

## **Pertemuan 5**

# **PEMODELAN SISTEM dengan UML (bagian 1)**

# 1. *Unified Modelling Language*

- Menurut Booch, et.al. UML adalah bahasa standar untuk menulis *blue print* PL. UML dapat digunakan untuk memvisualisasikan, menentukan, membuat, dan mendokumentasikan artefak dari sistem PL secara intensif.
- UML sesuai untuk sistem pemodelan mulai dari sistem informasi perusahaan, aplikasi berbasis web yang terdistribusi, bahkan sampai sistem *real time embedded* yang sulit.
- UML adalah proses yang independen, walaupun secara optimal harus digunakan dalam proses yang menggunakan *case driven, architecture-centric, iterative, dan incremental*.

# UML adalah bahasa untuk:

## **a. *Visualizing***

Beberapa hal dimodelkan secara tekstual atau dengan model grafis. UML adalah bahasa grafis yang menggunakan sekelompok simbol grafis. Setiap simbol dalam notasi UML didefinisikan dengan baik secara semantik, sehingga pengembang dapat menulis model UML dan dapat menafsirkan model itu dengan jelas.

## **b. *Specifying***

UML dapat membangun model yang tepat, tidak ambigu, dan lengkap. UML membahas spesifikasi semua keputusan analisis, perancangan, dan implementasi penting yang harus dilakukan dalam mengembangkan dan menerapkan sistem PL yang intensif.

## UML adalah bahasa untuk (lanjutan)

### ***c. Constructing***

UML bukan bahasa pemrograman visual, namun modelnya bisa langsung terhubung ke berbagai bahasa pemrograman, dan memungkinkan untuk memetakan ke bahasa pemrograman seperti Java, C ++, atau Visual Basic, atau bahkan ke tabel dalam basis data relasional atau penyimpanan database berorientasi objek yang tetap.

### ***d. Documenting***

UML membahas dokumentasi arsitektur sistem dan semua detailnya. UML menyediakan bahasa untuk mengekspresikan persyaratan dan tes. UML juga menyediakan bahasa untuk memodelkan kegiatan perencanaan proyek.

# UML versi 1.0 dibagi menjadi 2 kelompok:

## A. Diagram Struktur (*Structural Diagrams*)

1. Class Diagram
2. Object Diagram
3. Component Diagram
4. Deployment Diagram

## B. *Behavioral Diagrams*

1. Use Case Diagram
2. Sequence Diagram
3. Activity Diagram
4. Statechart Diagram
5. Collaboration Diagram

# Diagram-Diagram dalam UML

## 1. *Class Diagram*

Menunjukkan seperangkat kelas, antarmuka, dan kolaborasi dan hubungan di antara mereka. Class Diagram membahas desain statis dari suatu sistem.

## 2. *Object Diagram*

Menunjukkan satu set objek dan hubungan antara objek. Diagram objek memodelkan *instance* dari hal-hal yang terdapat dalam *Class Diagram*. Diagram objek digunakan untuk memodelkan desain statis suatu sistem, untuk memvisualisasikan, menentukan, dan mendokumentasikan model struktural, dan membangun aspek statis sistem melalui teknik maju (*forward*) dan mundur (*reverse*).

## Diagram-Diagram dalam UML (Lanjutan)

### 3. ***Component Diagram***

Menunjukkan organisasi dan ketergantungan antar sekumpulan komponen.

### 4. ***Deployment Diagram***

Menunjukkan konfigurasi komponen dalam proses eksekusi aplikasi. Diagram ini terdiri dari node yang merupakan perangkat keras dan membungkus satu atau lebih komponen.

### 5. ***Use Case Diagram***

Menunjukkan sebuah interaksi antara satu atau lebih aktor dengan sistem informasi yang akan dibuat. Diagram ini sangat penting dalam mengatur dan memodelkan perilaku suatu sistem.

# Diagram-Diagram dalam UML (Lanjutan)

## 6. *Sequence Diagram*

Diagram interaksi yang menekankan urutan waktu pada pesan. Menempatkan objek yang berpartisipasi dalam interaksi pada sumbu X dan menempatkan pesan antar objek sepanjang sumbu Y, sedangkan waktu digambarkan dari atas ke bawah.

## 7. *Activity Diagram*

Diagram yang menunjukkan arus dari aktivitas ke aktivitas dalam suatu sistem. *Activity Diagram* membahas pandangan dinamis suatu sistem, dan sangat penting dalam pemodelan fungsi suatu sistem dan menekankan aliran kontrol antar objek.



## Diagram-Diagram dalam UML (Lanjutan)

### 8. *Statechart Diagram*

Menggambarkan perubahan status atau transisi status dari sebuah mesin atau sistem atau objek. *Statechart* sangat penting dalam memodelkan perilaku antarmuka, kelas, atau kolaborasi dan menekankan perilaku dari suatu objek, yang sangat berguna dalam memodelkan sistem reaktif.

### 9. *Collaboration Diagram*

Diagram interaksi yang menekankan struktur organisasi objek yang mengirim dan menerima pesan. Diagram ini merupakan perluasan dari diagram objek, yaitu memberikan tambahan asosiasi antara objek, dan menunjukkan objek mengirimkan *message* ke objek-objek yang lain.

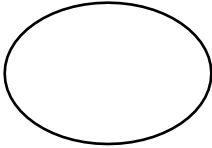
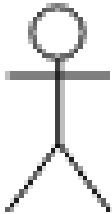
## 2. USE CASE DIAGRAM

- Use Case Diagram digunakan untuk menggambarkan serangkaian tindakan (*use cases*) bahwa sistem dapat melakukan interaksi di luar sistem (aktor) dengan sistem itu sendiri (abstraksi).
- Use Case juga digunakan untuk mengetahui fungsi apa saja yang ada dalam sebuah sistem dan siapa yang berhak menggunakan fungsi-fungsi itu.
- Nama Use Case didefinisikan semudah mungkin dan dapat dipahami.
- Dua hal utama pada Use Case yaitu pendefinisian aktor dan use case.



## USE CASE DIAGRAM (Lanjutan)

- Use case digunakan untuk
  - a. Merepresentasikan interaksi sistem – pengguna
  - b. Mendefinisikan dan mengatur persyaratan fungsional dalam suatu sistem
  - c. Menentukan konteks dan persyaratan sistem
  - d. Memodelkan aliran dasar peristiwa dalam use case

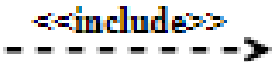

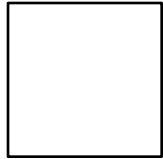
# Simbol-Simbol Use Case

No	Nama	Simbol	Keterangan
1.	Use Case		<ul style="list-style-type: none"> <li>Digambarkan dengan elips horizontal</li> <li>Nama Use case menggunakan <b>kata kerja</b></li> </ul>
2.	Aktor		<ul style="list-style-type: none"> <li>Menggambarkan orang, system/external entitas yang menyediakan atau menerima informasi</li> <li>Merupakan lingkungan luar dari sistem</li> <li>Nama Aktor menggunakan <b>Kata benda</b></li> <li>Aktor <b>utama</b> digambarkan pada pojok kiri atas dari diagram</li> </ul>

# Simbol-Simbol Use Case

No	Nama	Simbol	Keterangan
3.	Asosiasi		<ul style="list-style-type: none"> <li>Menggambarkan <b>bagaimana aktor</b> berinteraksi dengan use case</li> <li><b>Bukan</b> menggambarkan aliran data/informasi</li> </ul>
4.	Generalisasi		<ul style="list-style-type: none"> <li>Gambarkan generalisasi antara use case atau antara aktor dengan panah tertutup yang mengarah dari <i>child</i> ke <i>parent</i></li> </ul>

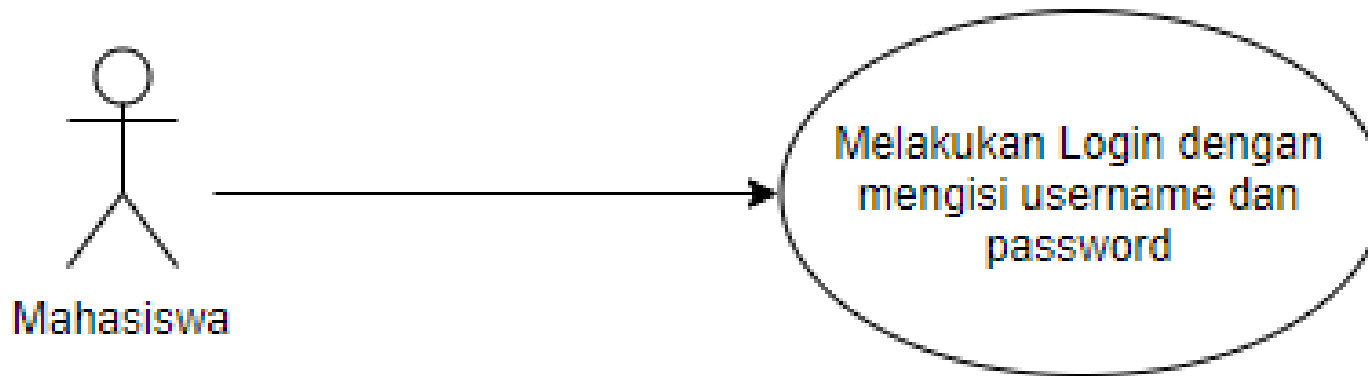
# Simbol-Symbol Use Case

No	Nama	Simbol	Keterangan
5.	Relasi <b>include</b>		<ul style="list-style-type: none"> <li>Hubungan antara dua use case untuk menunjukkan adanya perilaku use case yang dimasukkan ke dalam perilaku dari <i>base use case</i></li> <li>Tanda panah terbuka harus terarah ke <b>sub use case</b></li> </ul>
	Relasi <b>extend</b>		<ul style="list-style-type: none"> <li>Perluasan dari use case lain (<i>optional</i>)</li> <li>Tanda panah terbuka harus terarah ke <b>base use case</b></li> </ul>
6.	Boundary Boxes		<ul style="list-style-type: none"> <li>Untuk memperlihatkan batasan sistem dengan lingkungan luar sistem</li> </ul>

## Contoh penggunaan relasi *include* dan *extend*



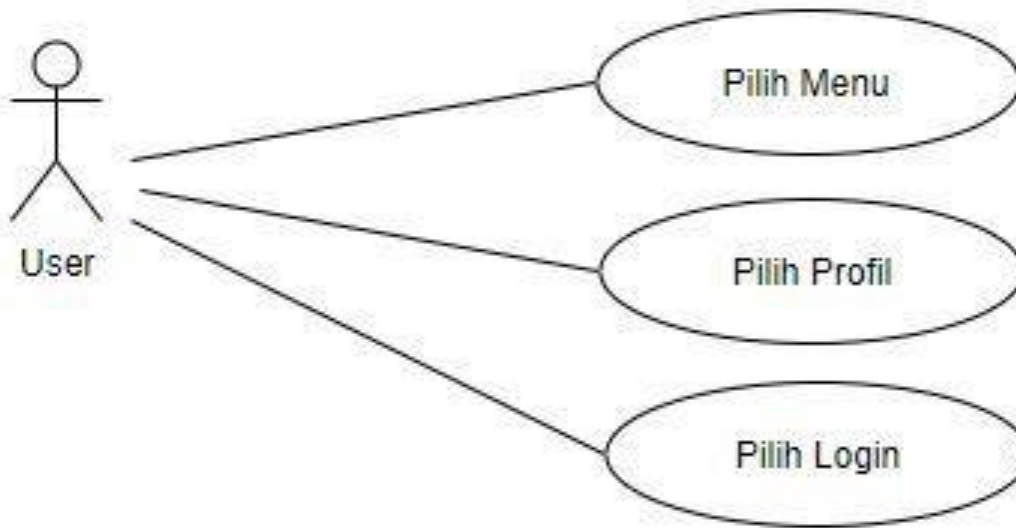
## Contoh Penggambaran Use Case (1)



- Pada gambar di atas, penulisan nama use case **tidak dituliskan secara berlebihan**, seharusnya hanya bisnis proses saja yang dituliskan yaitu LOGIN
- Asosiasi aktor dengan use case sebaiknya tidak menggunakan tanda panah, kecuali untuk penggunaan relasi include/extend, dan generalisasi



## Contoh Penggambaran Use Case (2)



Pada gambar di atas, penulisan nama use case **Pilih Menu**, **Pilih Profil**, **Pilih Login**, tidak seharusnya ditulis demikian. Use case dituliskan dengan bisnis proses, bukan aksi seperti dalam Activity Diagram

# Contoh

Berikut contoh Sistem Informasi  
Housekeeping Inventory

## 1.a. Daftar Kebutuhan Fungsional Use Case

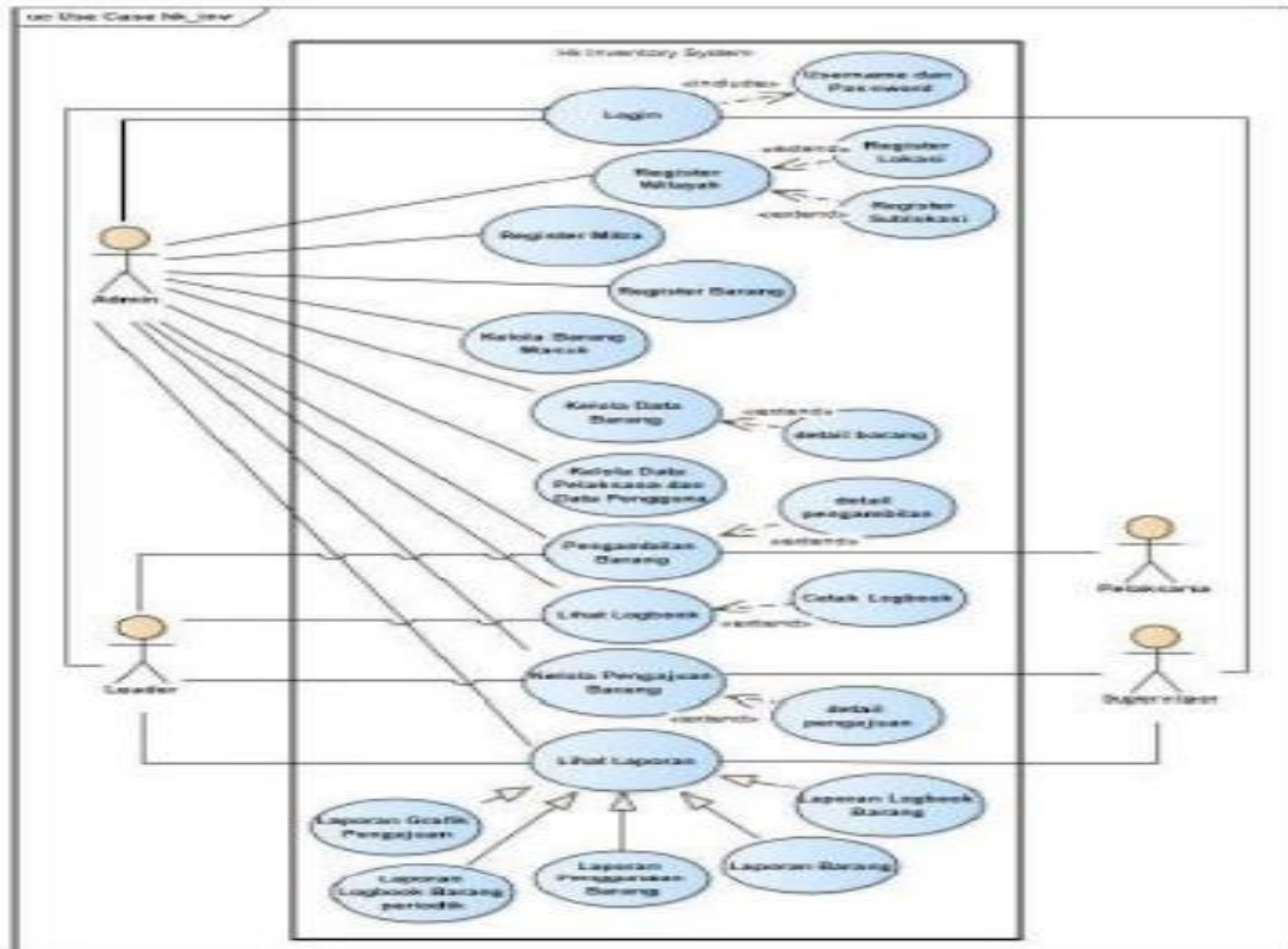
No	Use Case	Aktor	Deskripsi
1	Login	Admin, Pelaksana, Leader Supervisor	Use case ini berfungsi untuk masuk ke dalam sistem
2	Registrasi	Admin	Use case ini berfungsi untuk admin melakukan pendaftaran barang maupun pengguna
3	Kelola Barang	Admin, Pelaksana, Leader	Use case ini berfungsi untuk pelaksana melakukan pengelolaan barang dan leader melakukan persetujuan
4	Kelola Pengajuan Barang	Admin, Leader, Supervisor	Use case ini berfungsi untuk Leader melakukan pengajuan barang dan Supervisor menyetujui
5	Lihat Laporan	Admin, Leader, Supervisor	Use case ini berfungsi untuk Leader dan Superuser melihat laporan

## 1.b. Daftar Kebutuhan Fungsional Aktor

No	Aktor	Deskripsi
1	Admin	bertugas untuk mengelola seluruh data dan transaksi yang dilakukan oleh pelaksana.
2	Leader	bertugas mengawasi transaksi barang.
3	Supervisor	bertugas mengawasi jalannya sistem.
4	Pelaksana	Melakukan transaksi pada sistem

## 2. Use Case

# Sistem Informasi Housekeeping Inventory



### 3. Spesifikasi Use Case

Nama Use Case	Beli Tiket
Deskripsi	Use case ini menyediakan layanan pengelolaan inventory barang
Aktor	Admin
Pre-Condition	Mendaftarkan Pelaksana, Leader, dan Supervisor
Post-Condition	<ul style="list-style-type: none"><li>• Pelaksana, Leader, dan Supervisor melakukan login</li><li>• Pelaksana melakukan transaksi, Leader mengawasi</li><li>• Leader melakukan pengajuan barang, supervise mengawasi</li><li>• Leader dan Supervisi dapat melihat laporan</li><li>• Pelaksana, Leader, dan Supervisor melakukan logout</li></ul>
Relasi	Entend dari Login

## 4. Skenario Use Case

Aksi Aktor	Reaksi Sistem
1.Admin melakukan registrasi Pelaksana, Leader, dan Supervisor	
	2. Sistem menyimpan data registrasi
3. Pelaksana melakukan transaksi	
	4. Sistem harus memenuhi kaidah perubahan stok, dan Sistem harus menampilkan barang yang terdapat pada database
5.Pengguna harus melakukan pengambilan barang agar data logbook terisi dan total pengajuan barang terkalkulasi secara otomatis dan realtime	
	6. Sistem dapat menampilkan laporan logbook pengambilan barang, laporan stok peralatan, laporan stok pengharum & chemical, laporan stok tissue, dan laporan logbook pengambilan barang per periodik.

## 4. Skenario Use Case (Lanjutan)




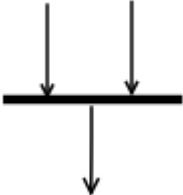
Aksi Aktor	Reaksi Sistem
7. Leader, dan Supervisor memilih menu laporan	
	10. Sistem dapat mengexport data laporan pengajuan barang, laporan penggunaan barang dan laporan logbook.
11. Pelaksana, Leader, dan Supervisor melakukan logout setelah selesai menggunakan aplikasi	
	12. Sistem keluar dari akun pengguna



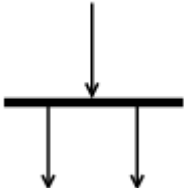
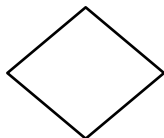

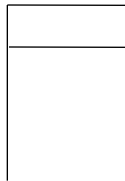
## 3. ACTIVITY DIAGRAM

- Activity Diagram adalah teknik untuk menggambarkan logika prosedural, proses bisnis, dan jaringan kerja antara pengguna dan sistem.
- Menggunakan notasi yang **mirip flowchart**, meskipun terdapat sedikit perbedaan notasi karena diagram ini mendukung behavior paralel.
- Activity diagram dibuat berdasarkan **sebuah atau beberapa use case** pada use case diagram
- Memungkinkan melakukan proses untuk memilih urutan dalam melakukannya atau hanya menyebutkan aturan-aturan rangkaian dasar yang harus diikuti, karena proses-proses sering muncul secara paralel.

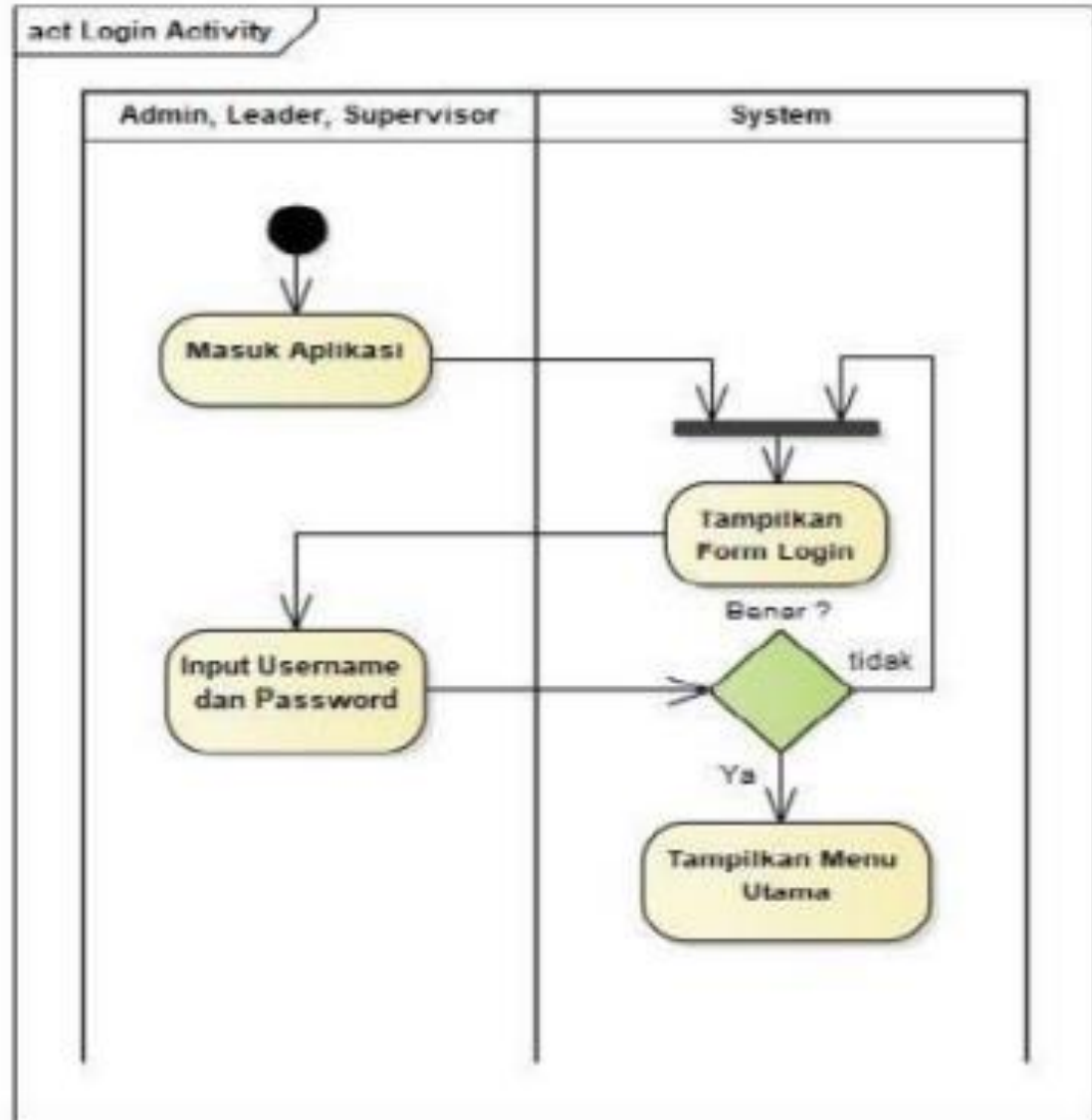
# Simbol-Simbol Activity Diagram

No	Nama	Simbol	Keterangan
1.	Start		<ul style="list-style-type: none"> <li>Menjelaskan awal proses kerja dalam activity diagram</li> <li>Hanya ada satu simbol start</li> </ul>
2.	End		<ul style="list-style-type: none"> <li>Menandai kondisi akhir dari suatu aktivitas dan merepresentasikan penyelesaian semua arus proses</li> <li>Bisa lebih dari satu simbol end</li> </ul>
3.	Activity		<ul style="list-style-type: none"> <li>Menunjukkan kegiatan yang membentuk proses dalam diagram</li> </ul>
4.	Join		<ul style="list-style-type: none"> <li>Menggabungkan dua atau lebih aktivitas bersamaan dan menghasilkan hanya satu aktivitas yang terjadi dalam satu waktu</li> </ul>

# Simbol-Symbol Activity Diagram

No	Nama	Simbol	Keterangan
5.	Fork		<ul style="list-style-type: none"> <li>Membagi aliran aktivitas tunggal menjadi beberapa aktivitas bersamaan</li> </ul>
6.	Decision		<ul style="list-style-type: none"> <li>Mewakili keputusan yang memiliki setidaknya dua jalur bercabang yang kondisinya sesuai dengan opsi percabangan</li> </ul>
7.	Connector		<ul style="list-style-type: none"> <li>Menunjukkan arah aliran atau aliran kontrol dari aktivitas</li> </ul>
8.	Swimlane		<ul style="list-style-type: none"> <li>Cara untuk mengelompokkan aktivitas berdasarkan aktor</li> <li>Menggunakan garis vertikal</li> </ul>

# Contoh Activity Diagram









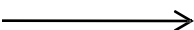
## 4. SEQUENCE DIAGRAM

- *Sequence diagram* menggambarkan interaksi antar objek di dalam dan di sekitar sistem (termasuk pengguna, *display*, dan sebagainya) berupa *message* yang digambarkan terhadap waktu.
- *Sequence diagram* menggambarkan interaksi yang fokusnya pada urutan pesan yang dipertukarkan, bersama dengan spesifikasi kemunculannya yang sesuai pada garis hidup.
- *Sequence diagram* terdiri atas dimensi vertikal (waktu) dan dimensi horizontal (objek-objek yang terkait).
- Diagram ini secara khusus berasosiasi dengan *use case diagram*

## Fungsi Sequence Diagram:

- a. Menentukan detail dari *Use Case*.
- b. Memodelkan logika prosedur, fungsi, atau operasi yang terdapat dalam sistem.
- c. Untuk melihat bagaimana objek dan komponen saling berinteraksi satu sama lain untuk menyelesaikan suatu proses.
- d. Merencanakan dan memahami fungsionalitas secara rinci dari skenario yang ada atau yang akan datang.

# Simbol *Sequence Diagram*

No	Simbol	Nama	Fungsi
1		Object	Komponen utama Sequence Diagram
2		Actor	Menggambarkan orang yang sedang berinteraksi dengan sistem
3		Entity Class	Menggambarkan hubungan kegiatan yang akan dilakukan
4		Boundary Class	Menggambarkan sebuah penggambaran dari form
5		Control Class	Menggambarkan penghubung antara boundary dengan tabel
6		Life Line	Menggambarkan tempat mulai dan berakhirnya sebuah message
7		Message	Menggambarkan pengiriman pesan

# Loop dan Kondisi

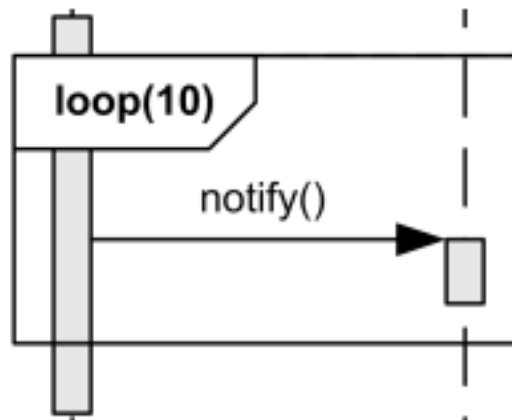
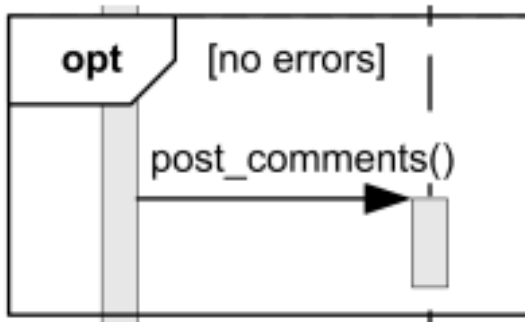
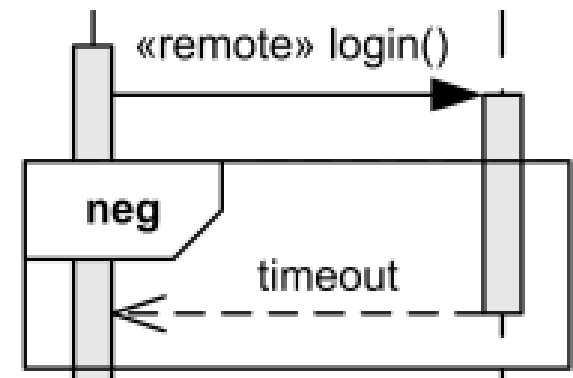
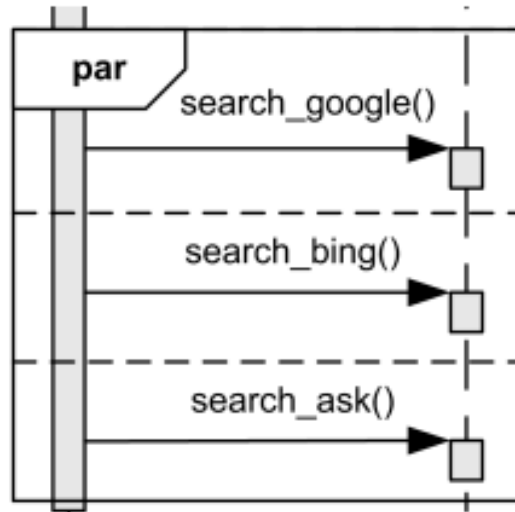
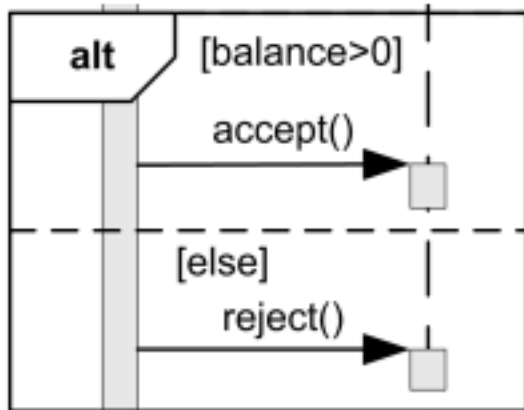
- *Loop* dan kondisi menggunakan kerangka interaksi, yaitu cara penandaan sebuah bagian *sequence diagram*.
- Kerangka terdiri dari beberapa daerah yang dipisahkan menjadi beberapa ***fragmen***.
- Setiap kerangka memiliki sebuah ***operator***.
- Setiap *fragmen* memiliki sebuah ***guard***.
- *Guard* merupakan sebuah ekspresi kondisional dalam tanda kurung [ ], dan menunjukkan bahwa pesan akan dikirimkan jika nilai *guard* benar.



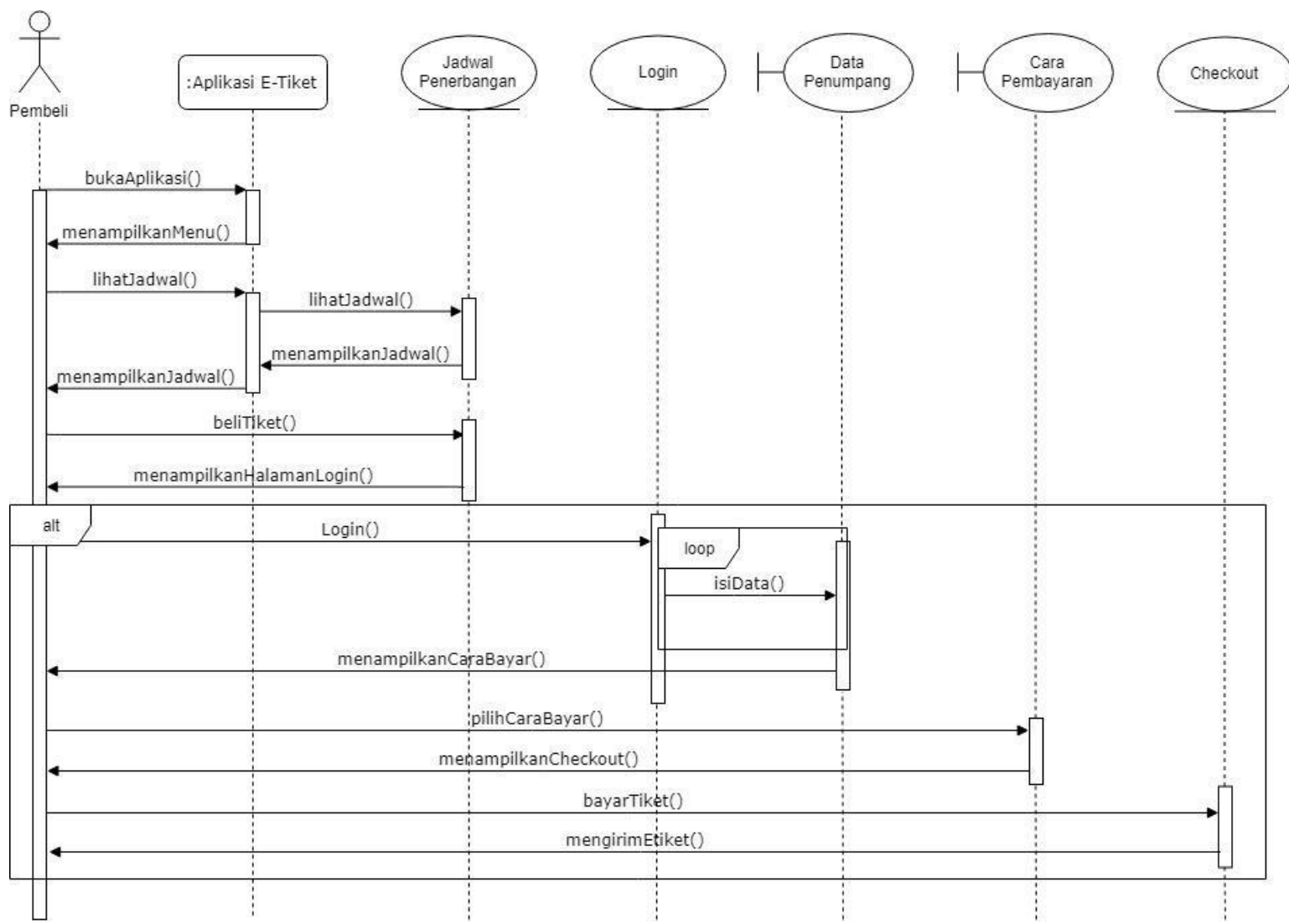
## Operator Umum untuk Kerangka Interaksi

Operator	Keterangan
alt	Alternatif dari banyak fragmen. <b>Hanya yang kondisinya true yang akan dijalankan</b>
opt	Optional; fragmen akan dijalankan <b>jika kondisi yang mendukungnya true</b>
par	Paralel; setiap fragmen dijalankan secara paralel
loop	Looping, fragmen mungkin dijalankan berulang kali dan guard menunjukkan basis iterasi
region	Critical region; fragmen hanya dapat mempunyai satu thread untuk menjalankannya
neg	Negatif; fragmen menunjukkan interaction yang salah
ref	Reference; menunjukkan ke sebuah interaction yang didefinisikan pada diagram yang lain
sd	Sequence diagram

# Contoh penggunaan operator:



# Contoh Sequence Diagram



# Latihan

Dosen memberikan contoh kasus, kemudian memodelkan kasus tersebut dengan diagram-diagram yang sudah dibahas