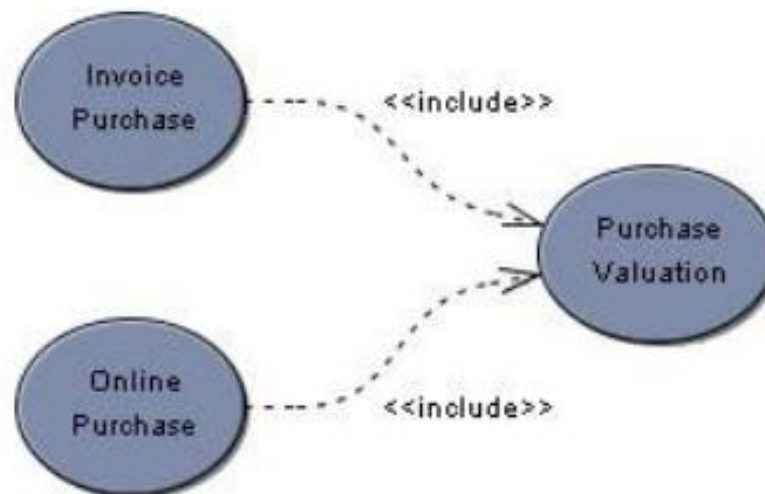
- association antara actor dan use case yang menggunakan panah terbuka untuk mengindikasikan bila actor berinteraksi secara **pasif** dengan system anda.



Relasi Use case diagram

INCLUDE atau <<include>>

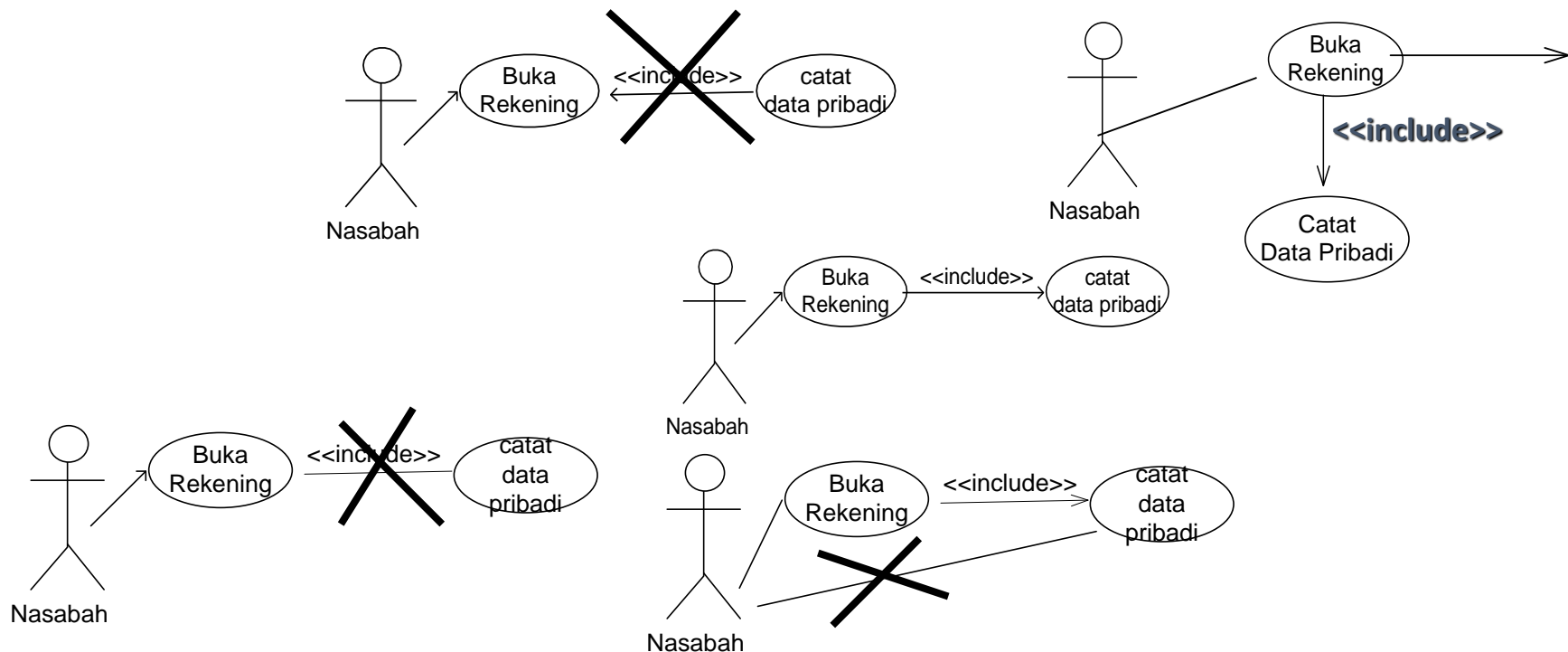
- Use case juga bisa berelasi dengan usecase lainnya dengan dua hubungan yang berbeda.
- Yang pertama includes, yang artinya sebuah use case adalah bagian dari skenario dari use case lain.
- Pada contoh dibawah, invoice dan online purchase adalah bagian dari skenario purchase evaluation



Association - Use Case Diagram

<<include>>

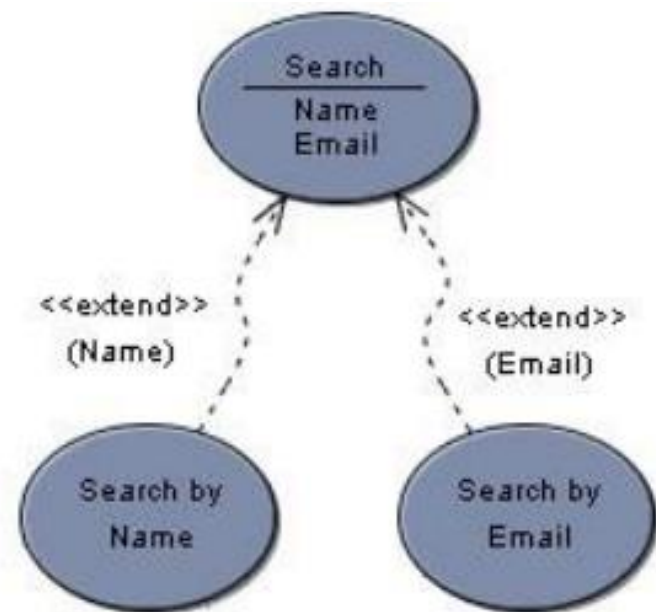
- termasuk didalam use case lain (required) / (diharuskan)
- Pemanggilan use case oleh use case lain
- contohnya adalah Pemanggilan sebuah fungsi program
- *Gambarkan association <<include>> secara horizontal*
- Tanda panah terbuka harus terarah ke sub use case
- Tidak boleh actor dihubungkan pada use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



Relasi Use case diagram

EXTENDS

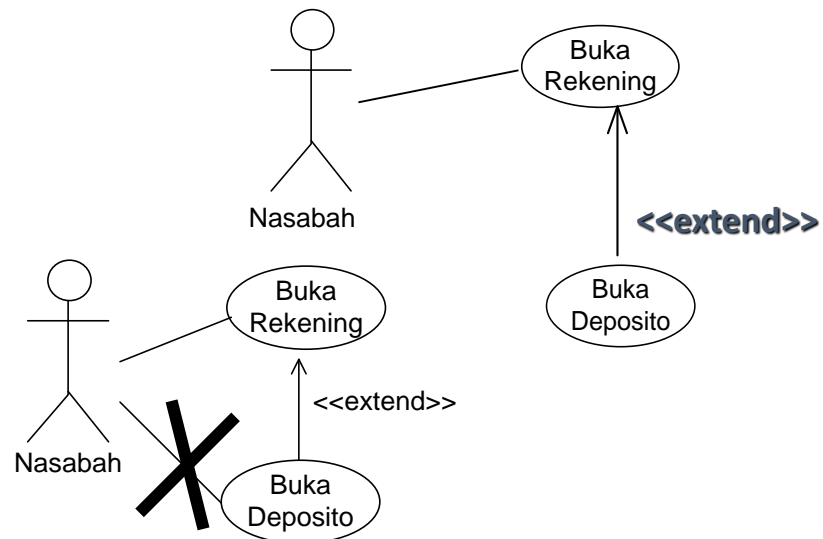
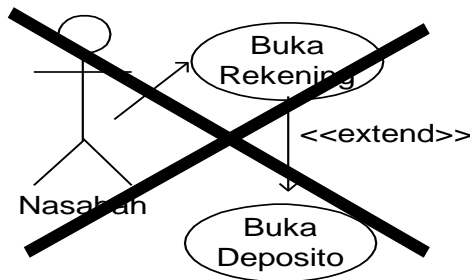
- Apabila ingin menggambarkan suatu use case yang memiliki bentuk variasi yang berbeda dari use case lainnya (perluasan dari suatu usecase) gunakan relasi extend
- Kurangi penggunaan relasi extend, terlalu banyak pemakaian relasi ini membuat diagram sulit dipahami.



Association - Use Case Diagram

Association antara use case

- <<extend>>
 - Perluasan dari use case lain jika kondisi atau syarat terpenuhi (*Optional Behaviour*)
 - 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>>



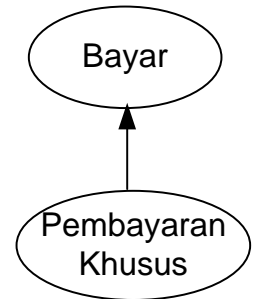
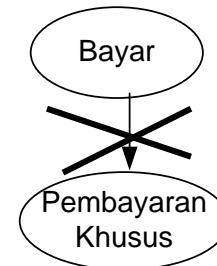
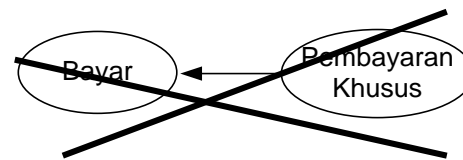
Association - Use Case Diagram

Generalization/inheritance

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

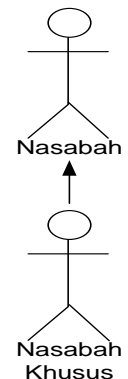
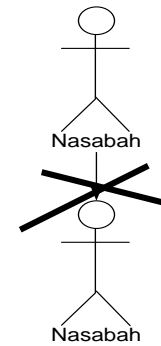
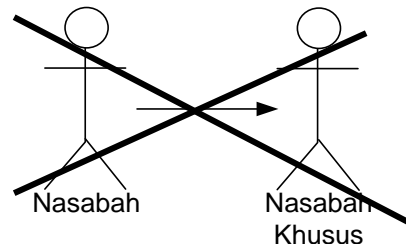
Generalization/inheritance antara use case

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



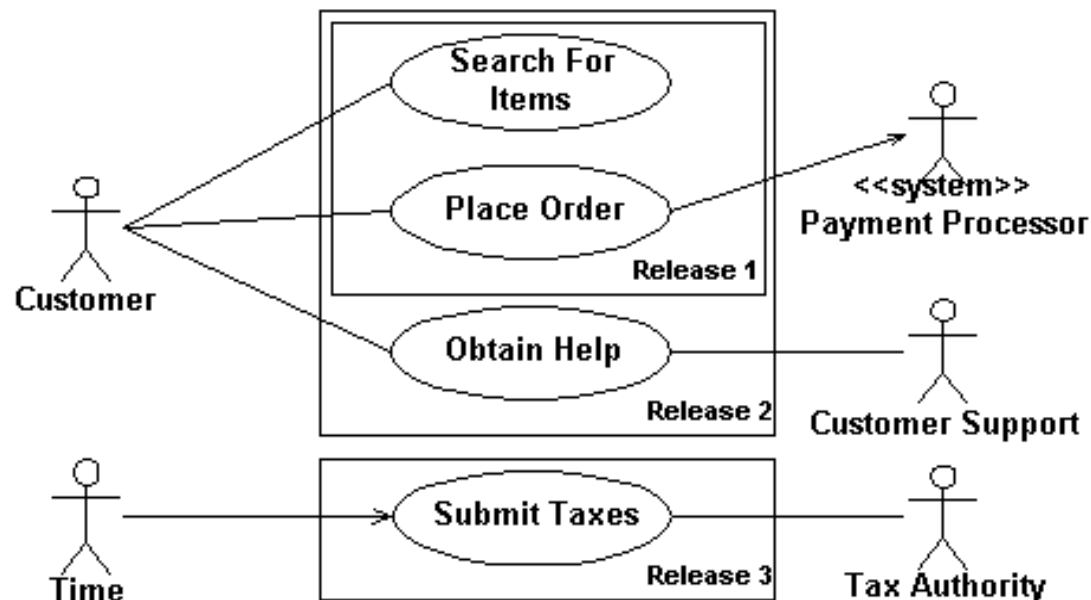
Generalization/inheritance antara actor

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

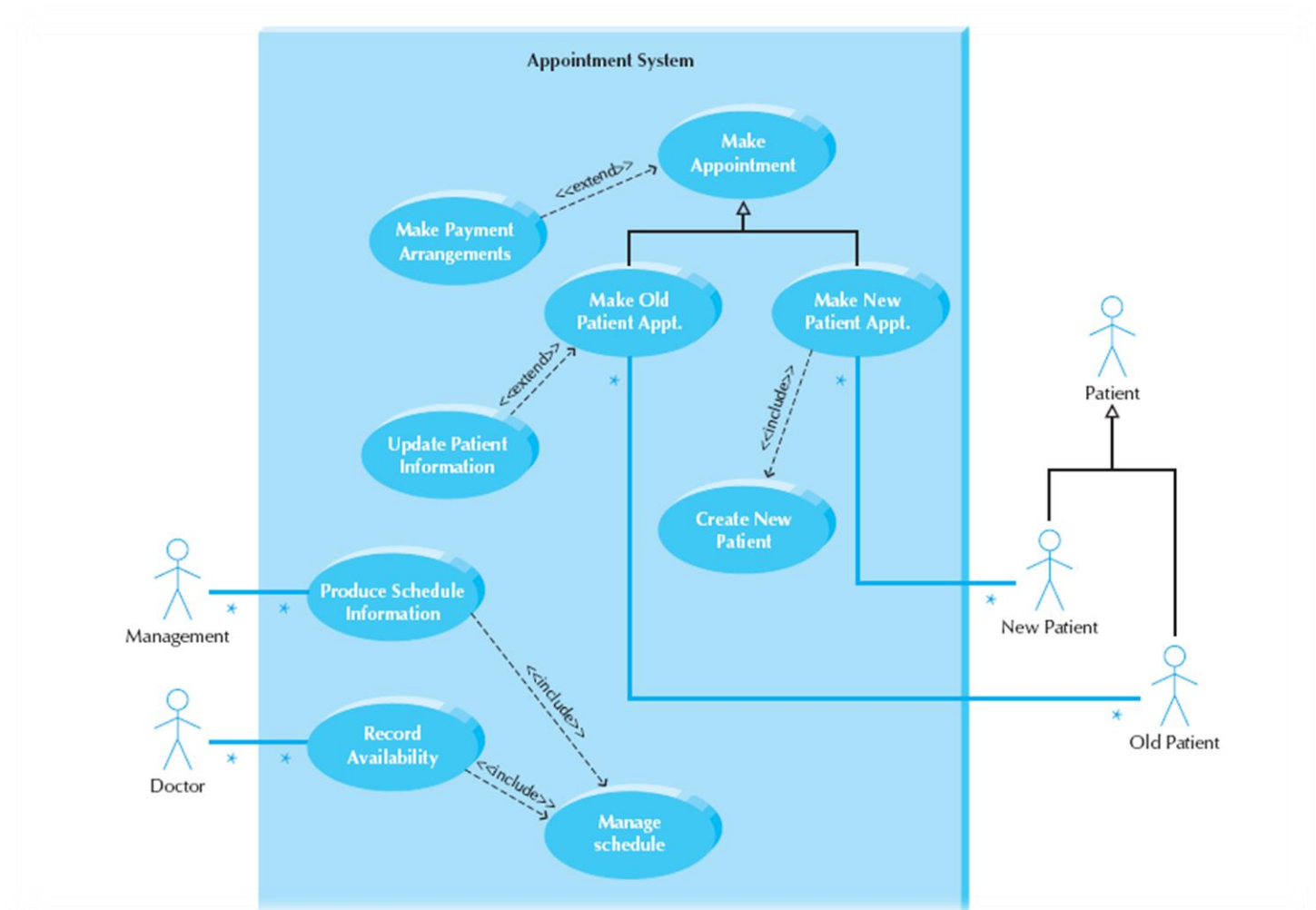


Use case System boundary boxes

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






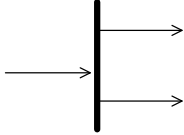
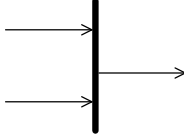

Contoh Use case diagram



ACTIVITY DIAGRAM

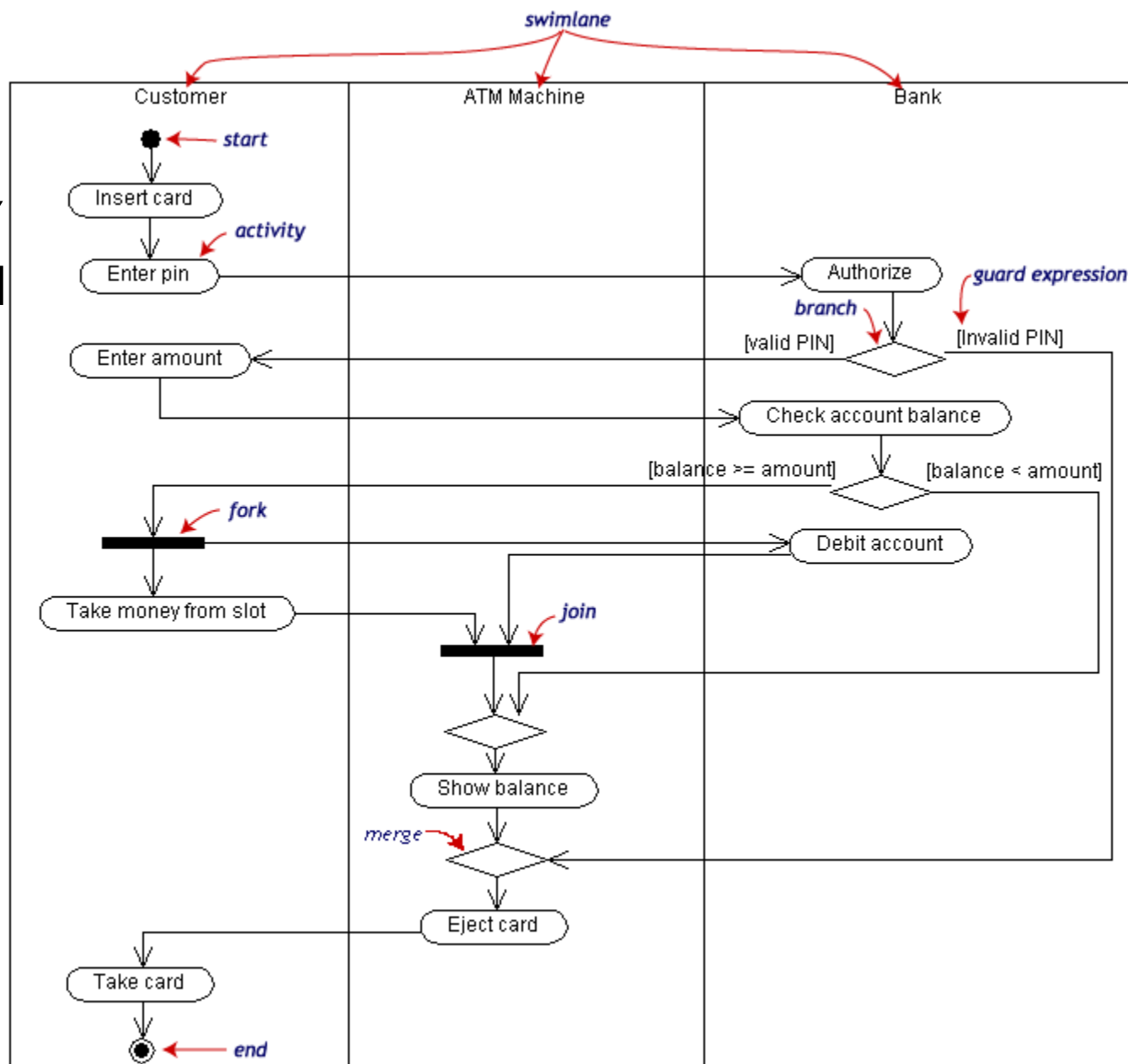
- ☐ Menggambarkan proses bisnis dan urutan aktivitas dalam sebuah proses
- ☐ Dipakai pada business modeling untuk memperlihatkan urutan aktifitas proses bisnis
- ☐ Struktur diagram ini mirip flowchart atau Data Flow Diagram pada perancangan terstruktur
- ☐ Sangat bermanfaat apabila kita membuat diagram ini terlebih dahulu dalam memodelkan sebuah proses untuk membantu memahami proses secara keseluruhan
- ☐ Activity diagram dibuat berdasarkan sebuah atau beberapa use case pada use case diagram

Simbol Activity Diagram

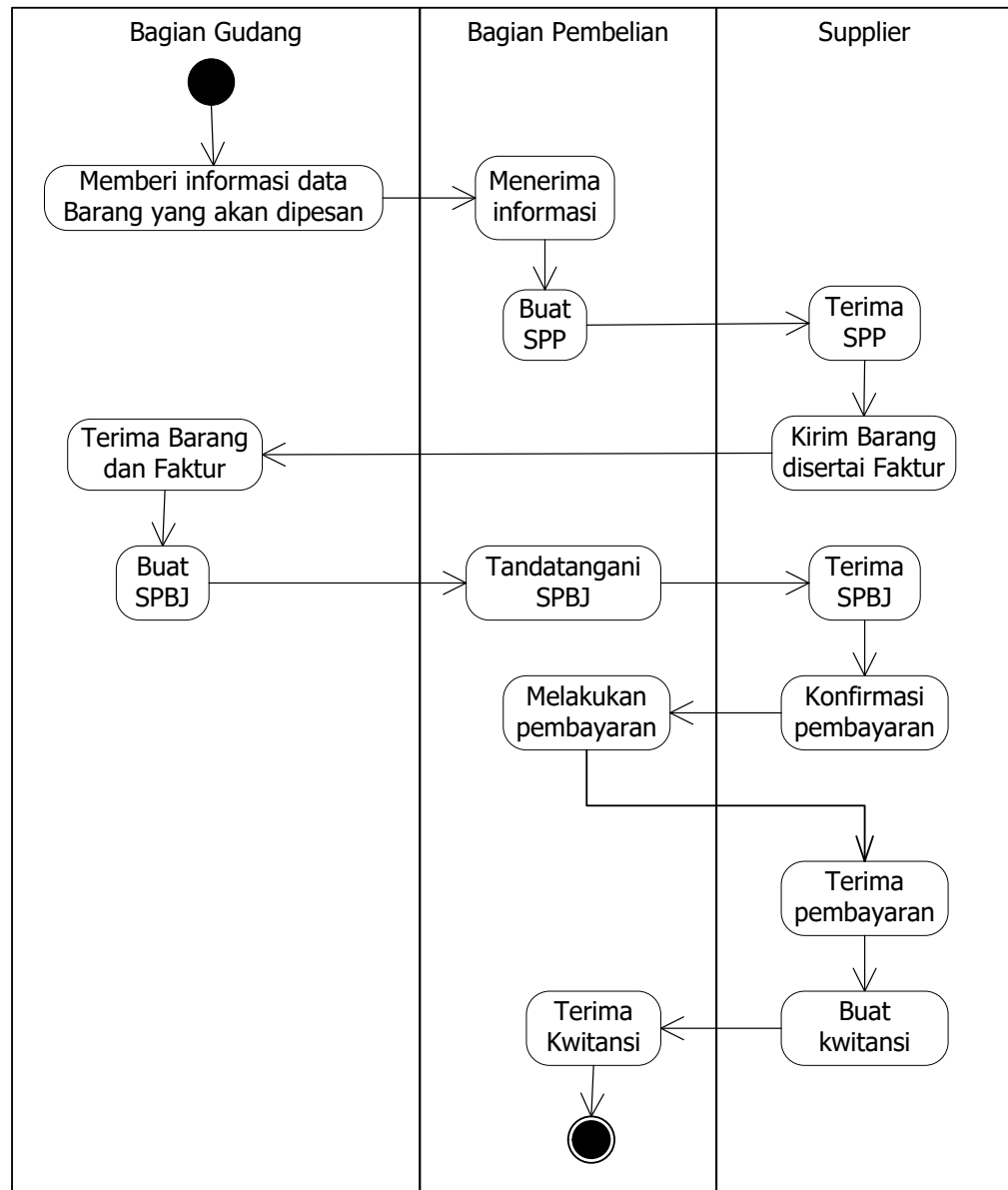
Simbol	Keterangan
	Start Point
	End Point
	Activities
	Fork (Percabangan)
	Join (Penggabungan)
	Decision
Swimlane	Sebuah cara untuk mengelompokkan activity berdasarkan Actor (mengelompokkan activity dalam sebuah urutan yang sama)

CONTOH ACTIVITY DIAGRAM

Penarik
an Uang
dari
Account
Bank
Melalui
ATM

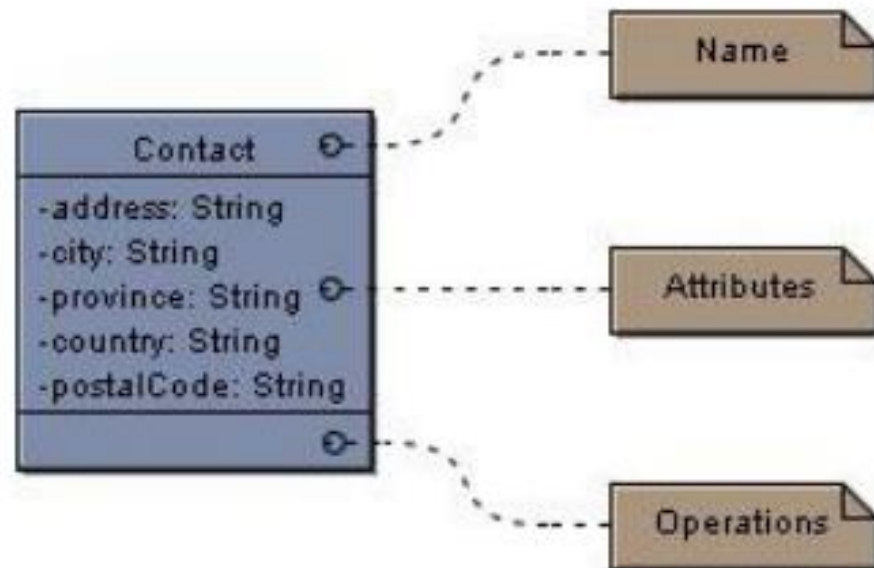


CONTOH ACTIVITY DIAGRAM



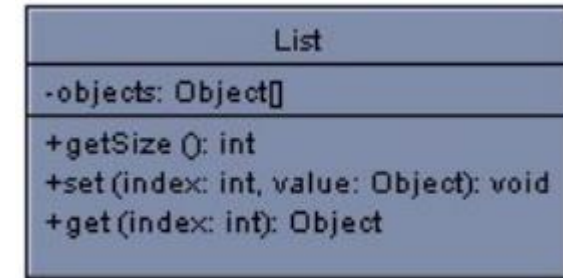
Class Diagram

- Class diagram adalah inti dari desain Object oriented yang menggambarkan tipe dari objek didalam sistem dan relasi diantara masing masing objek
- *Class* didalam OO digunakan untuk merepresentasikan entitas dari suatu sistem.
- Terdapat beberapa bagian didalam class yaitu nama, atribut, dan operations (method).



Class Diagram

- **Atribut** adalah properti dari sebuah class. Memiliki acces modifier sbb:
 - Private : tidak dapat dipanggil dari luar *class* yang bersangkutan
 - + Public : dapat dipanggil oleh siapa saja
 - # Protected : hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya
- **Operation** adalah fungsi atau method didalam class.



+ getSize (): int

(dibaca method getSize memiliki modifier public, tidak memiliki parameter dan memiliki nilai kembalian int)

+ set (index: int, value: Object): void

(dibaca method set memiliki modifier public, memiliki 2 buah parameter yaitu index dengan tipe parameter int dan value dengan tipe parameter object serta tidak memiliki nilai kembalian karena bertipe void).

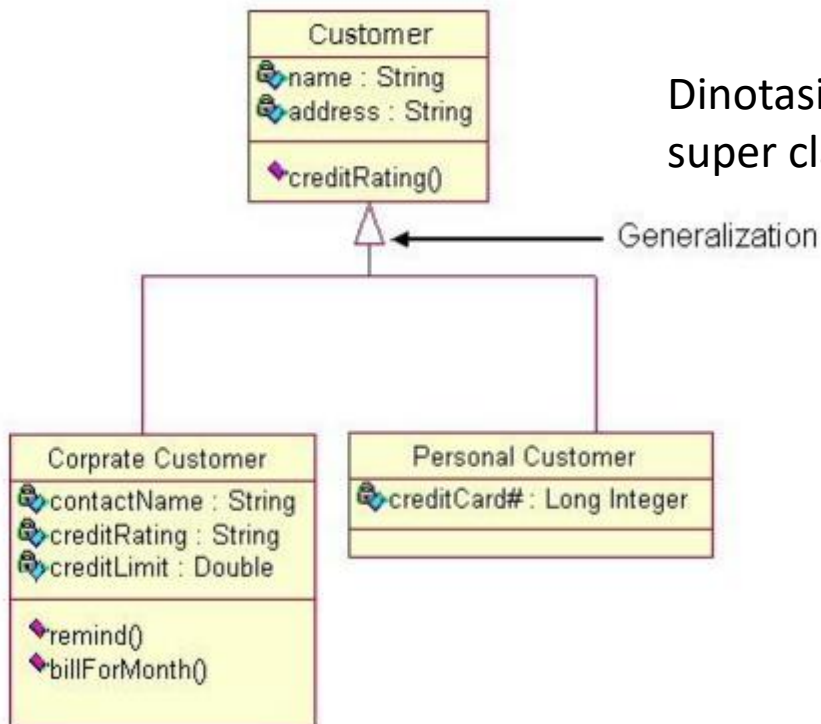
Multiplicity

- ❑ Indikasi berapa banyak objek yang bisa mengisi properti
 - 1 (pasti 1)
 - 0..1 (0 atau 1)
 - * (Tidak ada batasan, bisa 0, 1, ..., n)
- ❑ Biasanya didefinisikan batas bawah dan atas, kecuali untuk yang pasti bernilai 1
- ❑ Mirip dengan konsep one-to-one dan one-to-many pada relational database

Relasi dalam Class Diagram

Generalization (pewarisan)

- Relasi jenis ini memungkinkan suatu kelas mewarisi attribute dan operasi yang dimiliki oleh base Class. Attribute dan operasi yang bisa diwarisi oleh suatu kelas adalah yang memiliki access modifier public, protected dan default.

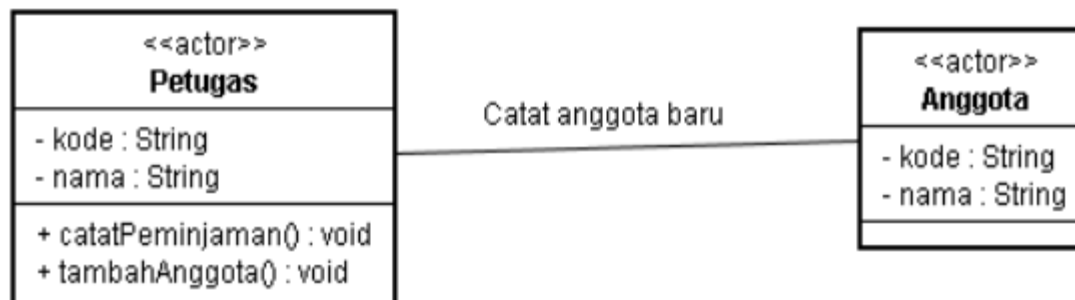


Dinotasikan dengan anak panah mengacu ke super class

Relasi dalam Class Diagram

Asosiasi

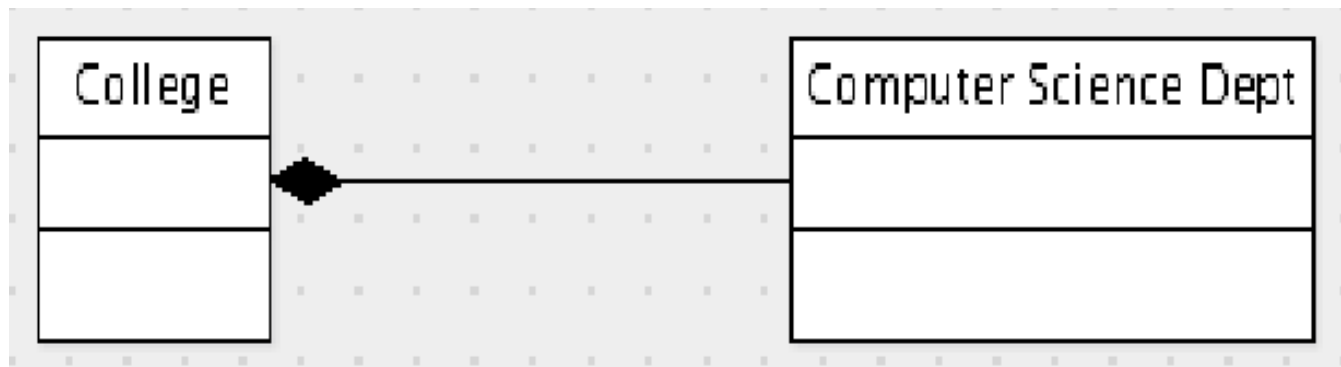
- Asosiasi adalah hubungan yang terjadi antara class yang ada. Asosiasi memungkinkan suatu class untuk menggunakan atau mengetahui atribut atau operasi yang dimiliki oleh class lain.
- Asosiasi juga menggambarkan interaksi yang mungkin terjadi antara satu class dengan class yang lain.



Contoh Asosiasi

□ Contoh:

- Kampus memiliki fakultas CS atau kampus terdiri dari fakultas CS (salah satunya)
- Tanpa ada kampus, maka tidak ada fakultas CS



Relasi dalam Class Diagram

Aggregasi

❑ Relasi agregasi adalah suatu bentuk relasi yang jauh lebih kuat dari pada asosiasi. Agregasi dapat diartikan bahwa suatu kelas merupakan bagian dari kelas yang lain namun bersifat tidak wajib.

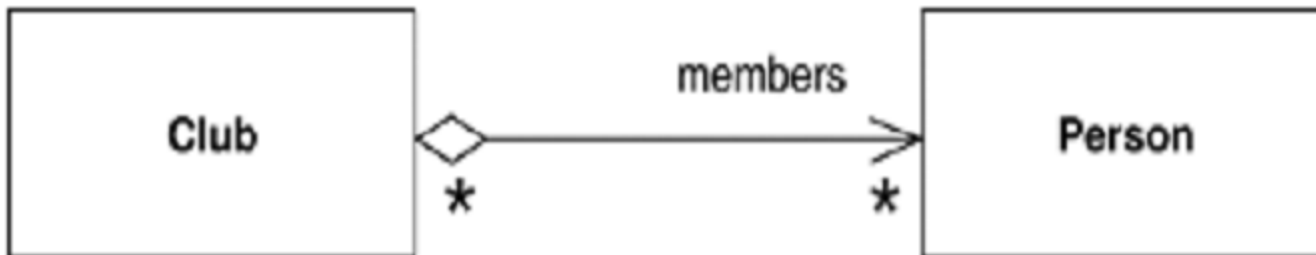
❑ Contoh:

sebuah buku memiliki pengarang, daftar pustaka, namun bisa saja suatu buku tidak memiliki daftar pustaka. Dari contoh kasus dapat diartikan bahwa daftar pustaka merupakan bagian dari buku namun buku tetap disebut sebagai buku meskipun tidak memiliki daftar pustaka.



Contoh Aggregation

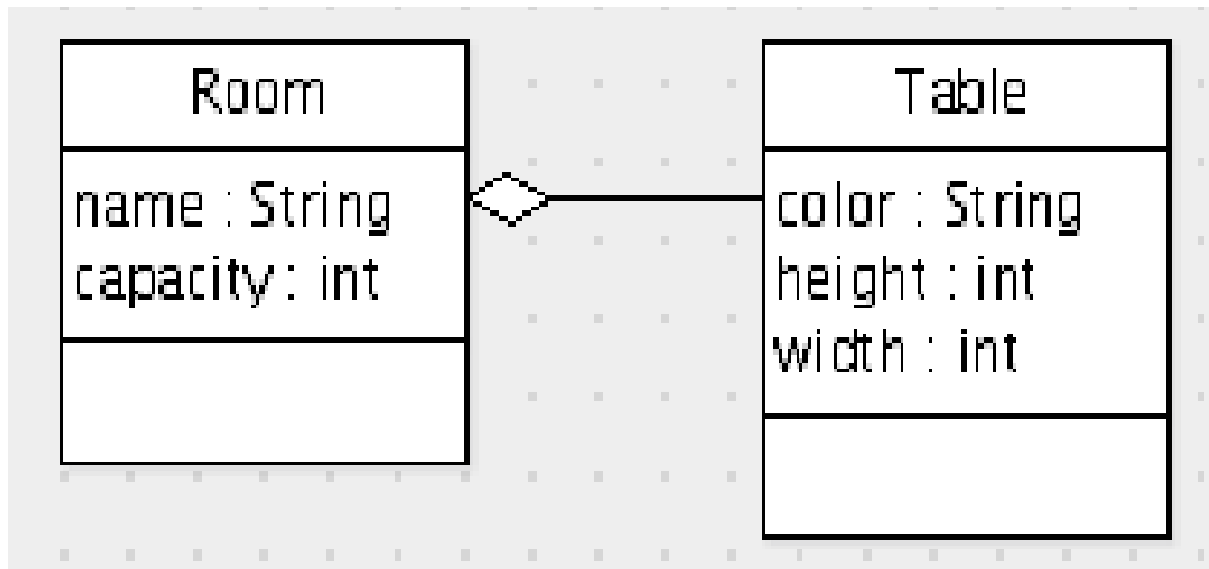
- ❑ Dinotasikan dengan diamond “kosong”
- ❑ Contoh:
 - Klub memiliki banyak anggota
 - Orang bisa memiliki makna tersendiri tanpa kehadiran sebuah klub
 - Jika dipisah, tidak merubah makna



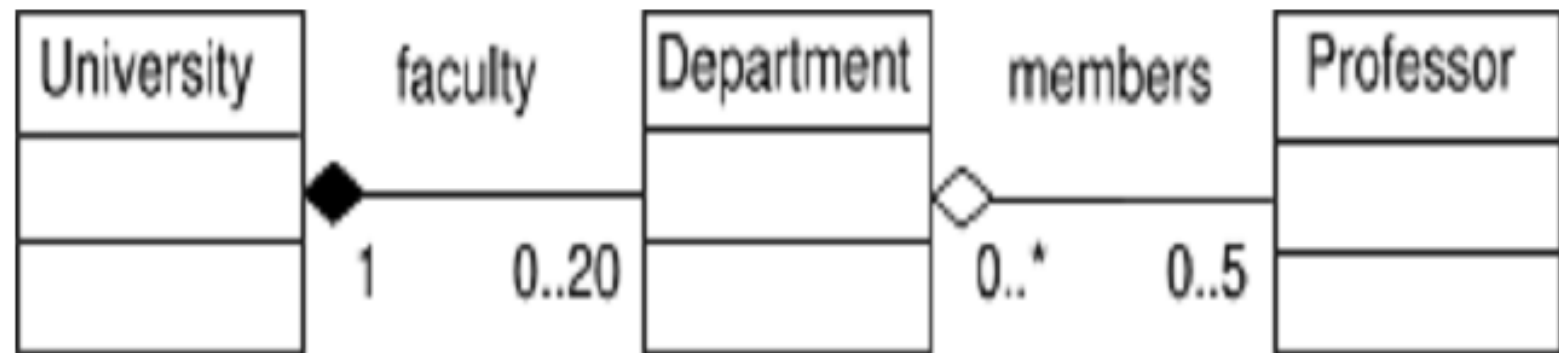
Contoh Aggregation

□ Contoh:

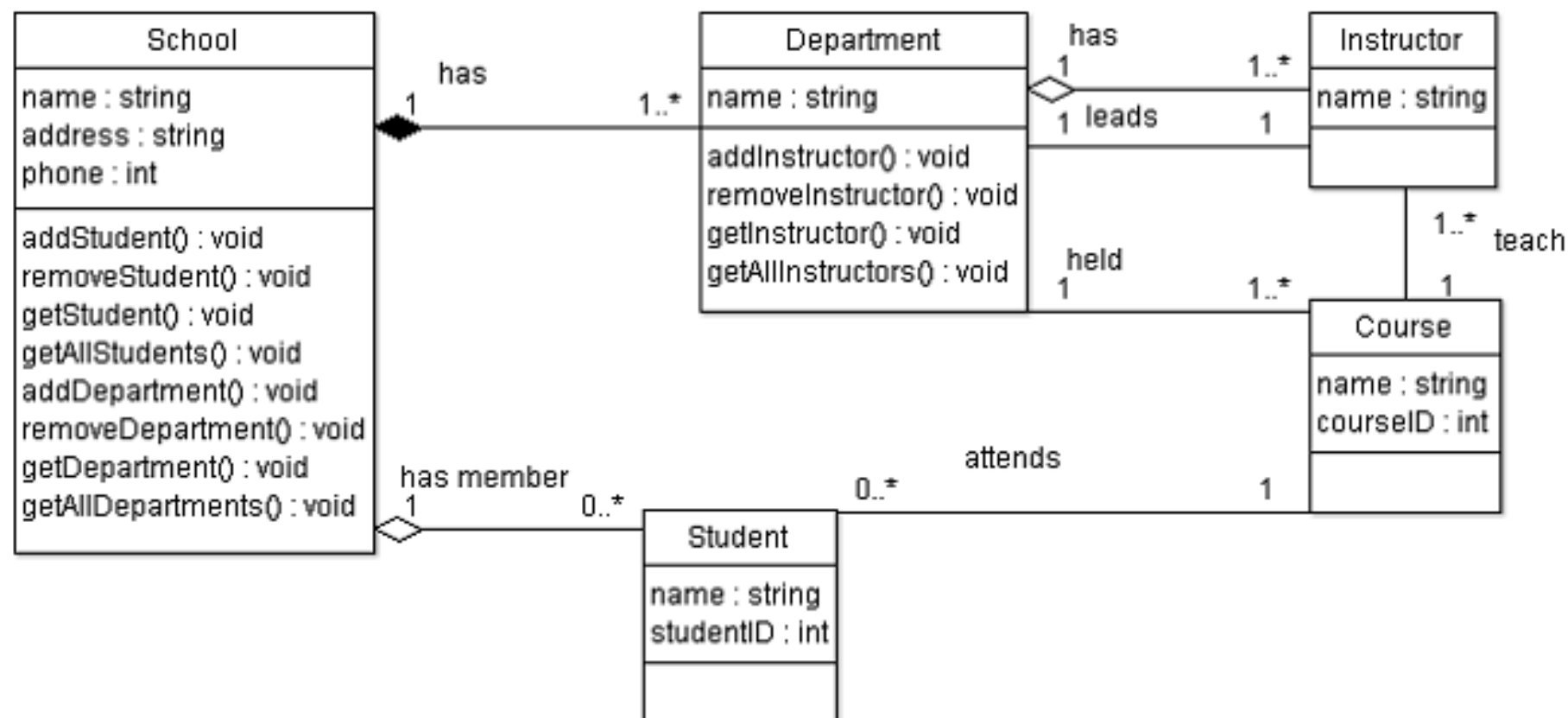
- Sebuah ruangan memiliki meja dan kursi
- Tanpa kehadiran ruang, meja dan kursi bisa tetap ada

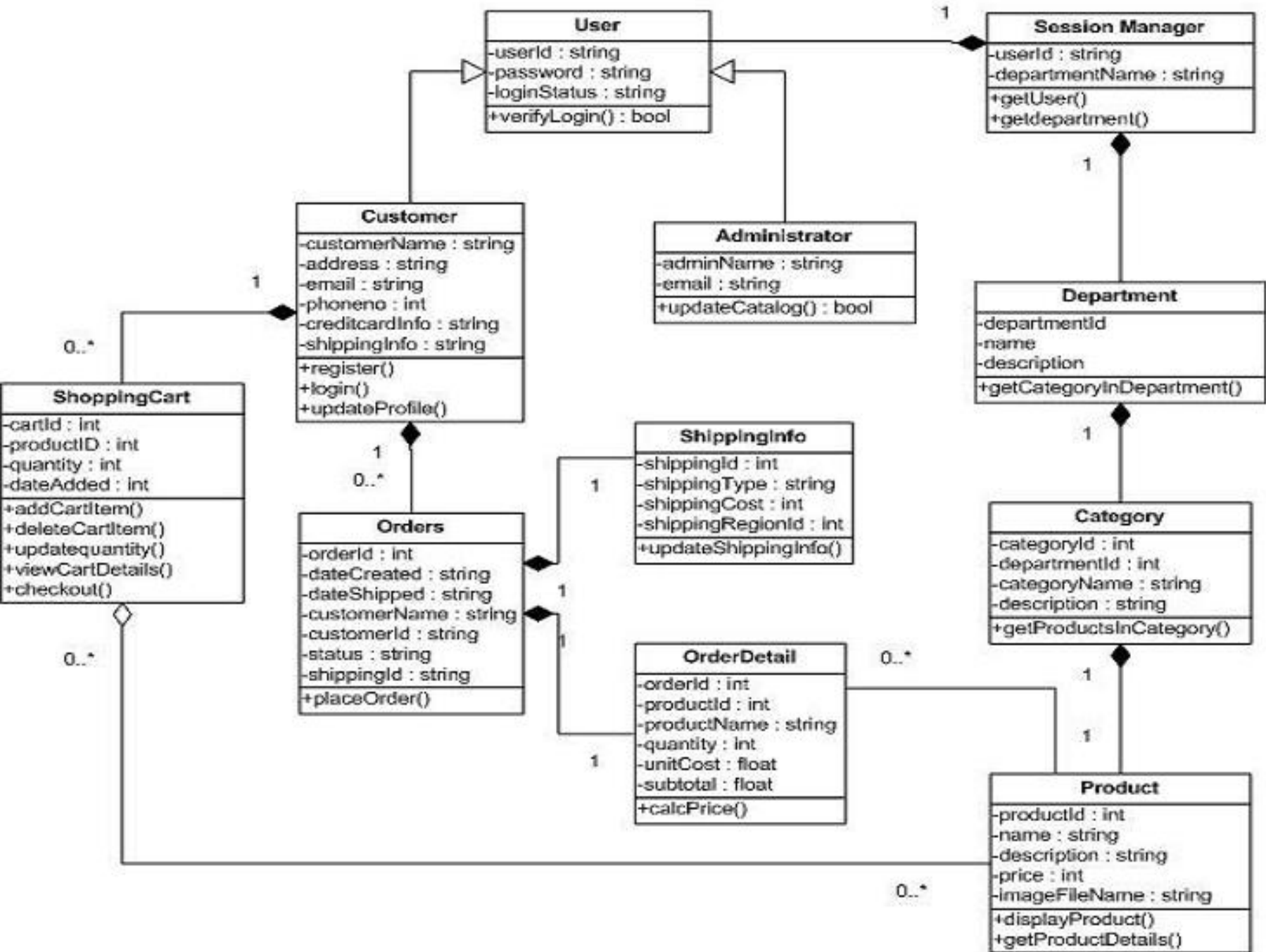


Contoh Aggregation dan Asosiasi



Contoh

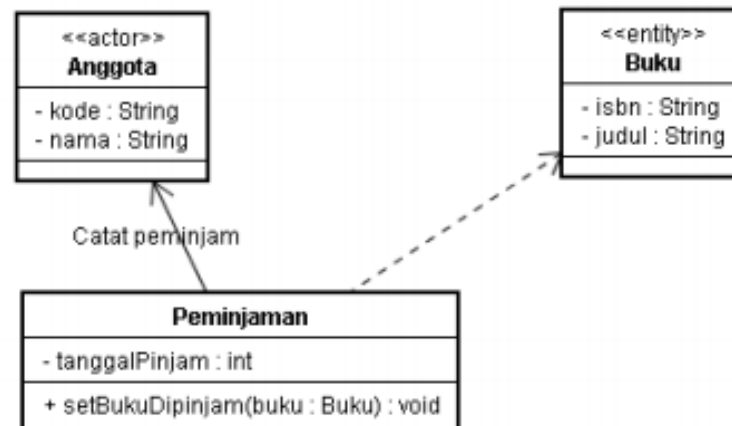




Relasi dalam Class Diagram

Dependency

- ❑ Relasi jenis ini menunjukkan bahwa sebuah kelas mengacu kepada kelas lainnya. Oleh sebab itu perubahan pada kelas yang diacu akan sangat berpengaruh pada kelas yang mengacu.
- ❑ **Contoh:** apabila seorang anggota hendak meminjam buku maka ada sebuah kelas yang bertanggung jawab melakukan pencatatan peminjam. Kelas ini akan mengetahui anggota yang meminjam dan membuat daftar buku apa saja yang dipinjam oleh anggota tersebut.



Relasi dalam Class Diagram

Composition

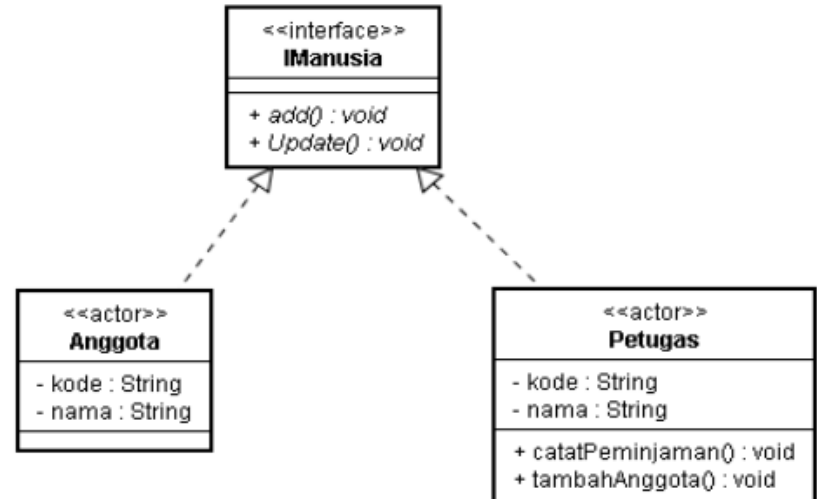
- ❑ Relasi ini merupakan relasi yang paling kuat dibandingkan dengan asosiasi dan agregasi. Pada komposisi diartikan bahwa suatu kelas merupakan bagian yang wajib dari kelas yang lain.
- ❑ Contoh kasus yaitu pada sebuah buku, sudah pasti terdapat halaman isi sekurang kurangnya satu.



Relasi dalam Class Diagram

Realization

- ❑ Merupakan suatu relasi yang menunjukkan penerapan terhadap suatu interface kepada sebuah Class. Relasi realisasi biasanya digunakan untuk mewajibkan suatukelas memiliki suatu Method yang sudah didefinisikan bentuk kerangkanya dalam suatu interface.
- ❑ Contohnya pada kelas petugas dan anggota, kedua kelas ini tentunya memiliki Method yang wajib dimiliki namun melakukan kegiatan yang berbeda seperti add dan update.



Sequence Diagram

- Sequence diagram menunjukkan perilaku dari objek di use case dengan menggambarkan urutan langkah untuk menyelesaikan sebuah task dalam sebuah timeline.
- Jenis class:
 1. **Boundary** Class:
 - Class yang berinteraksi dengan aktor langsung (user interface)
 - Form, input, UI ini masuk di sini
 2. **Control** Class:
 - Class yang berhubungan dengan pemrosesan, penghitungan, kalkulasi, komputasi, query, dst
 3. **Entity** Class:
 - Class yang berhubungan dengan data, penyimpanan data/file

Simbol Sequence Diagram

a. An Actor



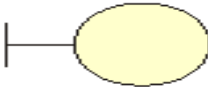
Menggambarkan orang yang sedang berinteraksi dengan sistem

b. Entity Class



Menggambarkan hubungan kegiatan yang akan dilakukan

c. Boundary Class



Menggambaran sebuah penggambaran dari form

d. Control Class



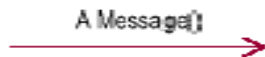
Menggambarkan penghubung antara boundary dengan tabel

e. A focus Of Control & A life line



Menggambarkan tempat mulai dan berakhirnya sebuah message

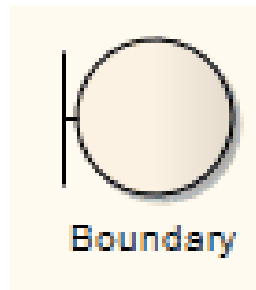
f. A message



Menggambarkan Pengiriman Pesan

Sequence Diagram

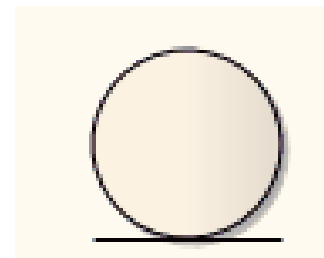
- ❑ Adalah class yang terletak antara sistem dengan dunia sekelilingnya. Semua form, laporan-laporan, antarmuka ke perangkat lunak seperti printer atau scanner, dan antarmuka ke sistem lainnya adalah termasuk kategori *boundary class*. UML menggambarkan class pembatas dengan gambar berikut:



- ❑ Boundary class adalah apa saja yang memungkinkan actor berinteraksi dengan sistem

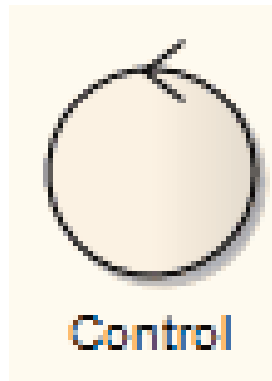
Sequence Diagram

- ❑ Menangani informasi yang disimpan dalam penyimpanan tetap.
- ❑ Perhatikan kata benda dalam aliran kejadian, beberapa kata benda akan menjadi entity class dalam sistem.
- ❑ Cara lain adalah dengan memperhatikan struktur basis data. Jika rancangan basis data telah dibuat, perhatikan nama-nama tabel. Tabel menangani record informasi secara permanen, sementara entity class menangani informasi didalam memori komputer.
- ❑ Notasi entity class adalah:

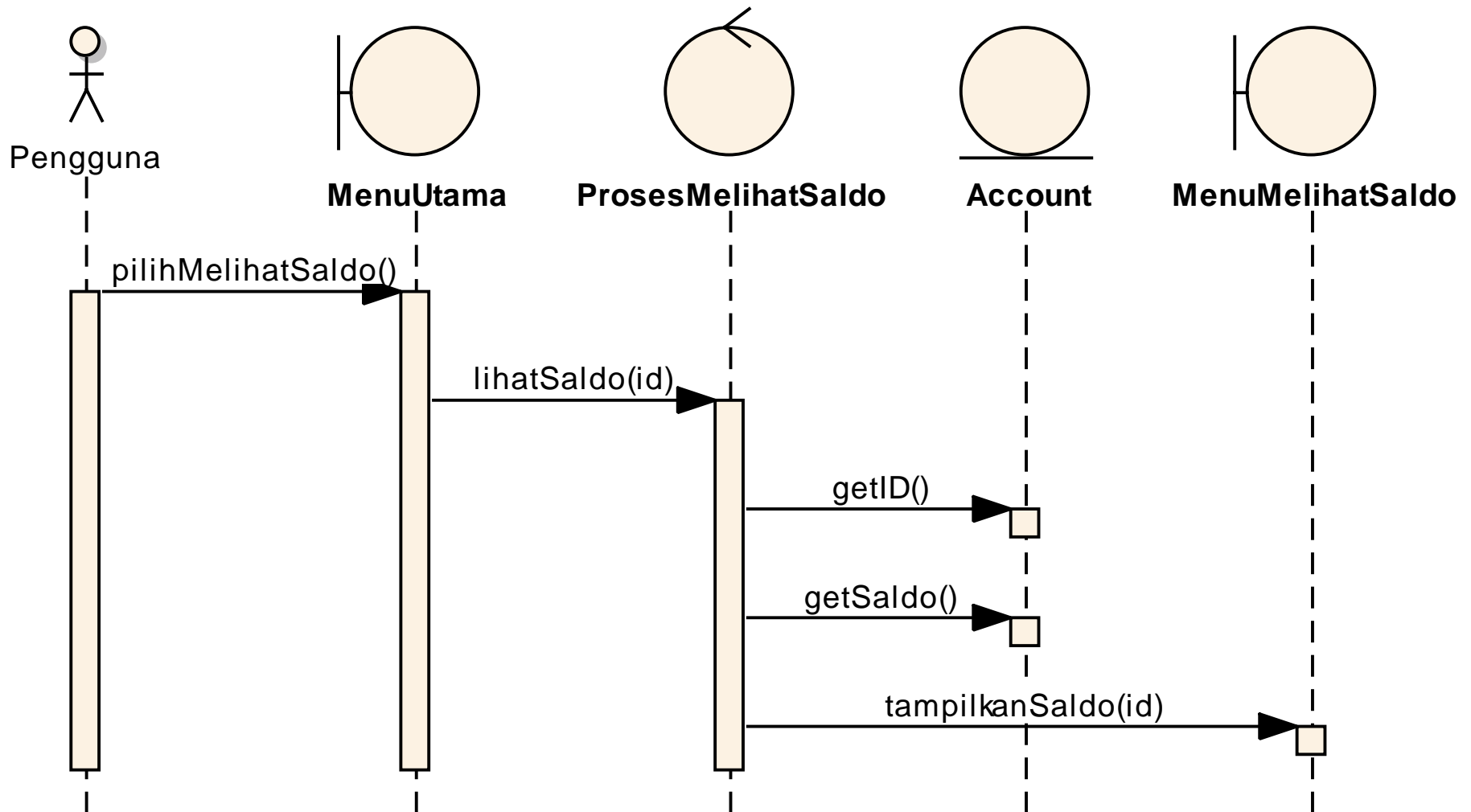


Sequence Diagram

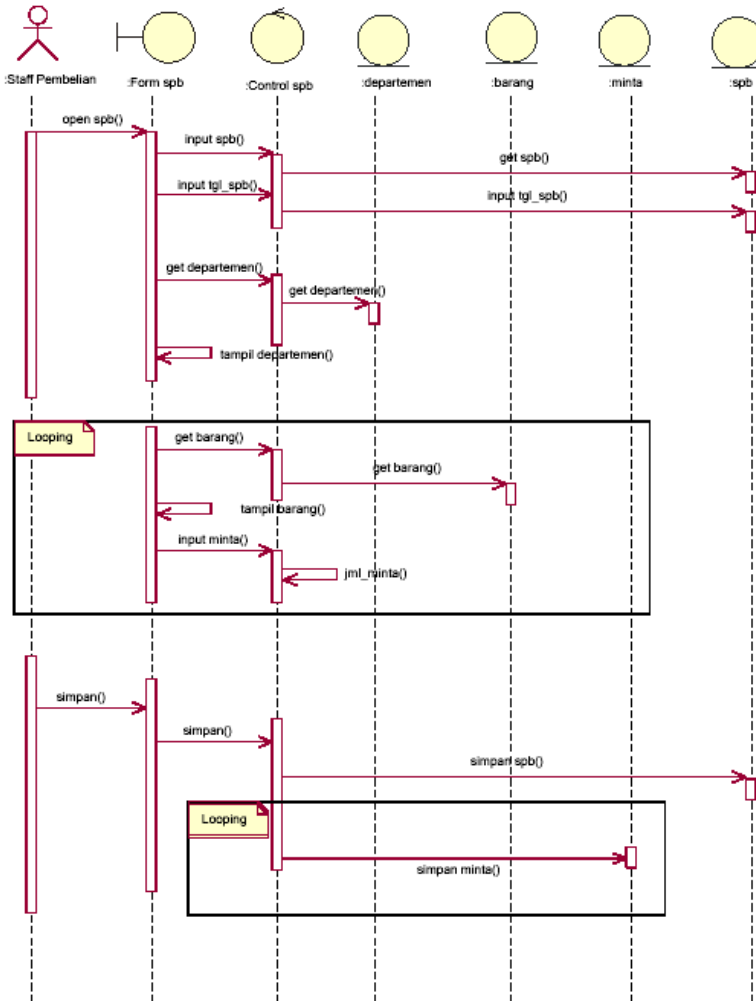
- ❑ Control class bertanggung jawab untuk mengkoordinasikan kegiatan-kegiatan terhadap class lainnya.
- ❑ Class ini bersifat opsional
- ❑ Notasi control class adalah:



Contoh Sequence Diagram



Contoh Sequence Diagram



Form spb

ASTRA CREDIT COMPANIES
Jl.Rs Patmawati No.9 Jakarta – Selatan 14240

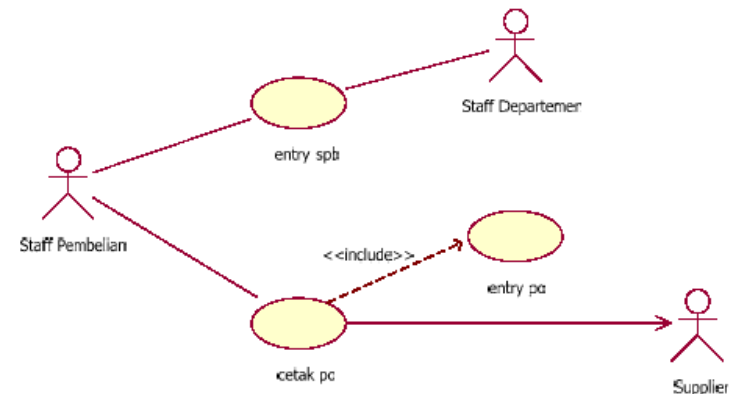
ENTRY SPB

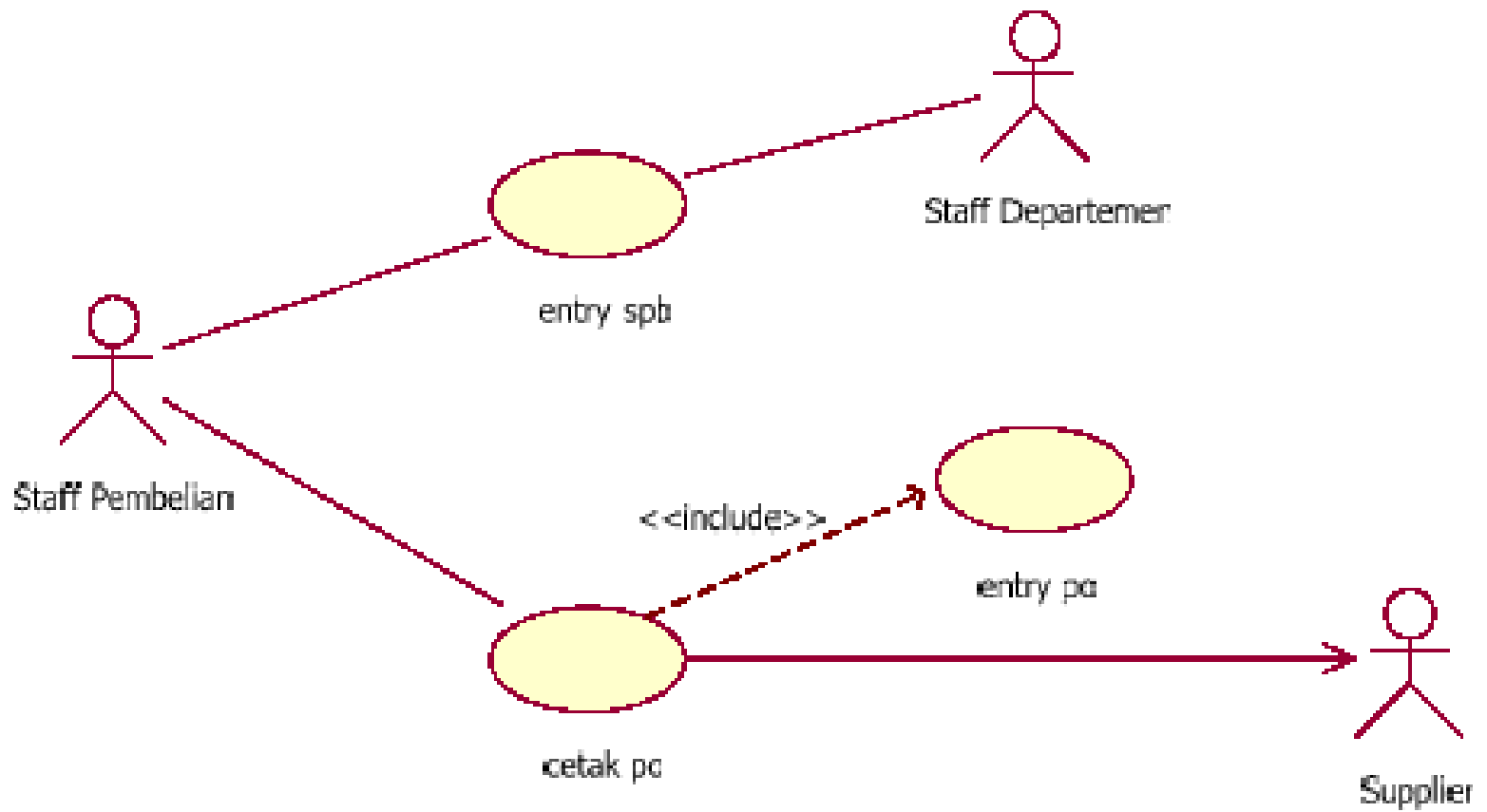
Spb:
 Nomor SPB Tanggal SPB

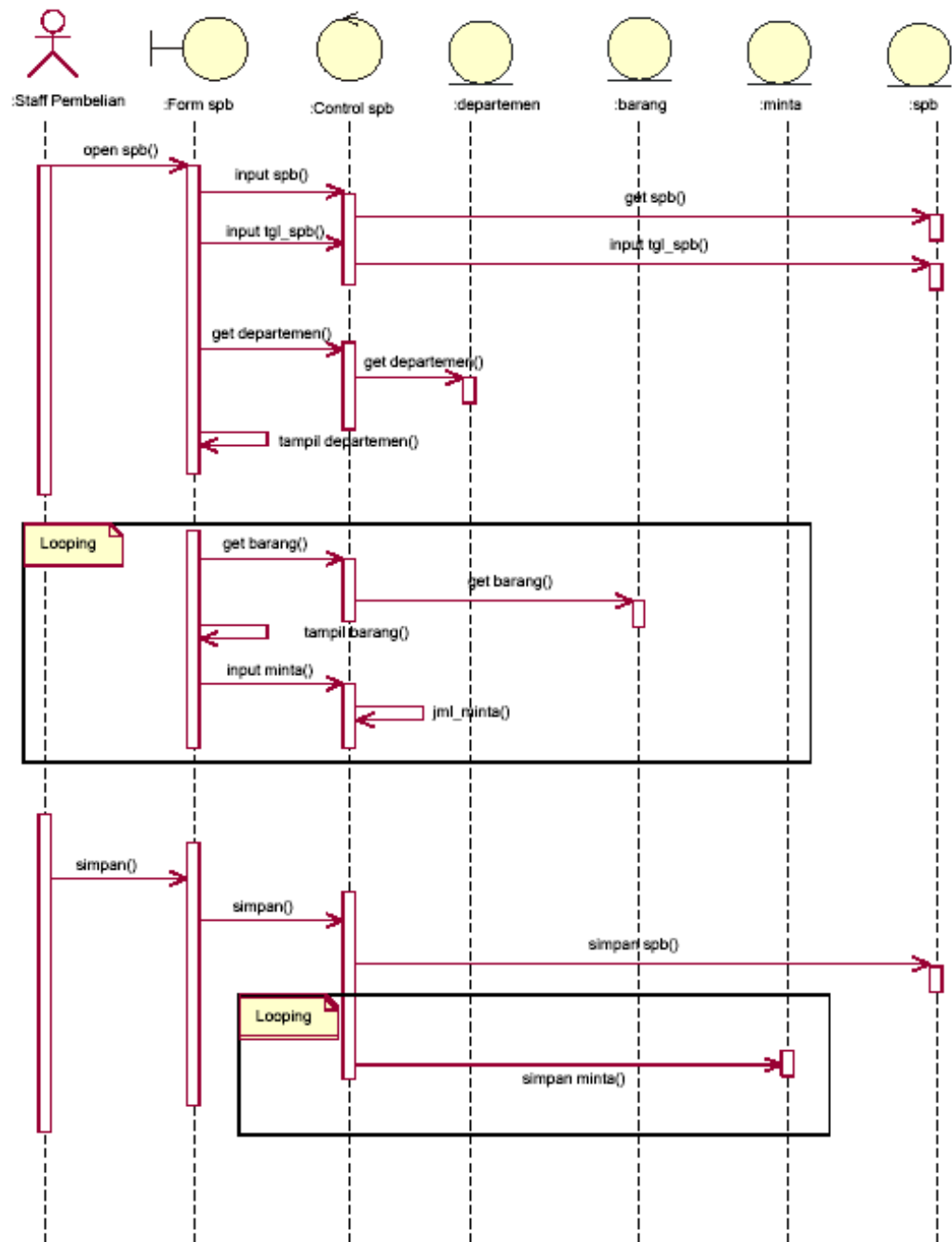
Departemen:
 Kode Departemen Nama Departemen

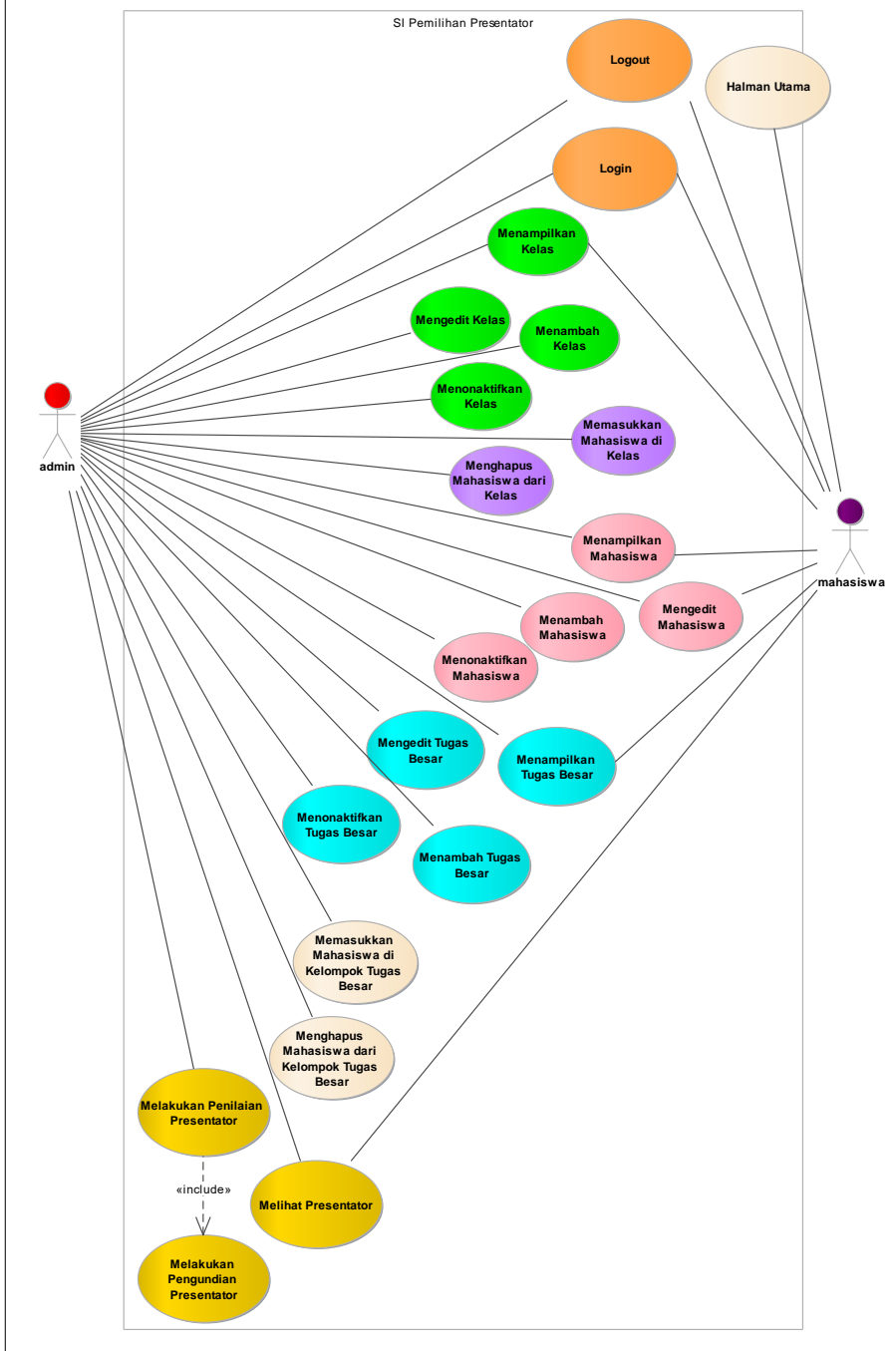
Barang:
 Kode Barang Satuan
 Nama Barang Jumlah Minta

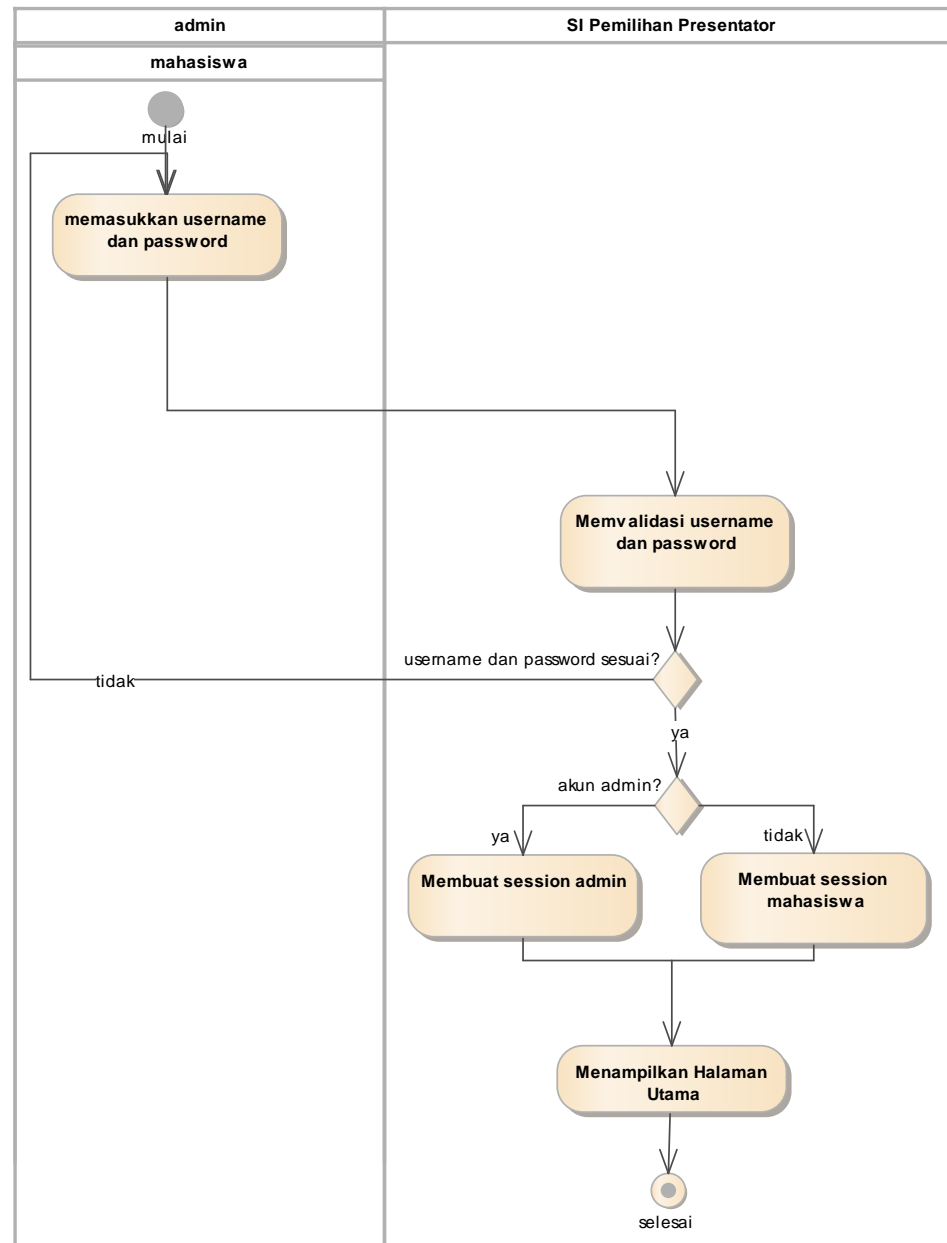
Kode Barang	Nama Barang	Satuan	Jumlah Minta
[tampil]	[tampil]	[tampil]	[tampil]

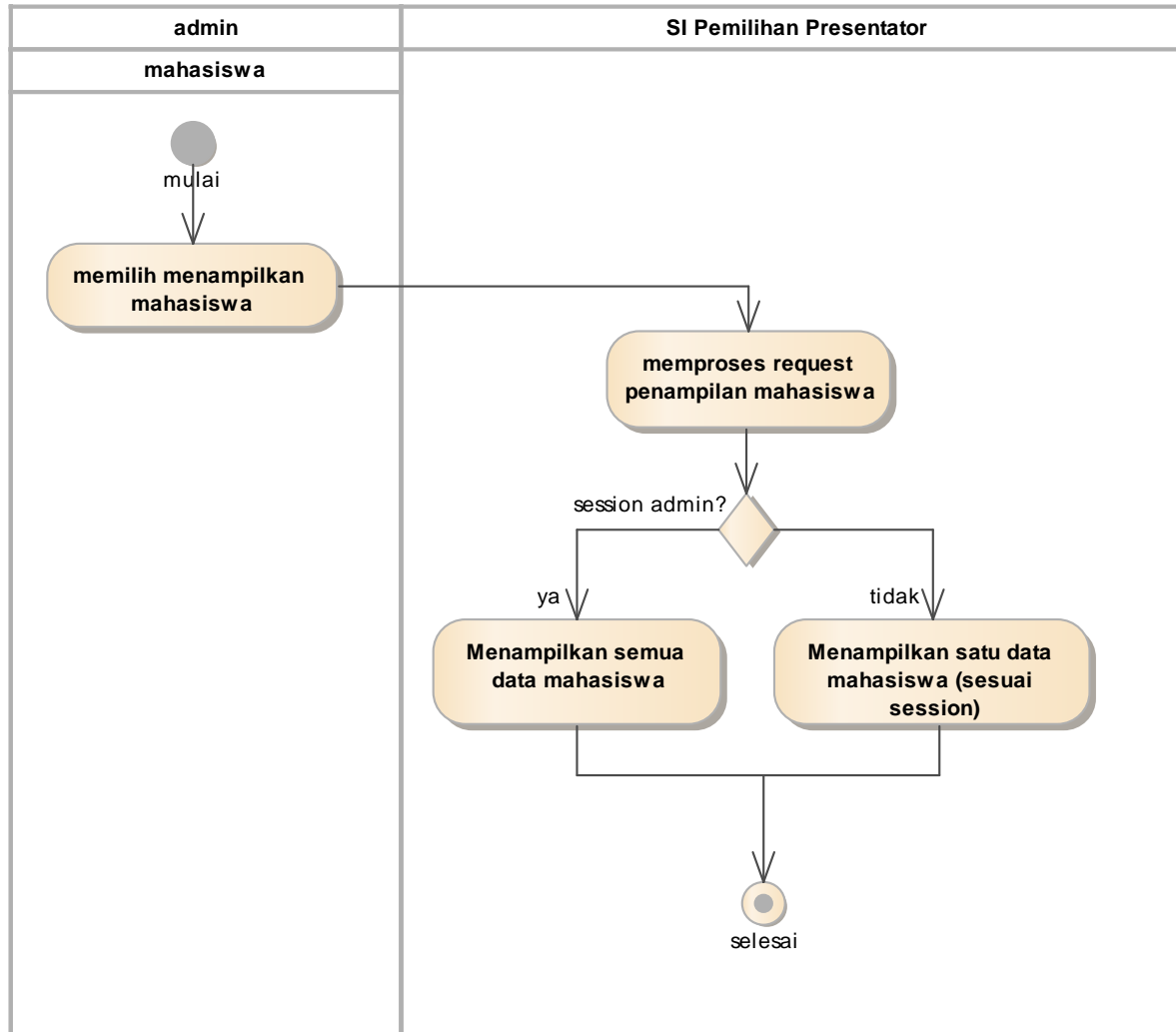


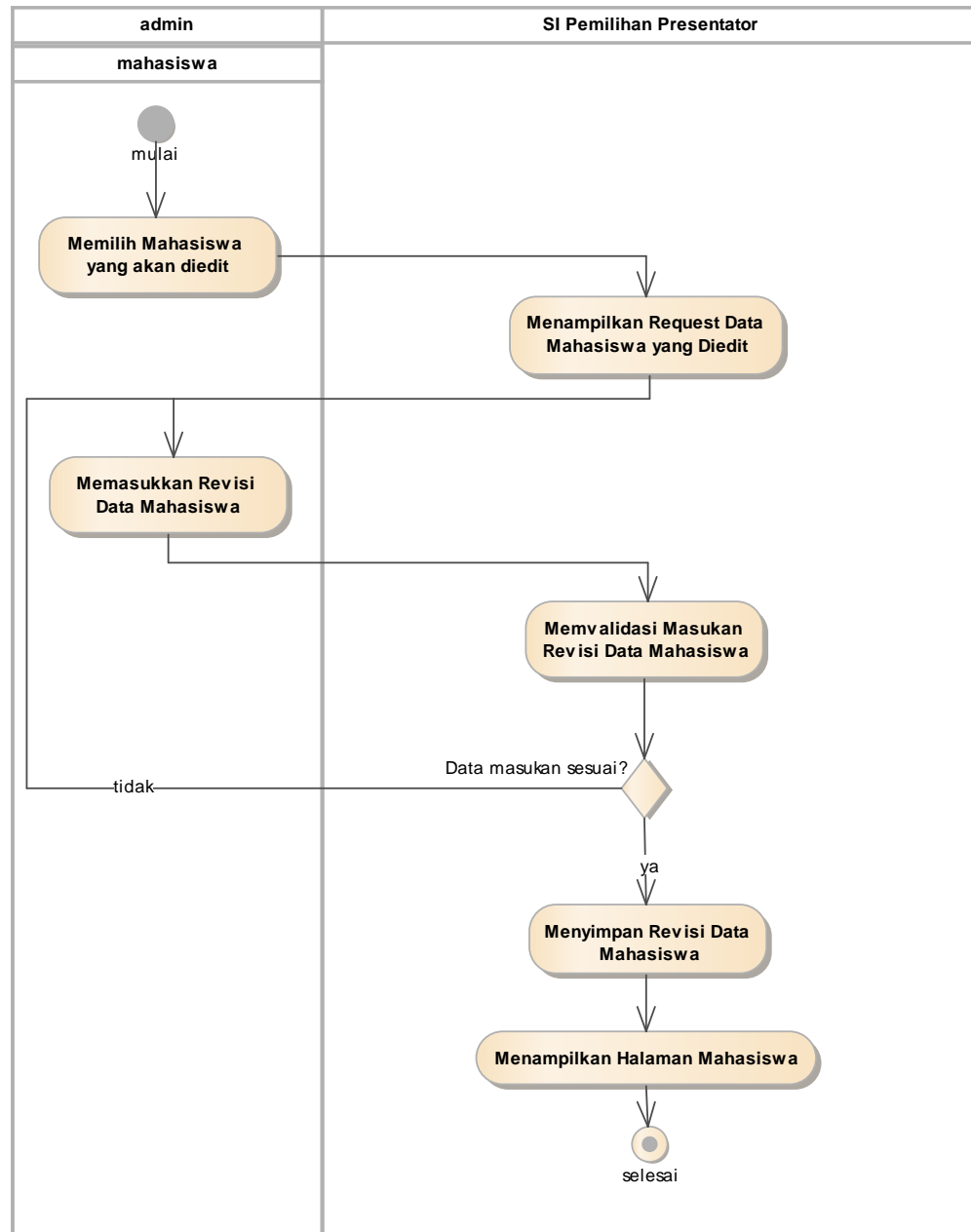


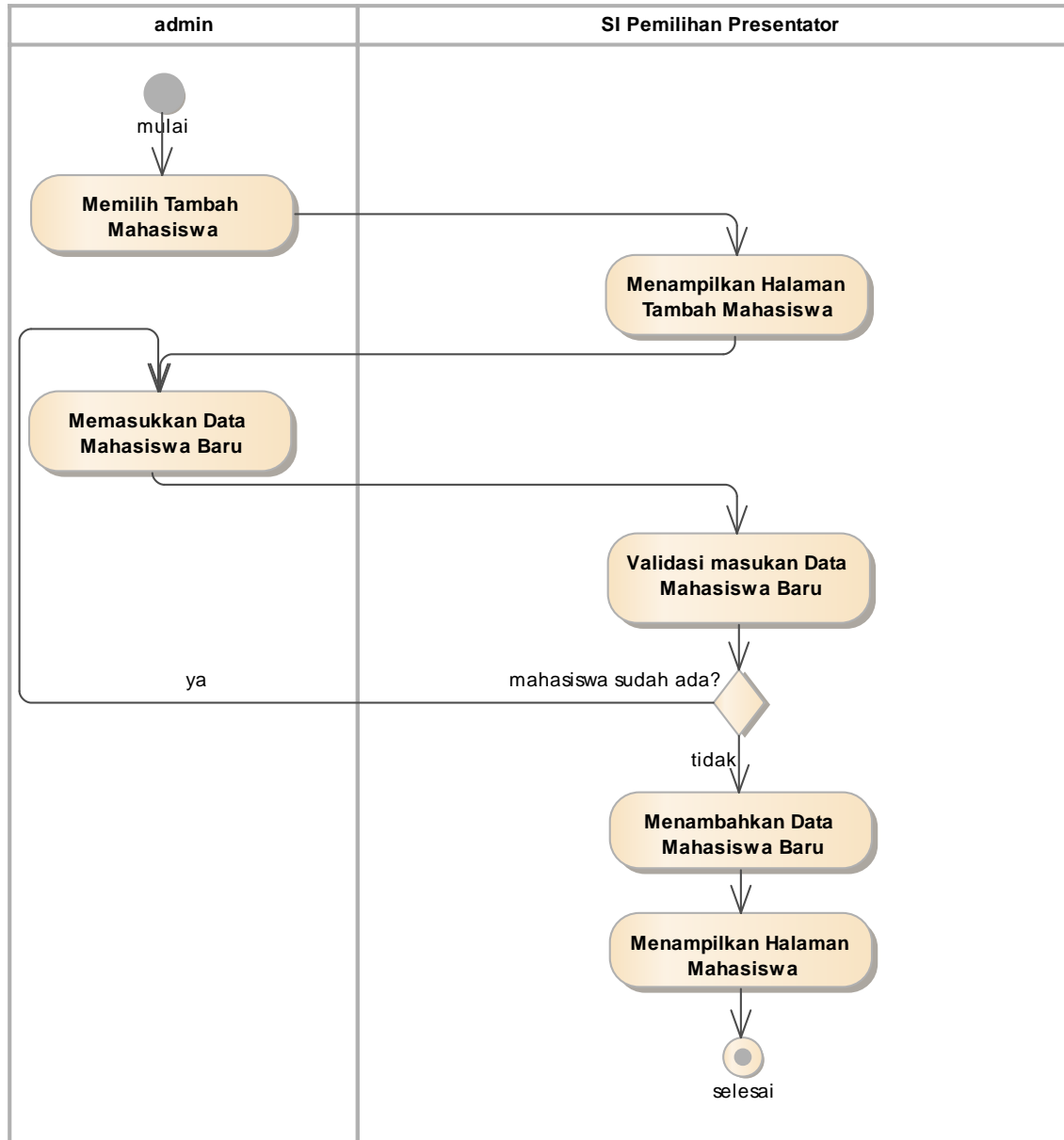




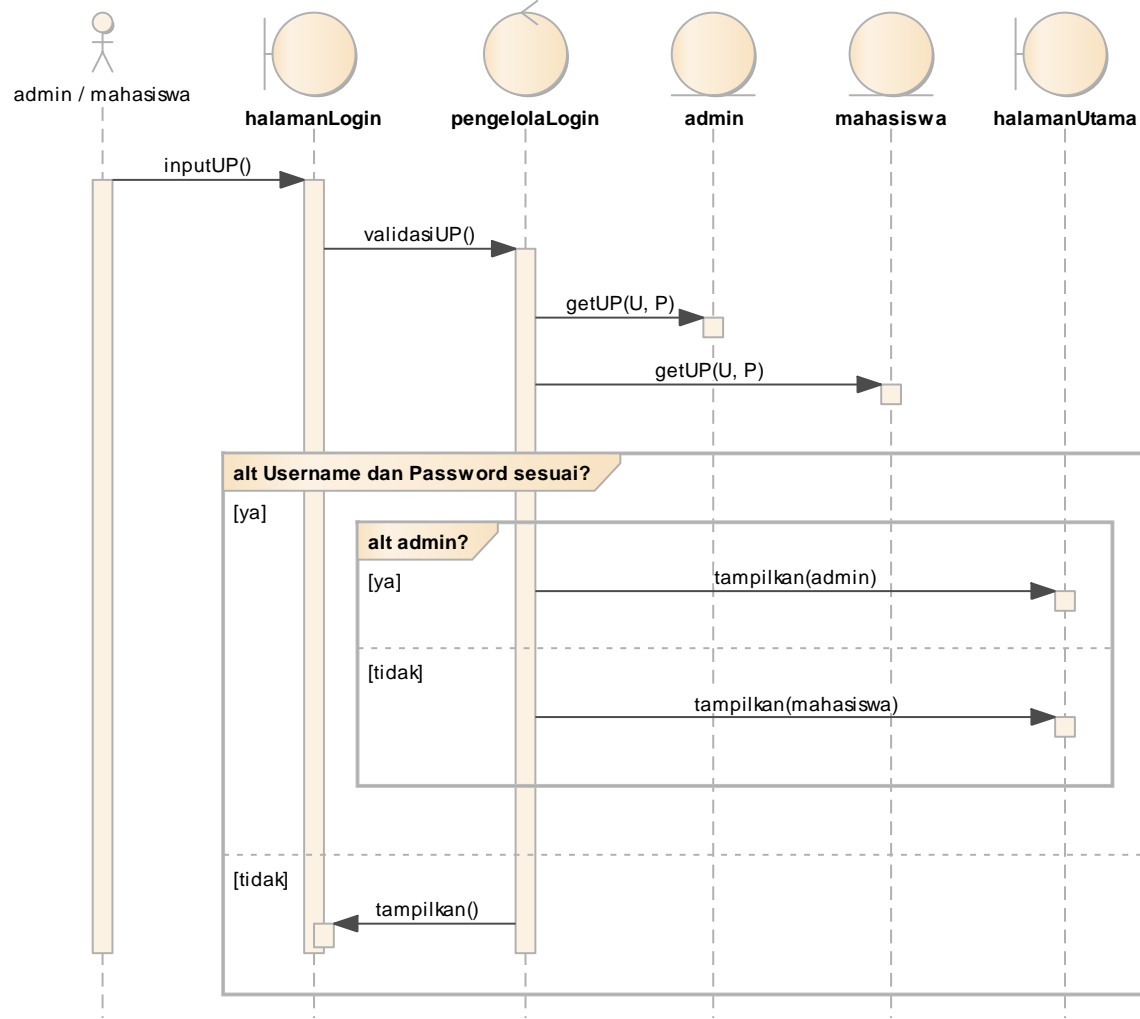




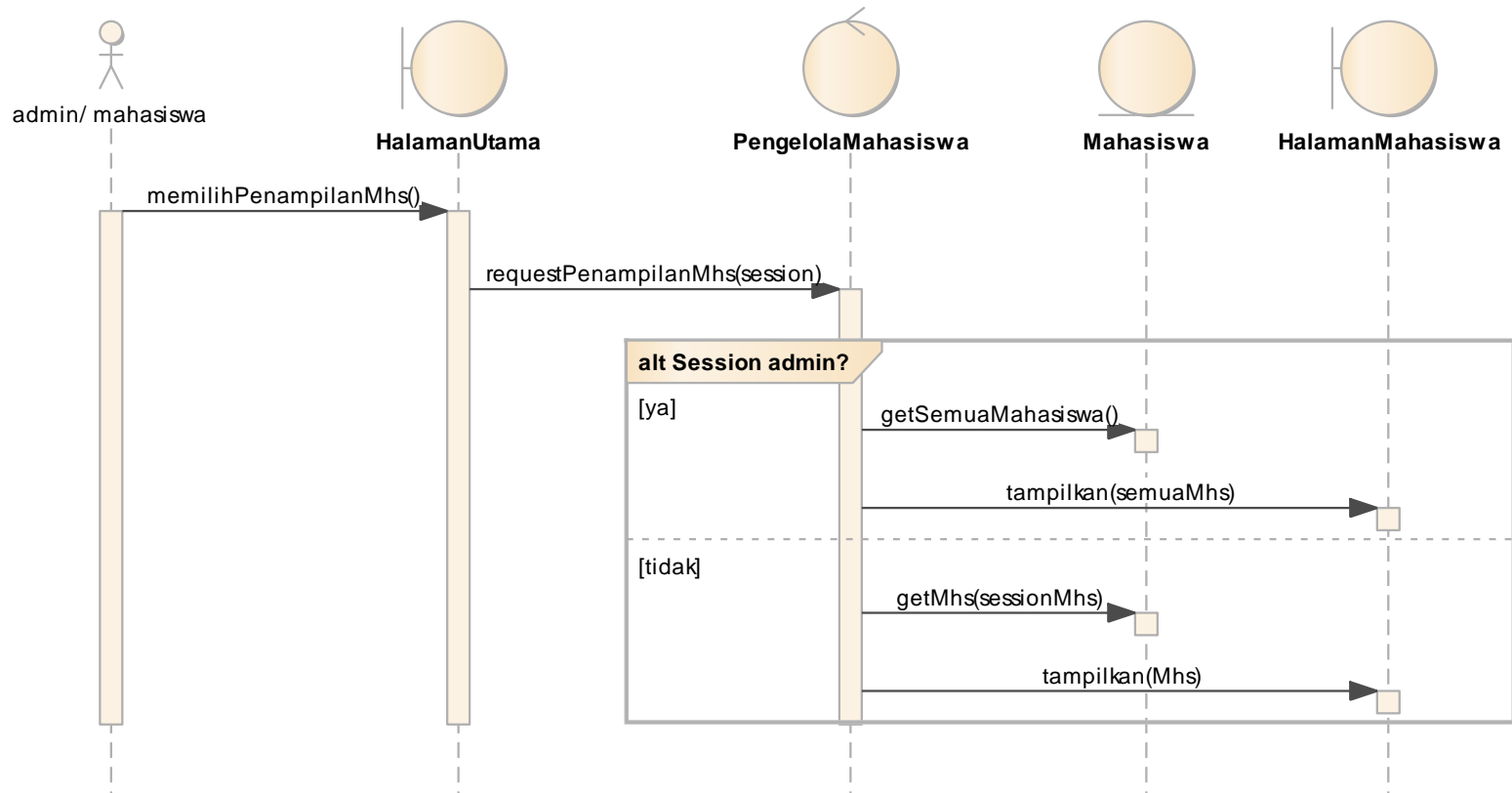




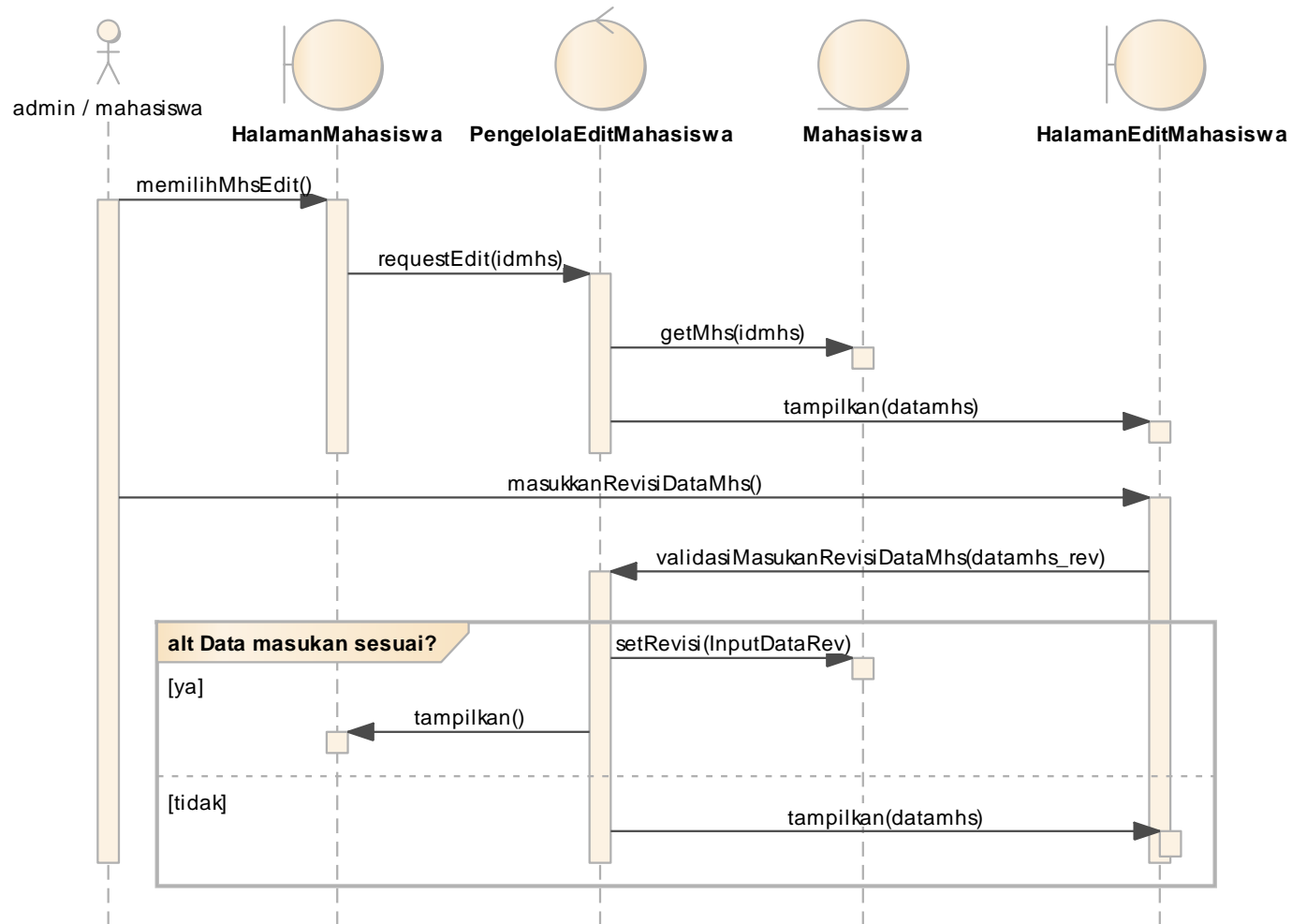
sd 03.1. SD Login



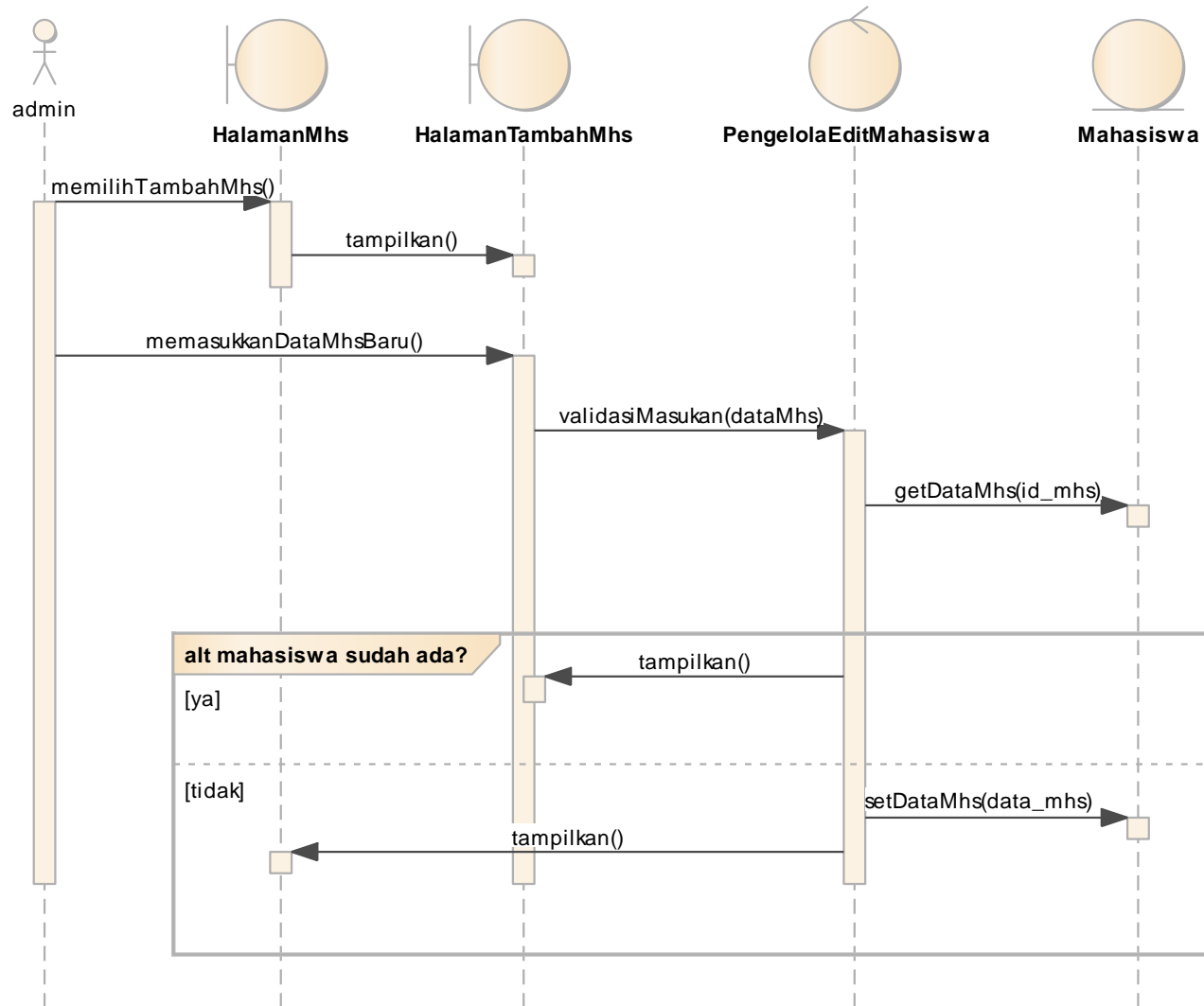
sd 03.02. SD Menampilkan Mahasiswa



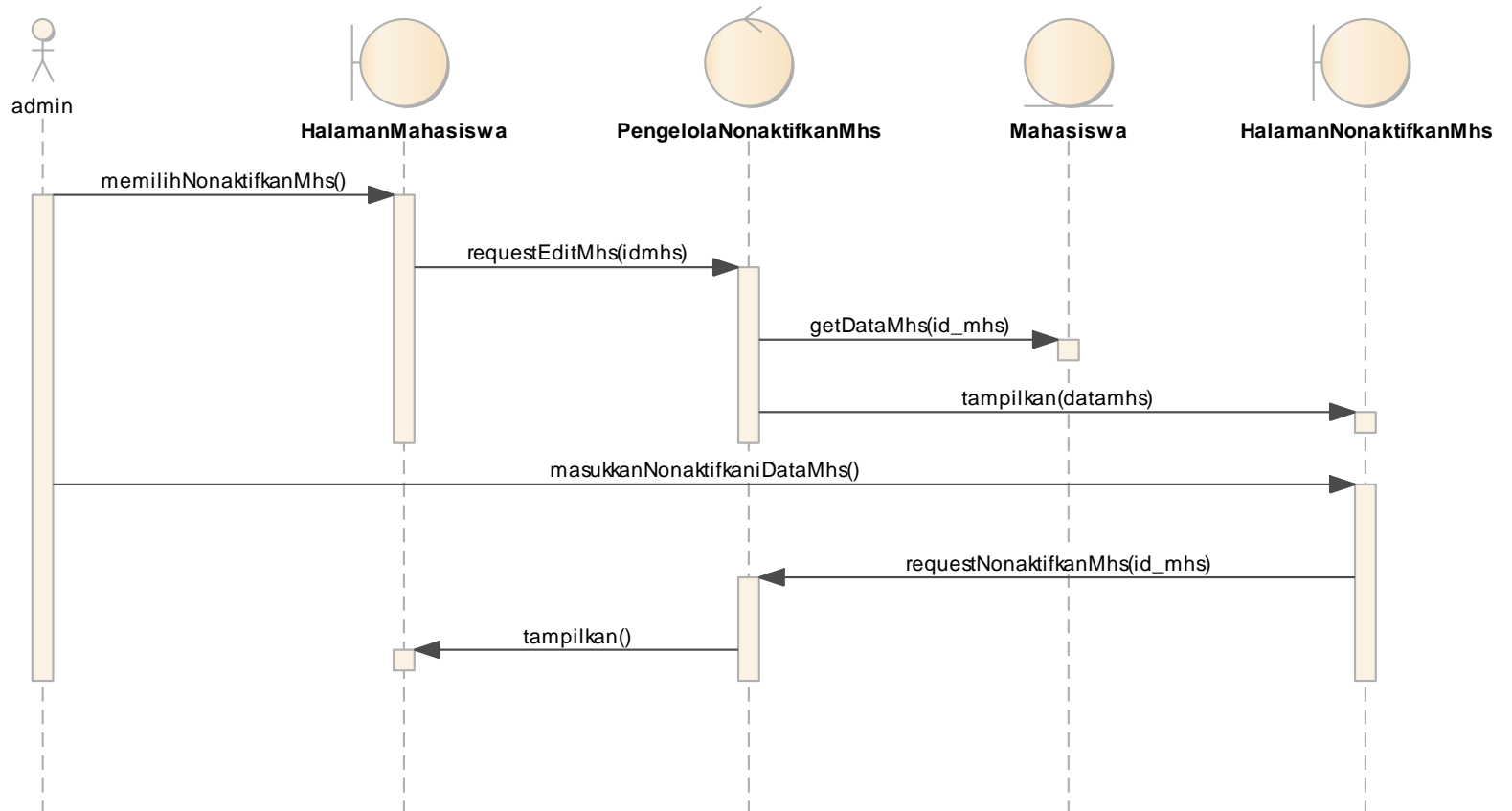
sd 03.03. SD Mengedit Mahasiswa



sd 03.04. SD Menambah Mahasiswa



sd 03.05. SD Menonaktifkan Mahasiswa



Tugas

- ❑ Buatlah **Pemodelan menggunakan UML** sesuai dengan project yang akan dibangun
- ❑ Mulai dari :
 - Use Case
 - Activity Diagram
 - Sequence Diagram

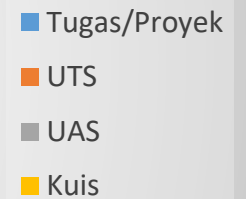
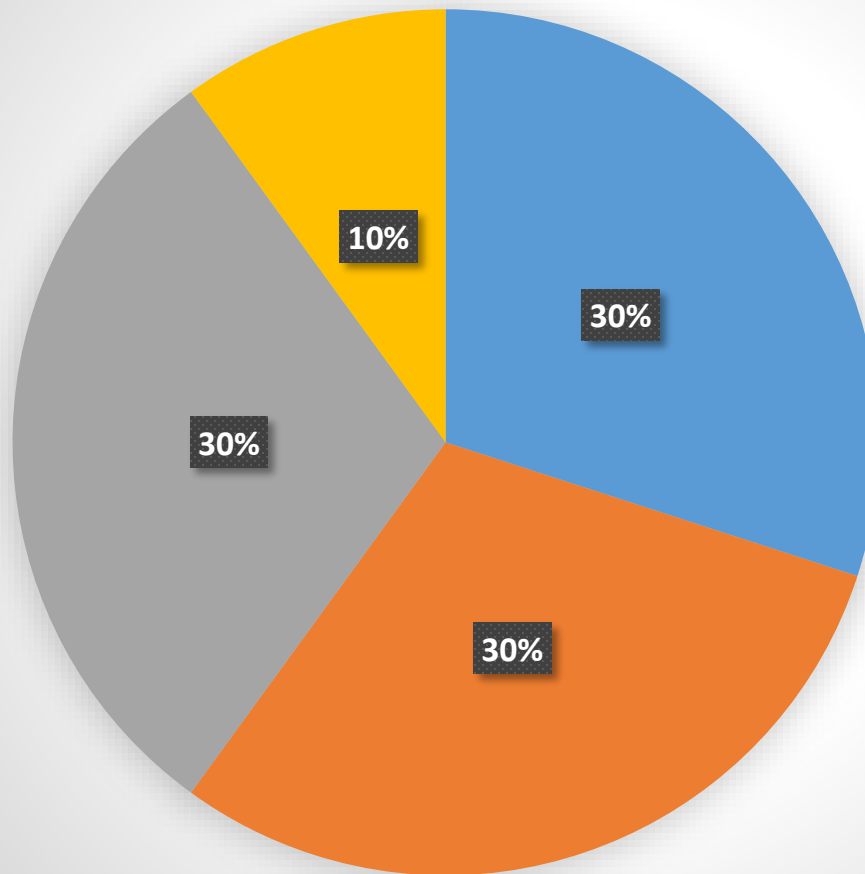
2) Kontrak Perkuliahan

c). Metode Penilaian

d). Tugas dan Proyek

Metode Penilaian ADSI SI2C

Bobot



Tata Tertib Perkuliahan

- ❑ Masuk sesuai jadwal Jam 09.30 WIB, Toleransi keterlambatan adalah 15 menit.
- ❑ Pakaian bebas rapi berkerah, bersepatu.
- ❑ Segala macam bentuk ijin ketidakhadiran diharuskan dengan alasan yang jelas
- ❑ Setiap mahasiswa dilarang mencontek dalam pengerjaan tugas dan ujian, jika terjadi maka pengerjaan tugas dan ujian akan dikurangi 20% atau Gugur.
- ❑ Setiap mahasiswa dilarang melakukan tindakan plagiat atas pengerjaan tugasnya, jika terjadi maka pengerjaan tugas akan dikurangi 20% atau Gugur.
- ❑ Setiap mahasiswa wajib mengerjakan ujian dan tugas baik tugas mandiri ataupun berkelompok.
- ❑ Wajib untuk bertutur kata yang sopan dan santun didalam kelas dan berpakaian rapih dan sopan

2) Kebutuhan Software



Kebutuhan Software

Editor

- Ms. Office
- LibreOffice
-

Pemodelan Tools

- UML : **Sparx Enterprise Architect**, Star UML
- Flowchart-DFD : **Visio**
- ERD : **Ms Access**
-

3) Contact



Contact

- ❑ Bahan Kuliah : github.com/doniaft
- ❑ Email : doniaft@gmail.com
- ❑ WA/Telegram :
- ❑ Komting ADSI SI2C : Taufiq R : 0853 3036 2686

4) Referensi

Referensi (1)

- ❑ Wendy Boggs & Micheal Boggs, “UML with Rational Rose 2003”, Sybex 2002
- ❑ Ian Sommerville, [Software Engineering 10th Edition](#), Addison-Wesley, 2015
- ❑ Roger S. Pressman, [Software Engineering: A Practitioner’s Approach 8th Edition](#), McGraw-Hill, 2014
- ❑ P. Bourque and R.E. Fairley, eds., [Guide to the Software Engineering Body of Knowledge Version 3.0](#), IEEE Computer Society, 2014
- ❑ Albert Endres dan Dieter Rombach, [A Handbook of Software and Systems Engineering](#), Pearson Education Limited, 2003
- ❑ Yingxu Wang, [Software Engineering Foundations: A Software Science Perspective](#), Auerbach Publications, Taylor & Francis Group, 2008
- ❑ Alan Dennis et al, [Systems Analysis and Design with UML – 4th Edition](#), John Wiley and Sons, 2012
- ❑ Dan Pilone and Russ Miles, [Head First Software Development](#), O’Reilly Media, 2008
- ❑ Barclay and Savage, [Object-Oriented Design with UML and Java](#), Elsevier, 2004
- ❑ Kenneth E. Kendall and Julie E Kendall, [Systems Analysis and Design 8th Edition](#), Prentice Hall, 2010
- ❑ Hassan Gomaa, [Software Modeling and Design: UML, Use Cases, Patterns, and Software Architectures](#), Cambridge University Press, 2011
- ❑ Layna Fischer (edt.), [BPMN 2.0 Handbook Second Edition](#), Future Strategies, 2012
- ❑ Daniel Galin, [Software Quality Assurance](#), Addison-Wesley, 2004
- ❑ Kshirasagar Naik and Priyadarshi Tripathy, [Software Testing and Quality Assurance](#), John Wiley & Sons, Inc., 2008
- ❑ Jeff Tian, [Software Quality Engineering](#), John Wiley & Sons, Inc., 2005
- ❑ G. Gordon Schulmeyer, [Handbook of Software Quality Assurance Fourth Edition](#), Artech House, 2008
- ❑ Daniel Galin, [Software Quality Assurance](#), Addison-Wesley, 2004
- ❑ Kshirasagar Naik and Priyadarshi Tripathy, [Software Testing and Quality Assurance](#), John Wiley & Sons, Inc., 2008
- ❑ Jeff Tian, [Software Quality Engineering](#), John Wiley & Sons, Inc., 2005
- ❑ G. Gordon Schulmeyer, [Handbook of Software Quality Assurance Fourth Edition](#), Artech House, 2008