

***Mata Kuliah***  
***Analisis Desain Berorientasi Objek***  
***With UML***  
***(UNIFIED MODELLING LANGUAGE)***

***TIS12019***  
***Dosen Pengampu:***  
***Agus Wahyuddin, S.T., M.Kom***

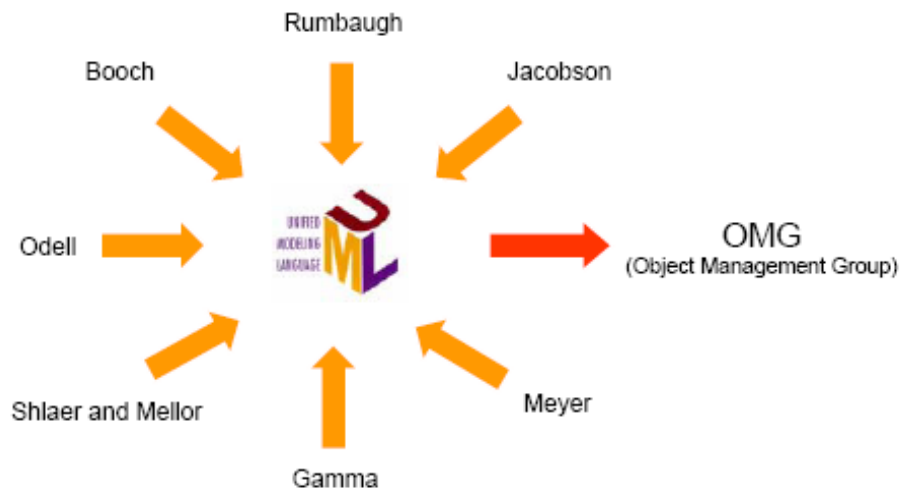
***FAKULTAS ILMU KOMPUTER***  
***TEKNIK INFORMATIKA***  
***TP 2020-2021***

## **UML (UNIFIED LANGUAGE MODELLING) Versi Fowler [2]**

- **UML** singkatan dari **Unified Modeling Language**, yaitu suatu metode modeling generasi ketiga.
- Penggunaan dari **UML** tidak terbatas hanya pada dunia software modeling, tetapi bisa pula digunakan untuk modeling hardware (**engineering systems**) dan sering digunakan sebagai modeling untuk proses bisnis dan juga modeling untuk struktur organisasi.
- **UML** adalah suatu metode terbuka yang digunakan untuk menspesifikasi, memvisualisasi, membangun dan mendokumentasikan artifak-artifak dari suatu pengembangan sistem software yang berbasis pada obyek.
- **UML** merupakan hasil kompilasi best engineering practice yang sudah terbukti sukses dalam pemodelan sistem yang besar, sistem yang kompleks, khususnya pada level arsitektural.

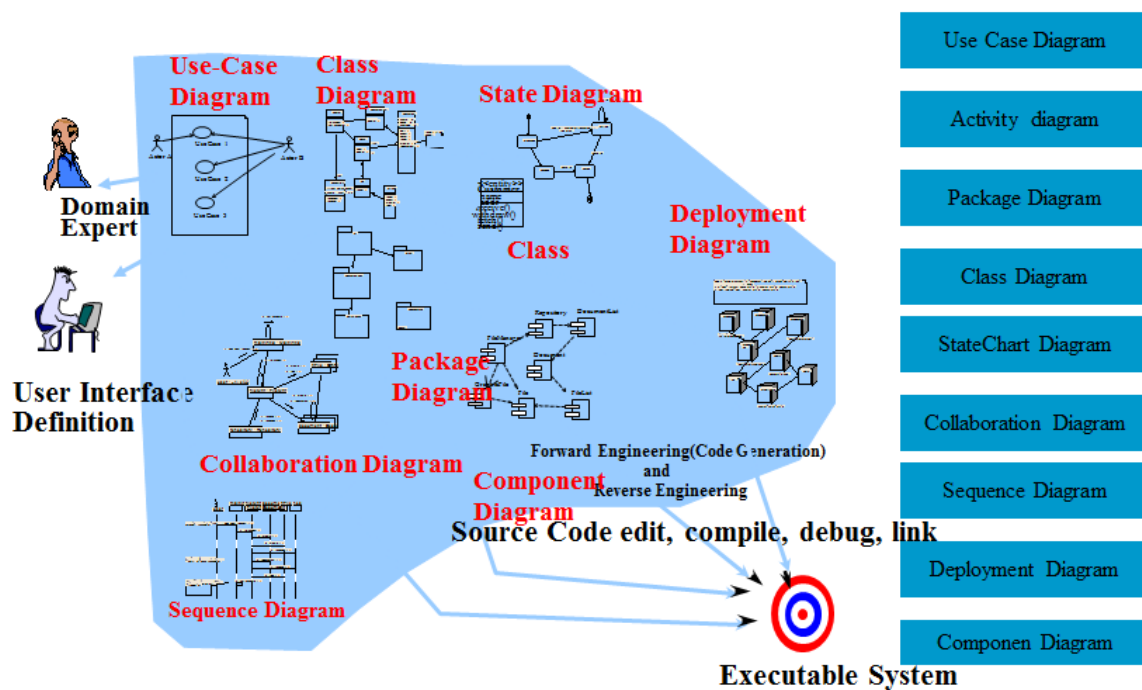
### **Sejarah**

- Dimulai pada bulan Oktober 1994 **Booch, Rumbaugh dan Jacobson**, yang merupakan tiga tokoh yang boleh dikata metodologinya banyak digunakan memelopori usaha untuk penyatuan metodologi pendesainan berorientasi objek. Pada tahun 1995 direlease draft pertama dari **UML (versi 0.8)**. Sejak tahun 1996 pengembangan tersebut dikoordinasikan oleh **Object Management Group** (OMG – <http://www.omg.org>). Tahun 1997 UML versi 1.1 muncul, dan saat ini versi terbaru adalah versi 1.5 yang dirilis bulan Maret 2003. Booch, Rumbaugh dan Jacobson menyusun tiga buku serial tentang UML pada tahun 1999 [7] [8] [9]. Sejak saat itulah UML telah menjelma menjadi standar bahasa pemodelan untuk aplikasi berorientasi objek.
- **UML** mengintegrasikan konsep **dari Booch, OMT, OOSE dan juga Class-Relation** dengan menggabungkan mereka menjadi suatu kesatuan bahasa modeling yang bisa berguna bagi siapa saja. UML bertujuan untuk menjadi standar bahasa modeling yang mampu untuk memodelkan sistem yang konkuren dan juga terdistribusi.



## ARTIFACT UML (BAGAN YANG TERDAPAT PADA UML)

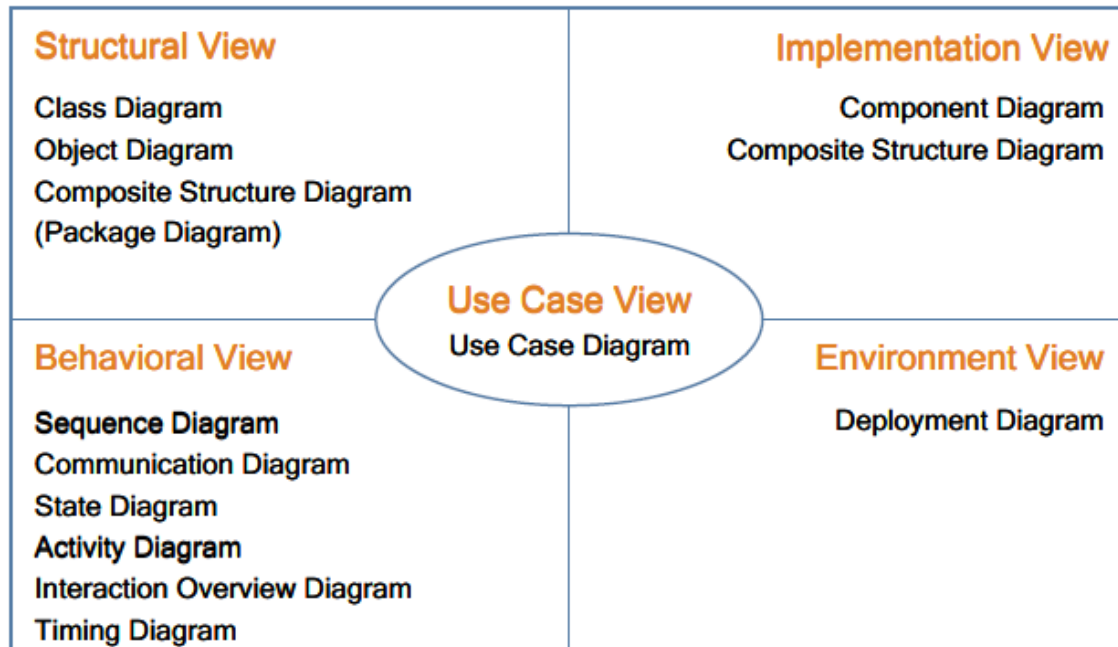
### Langkah – Langkah UML



**Arsitekture diagram pada UML Dibagi Dalam 4 + 1 model .:**

## UML Architectural Views and Diagrams

UML defines 13 diagrams that describe 4+1 architectural views



### **1. Structural View**

- (1) Class Diagram
- (2) Object Diagram
- (3) Composite Structure Diagram
- (4) Package Diagram

### **2. Behavior View**

- (1) Sequence Diagram
- (2) Communication Diagram
- (3) State Diagram
- (4) Interaction Overview Diagram
- (5) Timing Diagram

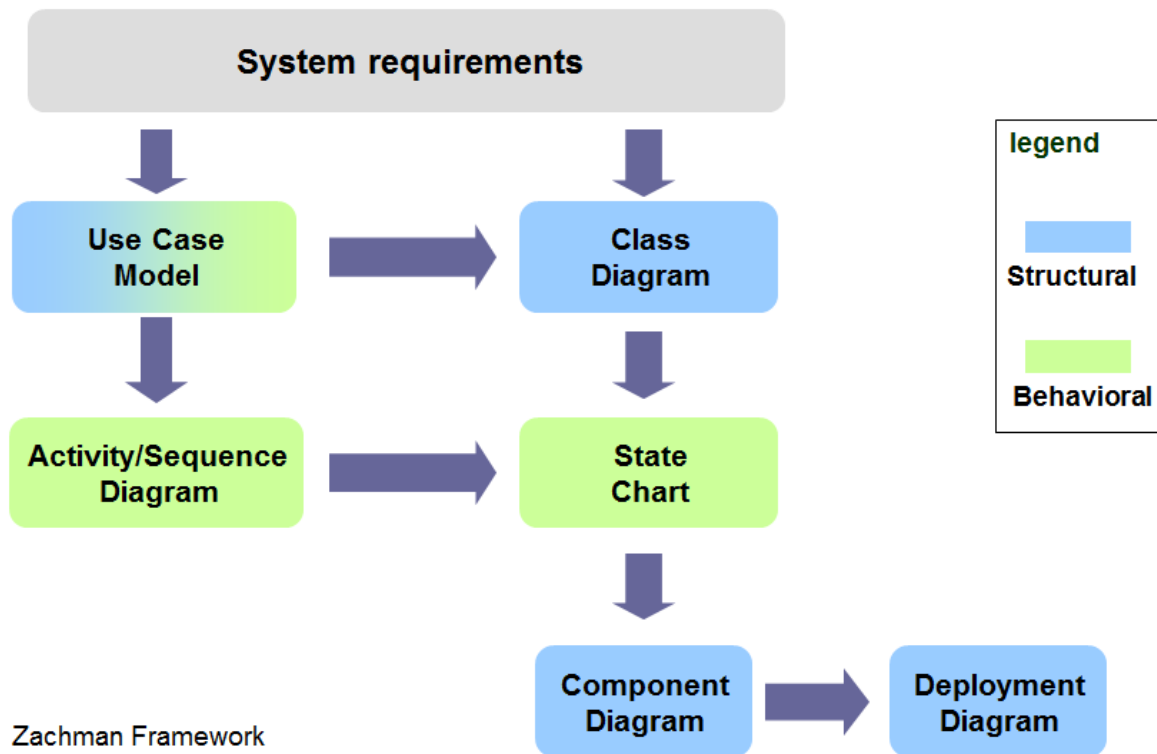
### **3. Implementation View**

- (1) Componen Diagram
- (2) Composite Struktur Diagram

### **4. Environmen View**

- (1) Deployment Diagram

# Analysis and Design Process



**Yang akan dipelajari dan diterapkan sebagai berikut :**

**1. Use Case**

- a. Use Case Bisnis
- b. Use Case System

**2. Skenario Use Case**

**Skenario Use Case Sistem**

**3. Aktiviti Diagram**

- a. Aktiviti Bisnis
- b. Aktiviti System
- c. Aktiviti Program

**4. Class Diagram**

**5. Sequence Diagram**

- a. Sequence Berbasis Class
- b. Sequence Berbasis Interface

## **CONTOH**

### **USE CASE DIAGRAM**

**Use Case** → Deskripsi fungsi dari sebuah system dari perspektif pengguna yang bekerja dengan cara mendeskripsikan tipikal interaksi antara user (pengguna) sebuah system dengan systemnya sendiri melalui sebuah cerita bagaimana sistem itu di pakai.

**Use Case** → *adalah alat bantu terbaik guna menstimulasikan pengguna potensial untuk mengatakan tentang suatu system dari sudut pandangnya.*

### **SKENARIO USE CASE**

**Skenario Use Case** → adalah urutan langkah-langkah yang menerangkan antara pengguna dengan system. Skenario mendeskripsikan urutan kejadian. Setiap urutan diinisialisasikan oleh orang, system yang lain, perangkat keras atau urutan waktu.

**Kesimpulan :**

**USE CASE** adalah serangkaian skenario yang digabungkan bersama-sama oleh tujuan umum pengguna.

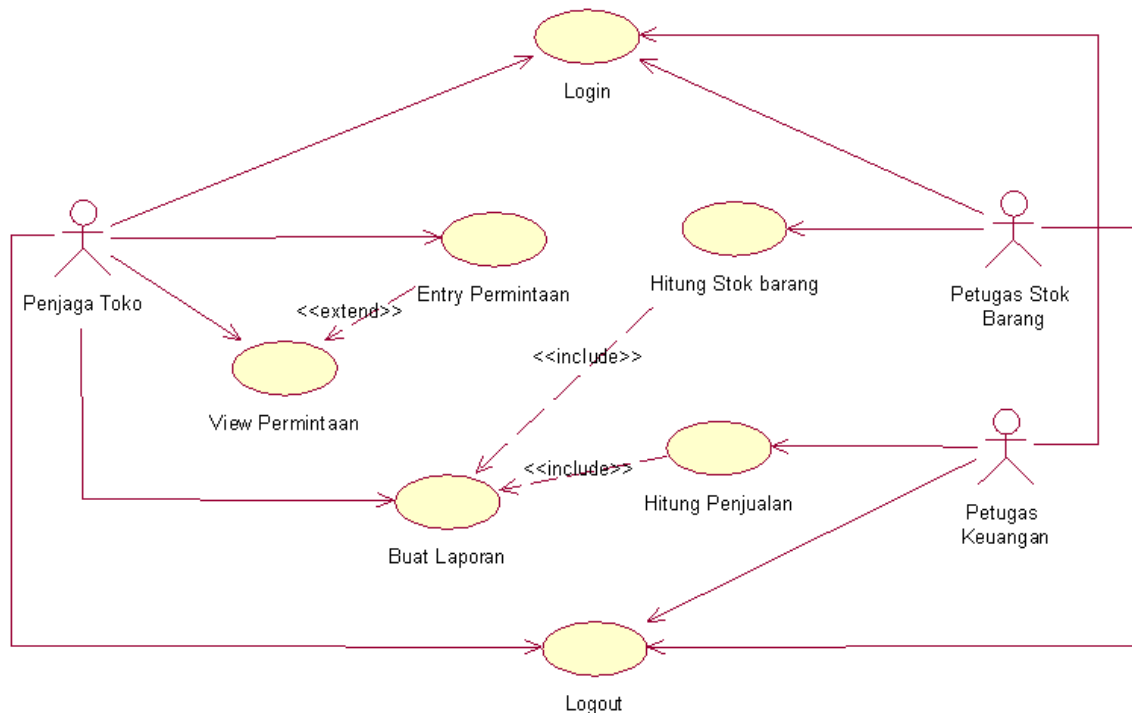
### **CONTOH PERANCANGAN UML : STUDY KASUS GALERRY VCD**

#### **Pendahuluan**

VCD House adalah sebuah galery yang bergerak dibidang retail dengan menjual produk-produk hiburan seperti VCD, DVD, CD PS, Accesories dan lain-lain. VCD House saat ini mempunyai cabang yang tersebar di berbagai mal di Bandung dan Jakarta.

Pada setiap cabang, VCD House menempatkan 2 orang petugas yang bekerja secara bergiliran untuk menjaga tokonya. Untuk mengaturnya VCD House memperkerjakan 2 orang bagian keuangan, 3 orang bagian pengontrolan stok dan 3 orang supervisor

## USE CASE SISTEM VCD HOUSE



## SKENARIO PROSES FUNGSI USE CASE

### Skenario Use Case Login

Fungsi ini memberikan hak akses setiap user untuk dapat menggunakan sistem dengan terlebih dahulu memasukkan login name dan password.

Identifikasi	
ID Use Case	VCDHOUSE.UC-01
Nama Use Case	Login
Tujuan	Memberikan hak akses ke sistem dengan validasi <i>userlogin</i> dan <i>password</i> .
Deskripsi	
Prioritas	Primary, Essential
Actor	Penjaga toko, petugas stok barang dan keuangan.
Skenario Utama	
Precondition	<ul style="list-style-type: none"> <li>Aktor memiliki hak akses</li> </ul>
Aksi Aktor	Reaksi Sistem
1. Memilih opsi <i>Login</i>	2. <i>Sistem</i> menampilkan isian nama login dan password. Untuk password diberikan tanda '*****' untuk alasan keamanan.
3. Memasukan <i>userlogin</i> dan <i>password</i> .	4. <i>Sistem</i> melakukan validasi dan verifikasi data pada database 5. <i>Sistem</i> menampilkan konfirmasi terhadap data yang dimasukan.
6. Aktor melakukan konfirmasi persetujuan terhadap data-data yang telah dimasukan	7. <i>Sistem</i> melakukan proses pembacaan basis data user. 8. <i>Sistem</i> menampilkan area akses user sesuai dengan status user.

<b>Post condition</b>	Jika pada akhir interaksi semua data valid maka sistem secara default akan menampilkan form-form di layar yang dapat diakses oleh user sesuai dengan hak otoritas yang diberikan Jika tidak valid maka sistem akan menampilkan pesan konfirmasi bahwa <b>“Akses ditolak!, cek kembali dan silahkan untuk mengulang lagi dengan memasukan data yang benar”</b> .
-----------------------	--

### Skenario Use Case Entry permintaan

Fungsi ini memfasilitas user untuk melihat kelengkapan barang atas dasar permintaan pelanggan yang di entry. Dengan tujuan untuk mengetahui barang apa saja yang paling banyak dicari dan barang apa saja yang stoknya sudah habis.

Identifikasi	
<b>ID Use Case</b>	VCDHOUSE.UC-2
<b>Nama Use Case</b>	Entry Permintaan
<b>Tujuan</b>	Untuk mengetahui barang apa saja yang sering dicari dan barang apa saja yang sudah tidak ada.
Deskripsi	
<b>Prioritas</b>	<b>Primary</b>
<b>Actor</b>	Penjaga Toko
Skenario Utama	
<b>Pre Condition</b>	<ul style="list-style-type: none"> <li>• Sudah terdaftar sebagai user.</li> <li>• Sudah melakukan login terhadap sistem</li> </ul>
Aksi Aktor	Reaksi Sistem
1. Memilih opsi <i>entry permintaan</i>	2. <i>Sistem</i> menampilkan isian yang meliputi nama barang, jenis dan stok barang.
3. Memasukan data-data barang permintaan.	4. <i>Sistem</i> melakukan validasi dan verifikasi data-data yang dimasukan dan menghitung stok secara otomatis. 5. <i>Sistem</i> menampilkan konfirmasi persetujuan.
6. melakukan persetujuan atas data-data yang telah dimasukan	7. <i>Sistem</i> menampilkan data-data yang dimasukan di layar. 8. <i>Sistem</i> menyimpan data-data dalam database.
<b>Post Condition</b>	Jika data-data yang dimasukan valis, maka sistem akan menampilkan data tersebut dilayar dan data akan disimpan dalam basis data. <b>Jika tidak valid akan tampil pesan “Cek kembali data-data yang dimasukan dan silahkan ulangi!”</b> .

### Activity Diagram

Adalah teknik untuk mendeskripsikan logika prosedural, proses bisnis dan aliran kerja dalam banyak kasus. Activity Diagram mempunyai peran seperti halnya flowchart, perbedaanya adalah Activity diagram dapat melakukan perilaku paralel sedangkan flowchart tidak.

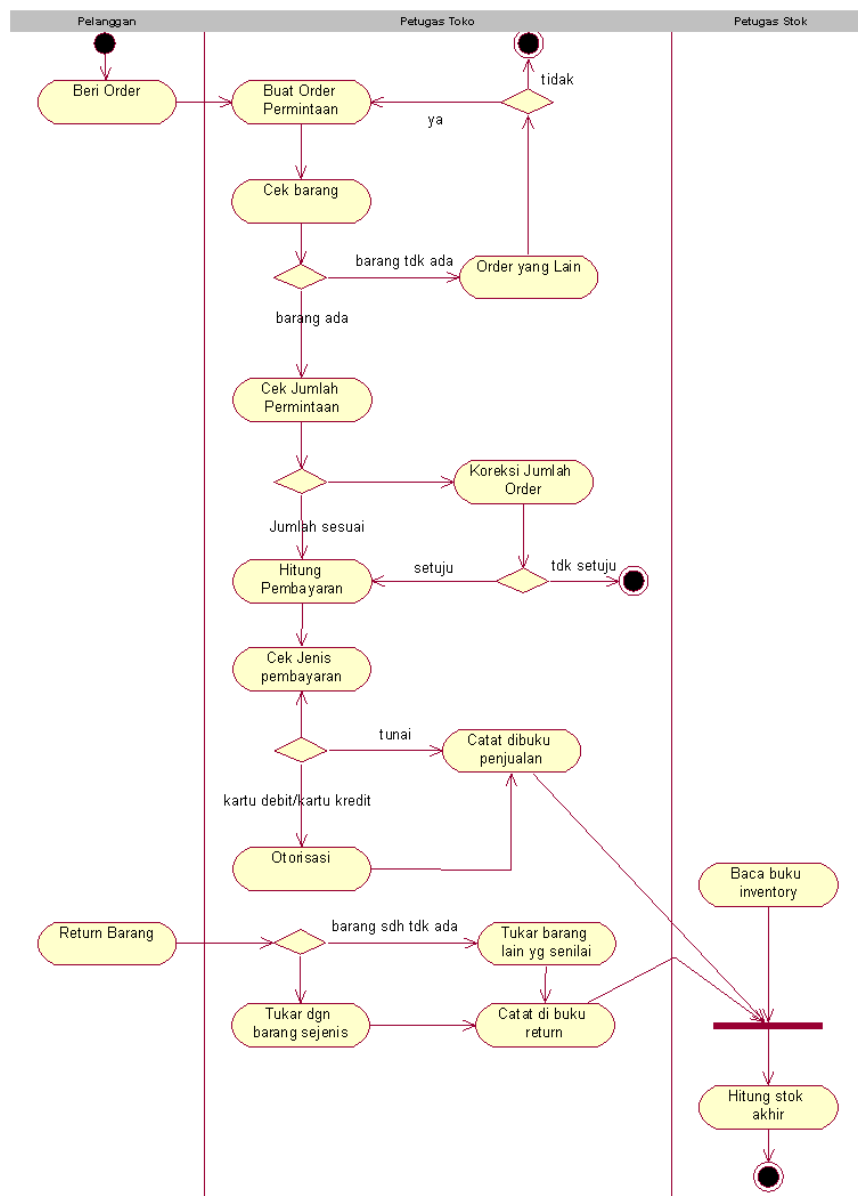


Berikut simbol-simbol Activity Diagram :

Simbol	Keterangan
●	Titik Awal
○	Titik Akhir
□	Activity
◇	Pilihan Untuk mengambil Keputusan
—	Fork; Digunakan untuk menunjukkan kegiatan yang dilakukan secara parallel atau untuk menggabungkan dua kegiatan paralel menjadi satu.

Prosedur berjalan penjualan akan digambarkan dengan **Activity Diagram** sebagai berikut

:



## CLASS DIAGRAM

**Class Diagram** sangat membantu dalam visualisasi struktur kelas dari suatu sistem. Hal ini disebabkan class adalah deskripsi kelompok objek-objek dengan property, perilaku (operasi) dan relasi yang sama.



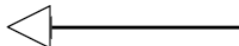
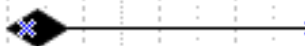
Sebuah *Class* memiliki tiga area pokok :

1. **Nama**, merupakan nama dari sebuah kelas
2. **Atribut**, merupakan peroperti dari sebuah kelas. Atribut melambangkan batas nilai yang mungkin ada pada obyek dari class
3. **Operasi**, adalah sesuatu yang bisa dilakukan oleh sebuah *class* atau yang dapat dilakukan oleh *class* lain terhadap sebuah *class*.

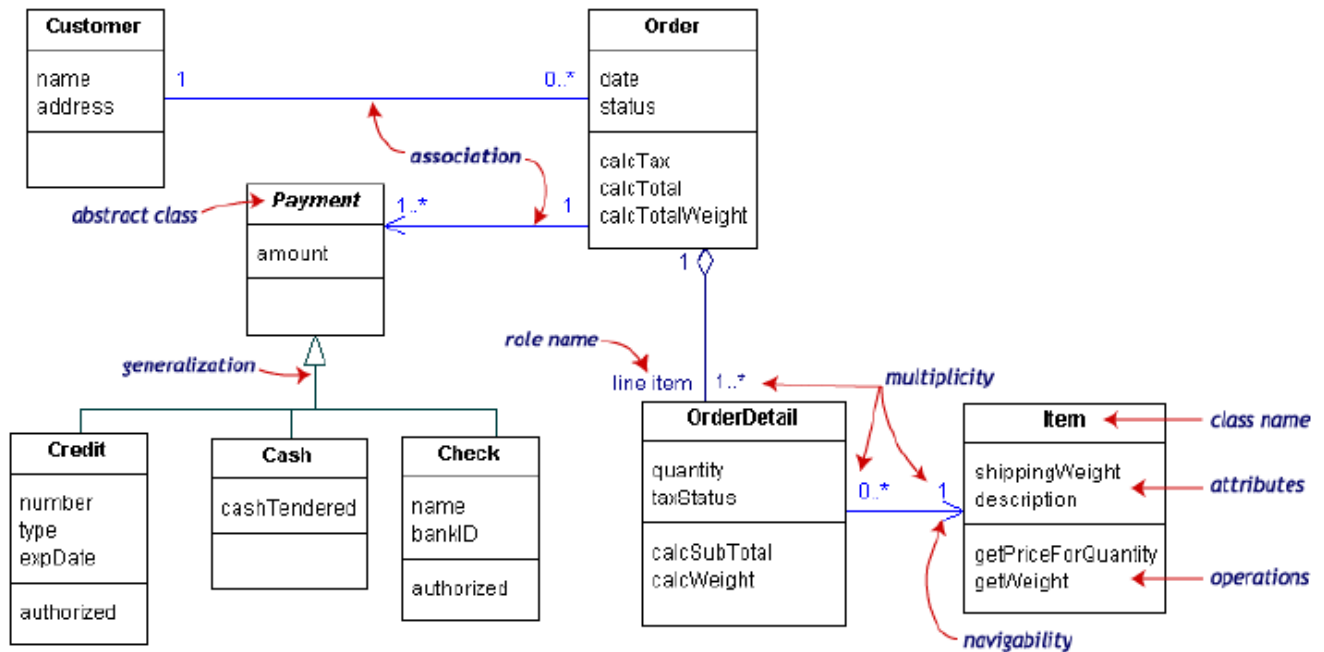
Atribut dan metoda dapat memiliki salah satu sifat berikut :

- **Private**, tidak dapat dipanggil dari luar *class* yang bersangkutan.
- **Protected**, hanya dapat dipanggil oleh *class* yang bersangkutan dan anak-anak yang mewarisinya.
- **Public**, dapat dipanggil oleh siapa saja.
- **Package**, hanya dapat dipanggil oleh instance sebuah class pada paket yang sama.

Berikut adalah tabel notasi – notasi yang ada pada class diagram :

<b>Dependency</b>	Kadangkala sebuah <i>class</i> menggunakan <i>class</i> yang lain. Hal ini disebut <i>dependency</i> . Umumnya penggunaan <i>dependency</i> digunakan untuk menunjukkan operasi pada suatu <i>class</i> yang menggunakan <i>class</i> yang lain. Sebuah <i>dependency</i> dilambangkan sebagai sebuah panah bertitik-titik.	
<b>Aggregation</b>	<i>Aggregation</i> mengindikasikan keseluruhan bagian <i>relationship</i> dan biasanya disebut sebagai relasi “mempunyai sebuah” atau “bagian dari”. Sebuah <i>aggregation</i> digambarkan sebagai sebuah garis dengan sebuah jajaran genjang yang tidak berisi/tidak solid.	
<b>Generalization</b>	Sebuah relasi <i>generalization</i> sepadan dengan sebuah relasi <i>inheritance</i> pada konsep berorientasi obyek. Sebuah <i>generalization</i> dilambangkan dengan sebuah panah dengan kepala panah yang tidak solid yang mengarah ke kelas “parent”-nya/induknya.	
<b>Composite</b>	sebuah tipe agregasi yang kuat dimana bagian dari proyek bergantung secara keseluruhan obyek. Sedemikian kuatnya hubungan ini, bila sebuah obyek composite dibuang, maka bagian yang tergantung pada komponen tersebut akan terbuang juga	

## Contoh Class Diagram

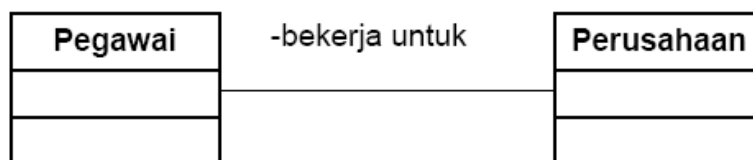


## Relasi pada Class Diagram

Relasi atau *relationship* merupakan keterhubungan antar kelas yang muncul pada saat sebuah kelas berinteraksi dengan kelas-kelas lainnya.

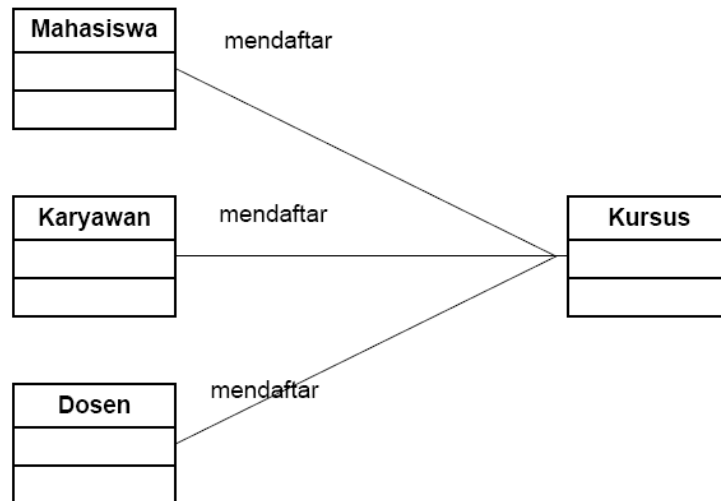
### a) Association (asosiasi).

Asosiasi adalah pada saat beberapa kelas saling terhubung satu sama lain secara konseptual.



Gambar 2.8. Contoh sebuah asosiasi

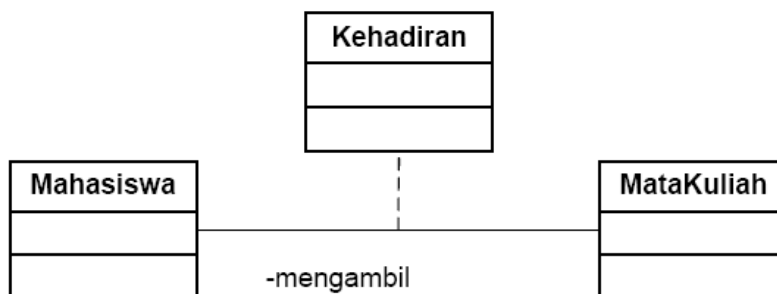
Asosiasi juga dapat menjadi lebih kompleks pada saat beberapa *class* terhubung ke satu *class*, seperti yang terlihat pada gambar berikut :



Gambar Asosiasi *class to class*

### b) *Associations Class*

Sebuah asosiasi dapat memiliki atribut dan operasi seperti halnya sebuah *class*. Sebuah *association class* sebenarnya diperlukan apabila salah satu dari kelas yang terhubung mempunyai sebuah atau beberapa atribut yang tidak layak dimiliki oleh kelas tersebut, karena secara logis atribut tersebut lebih layak dimiliki oleh asosiasi yang menghubungkan kedua kelas tersebut. Berikut adalah contoh sebuah *association class*.



Gambar Contoh *Association Class*

### c) *Multiplicity*

*Multiplicity* atau multiplisitas adalah jumlah banyaknya obyek sebuah *class* yang berelasi dengan sebuah obyek lain pada *class* lain yang berasosiasi dengan *class* tersebut. Untuk menyatakan multiplisitas anda dapat meletakkannya diatas garis asosiasi berdekatan dengan *class* yang sesuai. Tabel berikut menjabarkan multiplisitas yang dapat digunakan.

Tabel Nilai multiplisitas

Potencial Multiplicity Values	
Indicator	Meaning
0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
N	Only $n$ (where $n > 1$ )
0..n	Zero to $n$ (where $n > 1$ )
1..n	One to $n$ (where $n > 1$ )

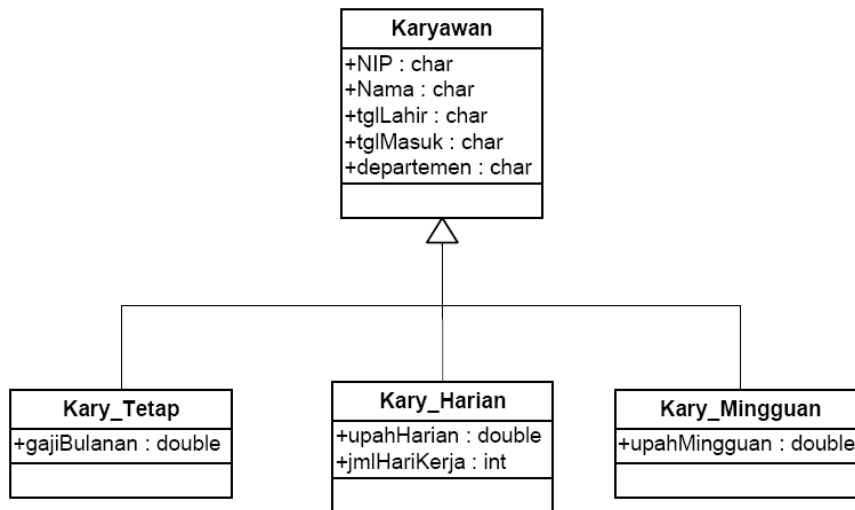


Gambar Asosiasi dengan multilisitas

#### **d) Generalization & Inheritance**

Sebuah *class* (*child class* atau *subclass*) dapat mewarisi atribut-atribut dan operasi-operasi dari *class* lainnya (*parent class* atau *super class*) dimana *parent class* bersifat lebih umum daripada *child class*.

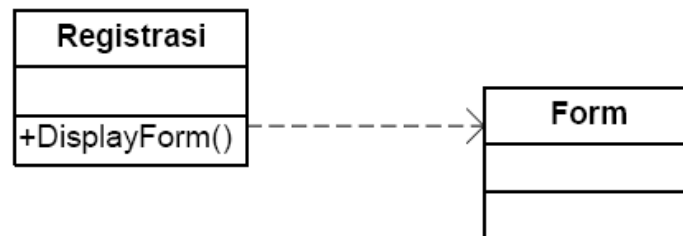
Generalisasi pada konsep *Object Oriented* digunakan untuk menjelaskan hubungan kesamaan diantara *class*. Dengan menggunakan generalisasi bisa dibangun struktur logis yang bisa menampilkan derajat kesamaan atau perbedaan diantara *class-class*.



Gambar Generalisasi dengan 3 subclass

#### e) Dependant Class

**Dependency** digunakan untuk menunjukkan operasi pada suatu class yang menggunakan class yang lain. Notasi untuk dependency pada UML dapat menggunakan garis putus-putus dan tanda panah pada ujungnya

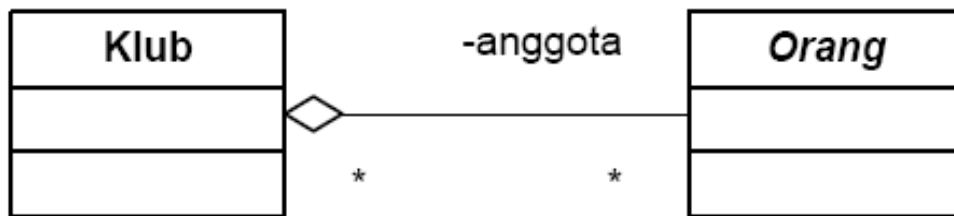


Gambar Dependency

#### f) Agregasi

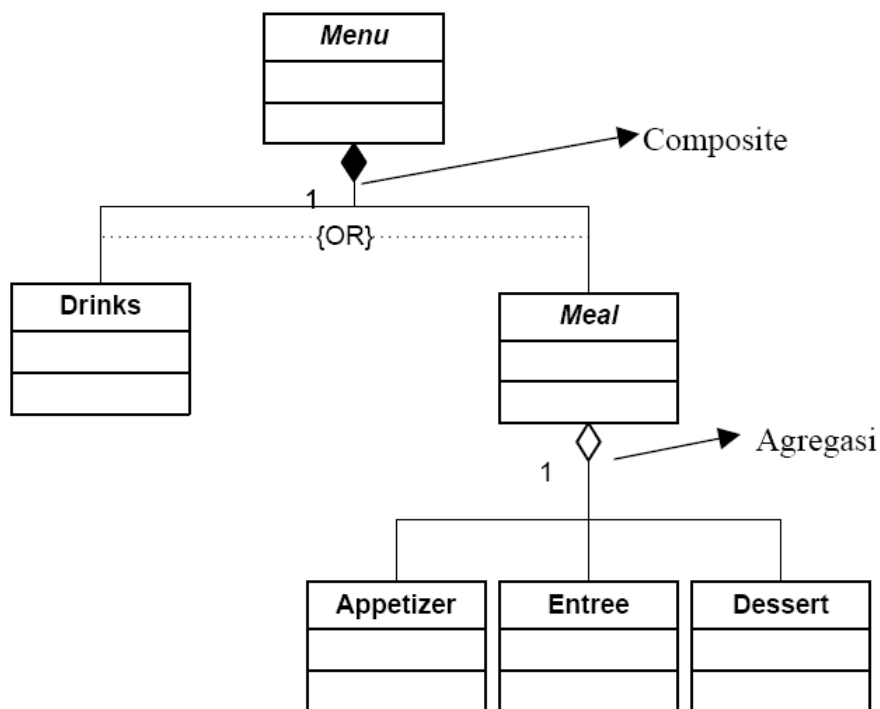
Didalam UML, ada relasi dengan perlakuan khusus yang disebut dengan “**bagian dari (part of)**” yang menangani antar obyek-obyek dimana salah satunya adalah bagian dari yang lain. Hal seperti ini di dalam UML disebut Agregasi.

Sebuah agregasi adalah kasus khusus dari asosiasi. Agregasi disimbolkan dengan jajaran genjang yang diletakkan pada class yang mengandung obyek



Gambar Agregasi

Kadangkala relasi OR harus digunakan pada agregasi. Untuk memodelkan hal tersebut, sebuah *constraint* – dengan kalimat atau dalam tanda kurung kurawal pada garis putus-putus harus dibuat untuk menghubungkan 2 obyek seperti gambar berikut :



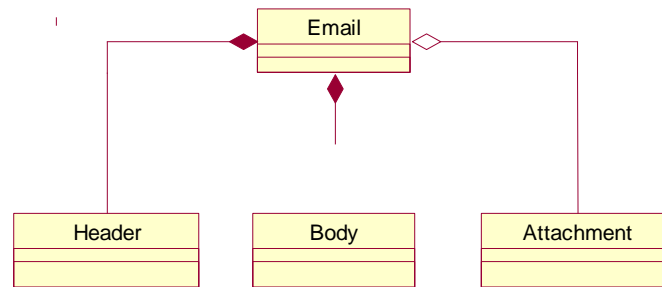
Gambar Agregasi dengan constraint dan komposit

#### g. Composite

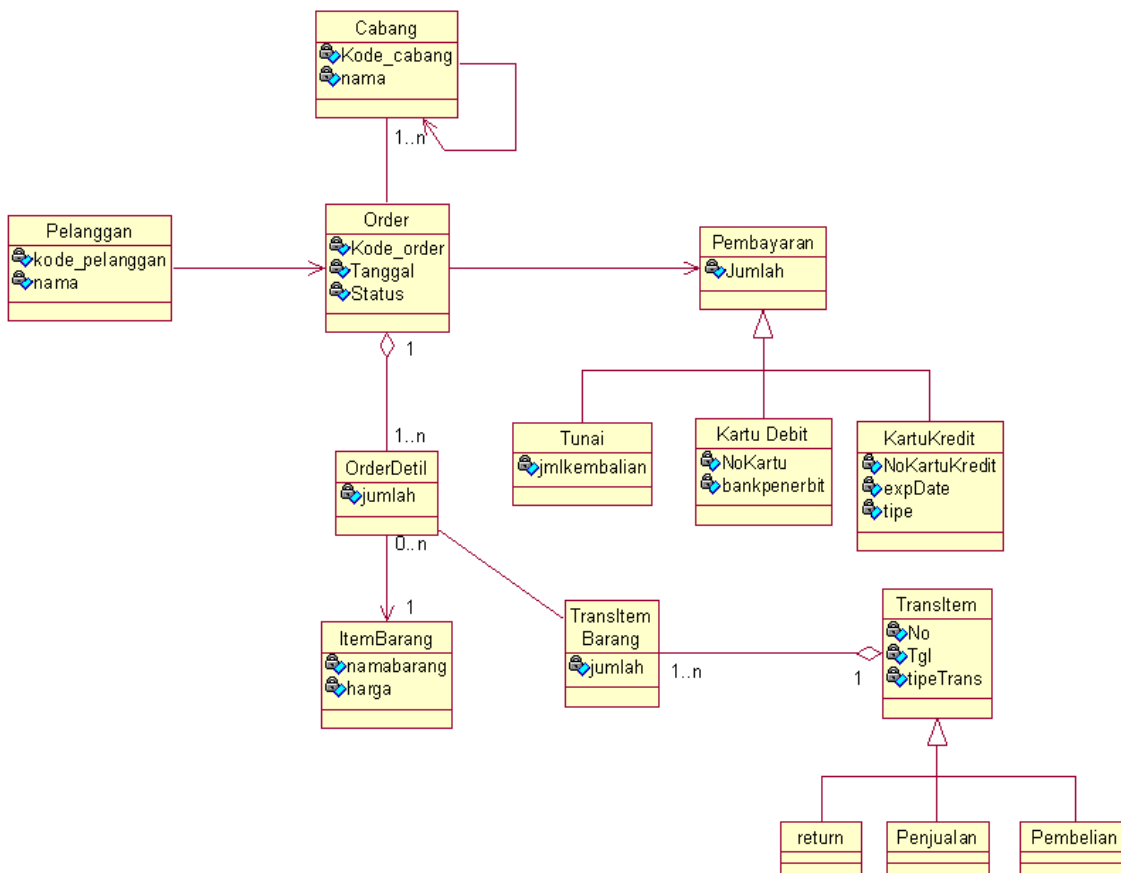
Adalah sebuah tipe agregasi yang kuat dimana bagian dari proyek bergantung secara keseluruhan obyek. Sedemikian kuatnya hubungan ini, bila sebuah obyek composite dibuang, maka bagian yang tergantung pada komponen tersebut akan terbangun juga.



Contoh :



## Class Diagram **GALLERY VCD**



### KETERANGAN :

- Order diasosiasikan dengan pelanggan yang melakukan pembelian dan pembayaran.
- Pembayaran bisa dilakukan dengan 3 cara : Tunai, Debit dan Kartu Kredit.
- Order mengandung OrderDetil dimana OrderDetil memperlihatkan barang apa saja yang dibeli pelanggan. Oleh karena itu OrderDetil diasosiasikan dengan ItemBarang.
- Sistem pembayaran digeneralisasikan dari class pembayaran.

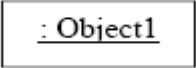



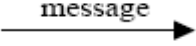
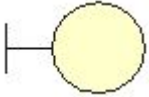
- Pembelian barang tidak disebutkan adanya kebutuhan secara spesifik untuk menangani pembelian barang dan juga return, maka diasumsikan kedua tipe ini dapat digeneralisasikan dari clas TransItem.
- Dengan digeneralisasikan TransItem menjadi Return, Penjualan dan Pembelian, maka perhiungan stok akhir akan mudah dilakukan

## SEQUENCE DIAGRAM

**Sequence diagram** mendokumentasikan komunikasi/interaksi antar kelas-kelas. Diagram ini menunjukkan sejumlah obyek dan *message* (pesan) yang diletakkan diantara obyek-obyek didalam *use case*

*Sequence Diagram* menunjukan bagaimana detil operasi dilakukan – pesan apa yang dikirim dan kapan. Berikut adalah notasi-notasinya :

Tabel Notasi *Sequence Diagram*

<b>Object</b>	Object merupakan instance dari sebuah class dan dituliskan tersusun secara horizontal. Digambarkan sebagai sebuah class (kotak) dengan nama obyek didalamnya yang diawali dengan sebuah titik koma.	
<b>Actor</b>	Actor juga dapat berkomunikasi dengan object, maka actor juga dapat diurutkan sebagai kolom. Simbol Actor sama dengan simbol pada Actor Use Case Diagram.	
<b>Lifeline</b>	Lifeline mengindikasikan keberadaan sebuah object dalam basis waktu. Notasi untuk Lifeline adalah garis putus-putus vertikal yang ditarik dari sebuah obyek.	
<b>Activation</b>	Activation dinotasikan sebagai sebuah kotak segi empat yang digambar pada sebuah lifeline. Activation mengindikasikan sebuah obyek yang akan melakukan sebuah aksi.	
<b>Message</b>	Message, digambarkan dengan anak panah horizontal antara Activation. Message mengindikasikan komunikasi antara object-object.	
<b>Boundary</b>	Adalah sistem atau interface yang menjembatani antar aktor dengan database.	

Berikut adalah *sequence diagram* dari kasus VCDHOUSE GALERY :

